البرمجة بلغة البايثون

إعداد

أعضاء هيئة تدريس

كلية الحاسبات والمعلومات

2021/2022

معلومات عن الكتاب:

الكلية: الحاسبات والمعلومات

التخصص: تدرس لأقسام علوم الحاسب ونظم المعلومات

العام الدراسي: 2022/2021

عنوان الكتاب: البرمجة بلغة البايثون

عدد الصفحات: 108

إعداد: أعضاء هيئة تدريس بكلية الحاسبات والمعلومات

مقدمة

أخي القارئ هذه مذكرة بسيطة لشرح وتبسيط أساسيات ومبادئ البرمجة والتعرف على لغة البايثون

وتهدف هذه المذكرة إلى فهم أساسيات البرمجة وتعود الطالب على التفكير المنطقي المرتب لحل المسائل والوصول إلى الحل الأمثل من خلال كتابة المخططات الانسيابية أو كود الشفرة ويحتوي المذكرة على خمسة فصول.

الفصل الأول بعنوان مقدمة حيث أنه يعرض مقدمة عامة عن البرمجة من خلال تعريف البرمجيات وخصائصها وأنواع الأخطاء فيها والعناصر المكونة للغة البرمجة. كما يقدم أيضا مقدمة عن أنواع البيانات والتعليمات والعمليات التي تستخدم في البرمجة وتعريف المتغيرات والثوابت وذلك بالإضافة إلى القواعد العامة المتعلقة بأنواع البيانات وأسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة.

الفصل الثاني بعنوان الخوارزميات: المفهوم والخصائص وطرق الصياغة ويحتوي على المراحل الأربعة الأساسية لحل المسائل (تعريف المسألة وتحليلها والبرمجة والتوثيق) ومفهوم الخوارزميات.

الفصل الثالث بعنوان تمثيل البيانات داخل الحاسب – البوابات المنطقية حيث يحتوى على شرح للأنظمة العدية (العشرى – الثنائى – الثمانى – السادس عشر)- شرح للبوابات المنقية والجبر بوليان.

الفصل الرابع بعنوان مقدمة فى لغة البايثون : يركز هذا الفصل على لغة االبايثون وهى الموضوع الرئيسي للدراسة والخطوات الرئيسية لكتابة برنامج بلغة االبايثون.

الفصل الخامس بعنوان الجمل الشرطية والتحكم فى لغة البايثون.

الفصل السادس بعنوان الدوال حيث يقوم بشرح الدوال الجاهزة فى لغة البايثون – كيفية عمل دالة فى لغة البايثون.

الفصل السابع تطبيقات عملية متنوعة.

في هذا الفصل سنتعرف على مقدمة عامة عن البرمجة من خلال تعريف البرمجيات وخصائصها وأنواع الأخطاء فيها والعناصر المكونة للغة البرمجة. كما سنتعرف على مقدمة عن أنواع البيانات والتعليمات والعمليات التي تستخدم في البرمجة وتعريف المتغيرات والثوابت والقواعد وذلك العامة المتعلقة بأنواع البيانات وأسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة بالإضافة إلى جملة من الأمثلة والتمارين التطبيقية لمحتويات الفصل

1.1) مقدمه عن البرمجيات INTRODUCTION TO SOFTWARE

في هذا الفصل سنقوم بعرض مقدمة عامة عن البرمجة من خلال تعريف البرمجة وخصائصها وأنواع الأخطاء بها والعناصر المكونة للغة البرمجة وذلك بالإضافة إلى مقدمة عامة عن أنواع البيانات والقواعد المتعلقة بها وأنواع التعليمات والعمليات المستخدمة في البرمجة وتعريف المتغيرات والثوابت وأسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة

1.1.1)تعريف البرمجيات Software definition

هي مجموعة من التعليمات (instructions) التي يتم استخدامها في بناء البرنامج وتقوم المكونات المادية للحاسب (الذاكرة – المعالج – وحدات الإدخال والإخراج) بتنفيذها لتؤدي مهام ووظائف معينه وتكتب هذه التعليمات بأحد اللغات المستخدمة في البرمجة مثل python, Fortran, Basic, Cpp , and Java

1.1.2)خصائص البرمجيات Characteristics of Software

تتميز البرمجيات بالخصائص التالية:

1) لها بداية

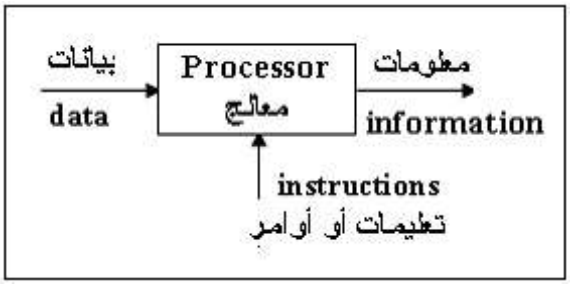
2) لها نهاية
3) لها جسم
4) تحقق التسلسل المنطقي للتعليمات
5) الدقة في التصميم
6) لها مدخلات (data – بيانات) ومخرجات (information - معلومات)

المعلومات (information) هي بيانات تمت معالجتها وأحيانا تسمى (processed data).

الشكل رقم 1 يوضح عملية الحصول على المعلومات من خلال معالجة البيانات بواسطة التعليمات (instructions)

1.1.3)أنواع الأخطاء البرمجية Types Of Software Errors

تنقسم الأخطاء البرمجية إلى أخطاء لغوية (syntax errors) وأخطاء منطقية (logical errors)



شكل 1: عملية الحصول على المعلومات من خلال معالجة البيانات بواسطة التعليمات

1) الأخطاء اللغوية (syntax errors):

هي أخطاء تعتمد علي اللغة المستخدمة وقواعدها ويمكن كشف هذه الأخطاء بواسطة استخدام المعالجات (compilers) أو المجمعات (assemblers) أو المفسرات (interpreters) .

فعلى سبيل المثال لغة C تشترط وضع (;) في نهاية الجملة على النحو التالي:

A = B + C ;

فإذا لم نضع هذه العلامة في نهاية الجملة فسيكون هناك خطأ لغوي و في العبارة التالية أيضا يوجد خطأ لغوى حيث وجود علامتي الجمع والطرح بصورة متتالية وهو منافي لقواعد العمليات الحسابية

```
A = A+-C ;
```

2) الأخطاء المنطقية (logical errors) :

هى أخطاء في طريقه كتابة المعادلات وإجراء الحسابات ويمكن اكتشافها باستخدام طرق المحاكاة (Simulation) فعلى سبيل المثال يريد المبرمج أن يوجد مجموع قيمتي المتغيرين A and B ولكنه استخدم العبارة:

```
C = A – B
```

بدلا من العبارة

```
C = A + B
```

لن يخبره المترجم عن وجود خطأ وبالتالي يجب استخدم طرق المحاكاة (Simulation) لاكتشاف هذا الخطأ.

بصورة مبسطة عملية المحاكاة (Simulation) تعني إدخال بيانات محددة معلوم مسبقا قيمة ناتج البرنامج لها فإذا كانت مخرجات البرنامج تساوي الناتج المعرف مسبقا كان البرنامج خاليا من الأخطاء المنطقية وإذا كان هناك اختلاف بين الناتج المعرف مسبقا ومخرجات البرنامج يكون هناك خطأ منطقي وبالتالي يجب مراجعة عبارات وخطوات البرنامج وإجراء المحاكاة مرة أخري حتى يتم التأكد من أن البرنامج خاليا من الأخطاء المنطقية.

1.1.4)مكونات لغة البرمجة Software Compositions

تتكون لغة البرمجة من مجموعة من القواعد والمصطلحات يستخدمها المبرمج لكتابة برنامج معين يفهمه الحاسب حسب نوعه ويتم تحويل البرنامج طبقاً للغة المكتوب بها إلي لغة الآلة (machine language) بواسطة المعالجات compilers أو المجمعات assemblers

1.1.5) العناصر الأساسية للغة البرمجة

The Main Element of Software Languages

تتكون لغات البرمجة من العناصر التالية:

a) مجموعة من الرموز الأساسية

1) حروف لاتينية : A, B, C, ….., X, Y, Z

2) أرقام عربية : 0,1,2,3,4,…… 9

3) رموز خاصة : $, % , / , * , - , +

b) الكلمات

وهي نوعان:

1) **كلمات محجوزة (reserved words):** هي كلمات لها معني خاص بالحاسب وتقوم بتنفيذ مهام محددة ويوجد قيود عليها حسب لغة المبرمجة المستخدمة مثل:

Read, Write, Add, for, and if

2) **كلمات يختارها المستخدم أو المبرمج (user defined words)** : هي كلمات يختارها المبرمج لتمثيل أسماء المتغيرات والثوابت ولا يوجد قيود على استخدامها مثل :

Alpha, A, ABD, sum, etc.

c) مجموعة التعليمات Instructions set

هي مجموعة من الرموز أو الكلمات الخاصة بلغة البرمجة لتنفيذ عملية أو مهام أو أمر معين وتنقسم إلى ثلاثة أنواع:

1) تعليمات خاصة بالذاكرة (memory reference instructions)

وتقوم غالبا بتنفيذ العمليات الحسابية أو المنطقية مثل: ADD, SUB, *, /, +, etc. فعلى سبيل المثال العبارة

C = A + B

تعنى جمع محتوى الذاكرة الموجودة في العنوان A مع محتوى الذاكرة الموجود في العنوان B ووضع الناتج في العنوان C داخل الذاكرة

2) تعليمات إدخال وإخراج (Input/ output instructions)

هي تعليمات لإدخال البيانات وإخراج المعلومات مثل: Write and Read. فعلى سبيل المثال:

Read (X, Y, Z)

Write (A, B, C)

العبارة الأولى تعني إدخال قيم المتغيرات (X, Y and Z) من خلال وحدة الإدخال والعبارة الثانية تعني إخراج قيم المتغيرات (A, B and C) إلى وحدة الإخراج

3) تعليمات تحكم (Control instructions):

تشمل التعليمات التي تؤدي إلي تسلسل تنفيذ البرنامج مثل

Goto, While, Do-while, switch-case, for …etc.

عندما يفهم الحاسب هذه التعليمات تقوم وحده التحكم بالجهاز (control unit) بإرسال الإشارات أو أوامر التحكم اللازمة إلي الوحدات المادية المعنية بذلك لتنفيذ هذه التعليمات

يتم تنفيذ كل تعليمة من خلال ثلاث مراحل أساسية :

1) مرحلة استحضار العبارة التي تحتوى على التعليمة من ذاكرة الحاسب (Fetch)
2) مرحلة تفسير هذه التعليمة لفهم مدلولها (decoding)
3) مرحلة تنفيذ هذه التعليمة (execution)

مثال: 1

A = A + B

لتنفيذ العبارة السابقة يتم إتباع التالي

1) الذهاب إلى ذاكرة الحاسب لاستحضار العبارة السابقة حيث أنها تكون مخزنة مسبقا في ذاكرة الحاسب بلغة الآلة (machine language)

2) تفسير التعليمة (instruction) الموجودة بالعبارة وإجراء التالي:

- صدور أمر بواسطة وحدة التحكم بالحاسب (control unit) للذهاب إلي الذاكرة في العنوان A لمعرفة محتواه وإرساله إلى وحدة العمليات الحسابية والمنطقية (Arithmetic and logic unit – ALU)
- صدور أمر بواسطة وحدة التحكم بالحاسب (control unit) للذهاب إلي الذاكرة في العنوان B لمعرفة محتواه وإرساله إلى وحدة العمليات الحسابية والمنطقية (ALU)

3) صدور أمر لوحدة العمليات الحسابية والمنطقية (ALU) لإجراء عملية الجمع لمحتوى العنوان A ومحتوى العنوان B

4) في النهاية صدور أمر آخر بواسطة وحدة التحكم لتخرين المجموع في العنوان A

الشكل رقم 2 يوضح محتوى ذاكرة الحاسب قبل وبعد تنفيذ العبارة: A = A + B

	قبل تنفيذ العبارة
A	$(7)_{10}=(0111)_2$
	.
	.
	.
B	$(10)_{10}=(1010)_2$

	بعد تنفيذ العبارة
A	$(17)_{10}=(10001)_2$
	.
	.
	.
B	$(10)_{10}=(1010)_2$

شكل 2: محتوى ذاكرة الحاسب قبل وبعد تنفيذ العبارة: A = A + B

d) قواعد اللغة:

هي مجموعة من القواعد والقيود التي يجب الالتزام بها عند كتابه البرامج وعند مخالفه هذه القواعد والقيود يعطي البرنامج أثناء مرحله الترجمة أخطاء لغوية (Syntax errors)

1.2) المصطلحات الأساسية للغات البرمجة

قبل البدء في معرفة المفاهيم الأساسية للبرمجة يجب أن نتناول معنى المصطلحات الرئيسية للغات البرمجة بالشرح والتفصيل

1.2.1)البيانات Data

هي مدخلات الحاسب وتعتبر هي المادة الخام للبرنامج والتي لا تعطي معني بذاتها

أنواع البيانات Data types

البيانات تستخدم أنواع مختلفة يجب أن تحدد بدقة قبل استخدامها داخل البرنامج. الأنواع الأساسية للبيانات هي:

1) بيانات عدديه (Numeric data):

هي الأرقام الصحيحة (integer) والحقيقية (real) والثنائية (binary) مثل:

$(123)_{10}$, $(12.56)_{10}$, and $(1011011)_2$

البيانات الثنائية تمثل القيمة المناظرة للبيانات داخل الذاكرة

2) بيانات حرفيه (Character data):

البيانات الحرفية character data تتكون من جميع الأرقام (9 ,... ,2 ,1 ,0) والحروف الهجائية الصغيرة lower-case letters والحروف الهجائية الكبيرة upper-case letters والحروف الخاصة (&, $, #,..etc) والتي تكون في مجملها 256 حرفا. البيانات الحرفية تمثل داخل برامج الحاسب بوضعها بين العلامتين (' ') فعلى سبيل المثال البيانات التالية تمثل character data :

'9', 'a', 'A', and '%'

3) كلمات (String data)

هي مجموعه من الحروف تشكل كلمات. String data تمثل داخل برامج الحاسب بوضعها بين العلامتين (" ") فعلى سبيل المثال البيانات التالية تمثل string data :

“abck”, "Ahmed”, and "1234"

4) بيانات منطقية Logical data(Boolean data) :

هي بيانات تأخذ احد القيمتين (False or True). بمكن استخدام T أو yes بدلا من القيمة true و F أو no بدلا من القيمة false. تستخدم البيانات المنطقية في التركيبات الشرطية

5) بيانات مركبة (Composite data):

تنقسم البيانات المركبة إلى نوعين رئيسيين هما المصفوفات (arrays) والسجلات /التركيبات (records/structures)

a) المصفوفات (array):

هي مجموعة من البيانات من نفس النوع

مثل : مصفوفة الأرقام الصحيحة {12, 15, 34} التي تحتوي على ثلاثة عناصر من البيانات الصحيحة و مصفوفة الأرقام الحقيقية {12.6, 12.4, 1.6} التي تحتوي على ثلاثة عناصر من البيانات الحقيقية

b) السجلات (records) أو التركيبات (structures) :

هي مجموعة من البيانات المختلفة في النوع مثل سجلات الموظفين حيث أن كل سجل يحتوي على اسم الموظف (كلمات string) ورقم الموظف (رقم صحيح integer) وراتبه الشهري (رقم حقيقي real). فعلى سبيل المثال

(“Ahemd”, 12450, ‘B’, 2020.5)

البيانات السابقة تكون سجل أو تركيب يحتوي على أربعة عناصر. العنصر الأول "Ahmed" (string data) يمثل اسم الموظف والعنصر الثاني 12450 (integer data) يمثل رقم الموظف والعنصر الثالث ‘B’ (character data) يمثل درجة الموظف الوظيفية والعنصر الرابع 2020.5 (real data) يمثل راتب الموظف الشهري

1.2.2)التعليمات Instructions

تنقسم التعليمات إلى تعليمات خاصة بالذاكرة و تعليمات خاصة بوحدات الإدخال والإخراج وتعليمات خاصة بالتحكم وهو ما تم تناوله سابقا

1.2.3)المعلومات Information

ناتج معالجة البيانات (Processed data) بواسطة إجراء التعليمات أو العمليات على البيانات تعطي قيم لها معني بذاتها تسمى معلومات . يمكن تعريف المعلومات أيضا بأنها هي مخرجات البرنامج التي تظهر في صورة تقارير أو جداول أو رسوم .

1.2.4)العوامل Operators

يجب إخبار الحاسب عن كيفية إجراء المعالجة للبيانات وذلك من خلال استخدام مجموعة من العوامل operators التي تعتبر data connectors في التعبيرات expressions والمعادلات equations . العوامل operators هي إشارات signs أو رموز symbols تخبر الحاسب عن كيفية معالجة البيانات وأيضا تخبر الحاسب عن نوع المعالجات (عمليات حسابية أو منطقية) المراد إجراؤها على البيانات وذلك بالإضافة إلى أنواع المتغيرات والثوابت التي يمكن استخدامها مع هذه العوامل .

العوامل Operators والمتغيرات variables والثوابت constants تجمع سويا لتكوين التعبيرات expressions أو المعادلات equations

أنواع العوامل المستخدمة في الحسابات وحل المسائل هي : معاملات حسابية ومعاملات منطقية ومعاملات علاقة وهو ما سنتناوله لاحقا.

1.2.5)المعاملات والمحصلات Operands and resultants

المعاملات operands هي البيانات التي توصل وتعالج بواسطة العوامل operators . المحصلات resultants هي الناتج أو المخرجات بعد إجراء المعالجات اللازمة على المعاملات فعلى سبيل المثال في التعبير 15 + 12 : + تمثل العامل operators والبيانات الرقمية 12 and 15 هما العوامل operands وناتج المعالجة 27 هو المحصلة resultant

المعاملات operands يمكن أن تكون متغيرات variables أو ثوابت constants.
نوع البيانات للمعاملات تعتمد على نوع العوامل operators . نوع المحصلة resultant تعتمد على نوع كل من العوامل والمعاملات

1.2.6)التعبيرات Expressions والمعادلات Equations

التعبيرات والمعادلات تمثل تعليمات instruction تستخدم للوصول إلى حل المشكلة

1.2.7)الدوال Functions

الدوال Functions هي مجموعة من التعليمات instructions تؤدي وظيفة أو عدة وظائف محددة

1.3) المتغيرات والثوابت VARIABLES AND CONSTANTS

الحاسبات تستخدم المتغيرات والثوابت في حل المسائل. المتغيرات والثوابت هما بيانات تستخدم في معالجات الحاسب. بصورة أوضح البيانات التي نتعامل بها من خلال البرامج تمثل قيمة متغير أو قيمة ثابت

1.3.1)المتغيرات Variables

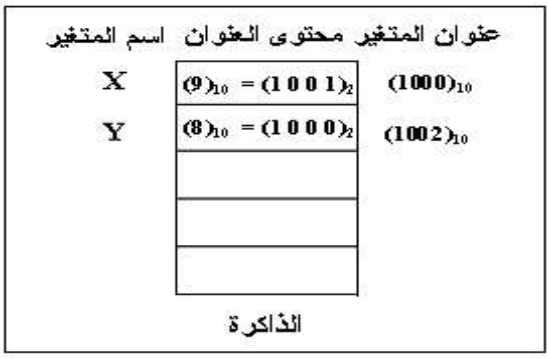
هي بيانات تتغير قيمتها أثناء تنفيذ البرنامج. يمكن تسمية هذه المتغيرات بحروف أو كلمات. من المهم جدا فهم الفارق بين اسم المتغير وقيمته. اسم المتغير يمثل علامة label يستخدمها الحاسب لإيجاد الوضع الصحيح للمتغير داخل ذاكرة الحاسب وقيمة المتغير تمثل محتوى هذا الموضع.

الحاسب يستخدم اسم المتغير لإيجاد موضعه داخل الذاكرة ويستخدم قيمة المتغير لإجراء المعالجات والعمليات. حيث أن اسم المتغير يمثل موضع تخزين المتغير فبالتالي يمكن القول أن اسم المتغير يمثل عنوان address تخزين قيمة البيانات التي تمثل محتوي هذا العنوان وحيث أن الموضع داخل الذاكرة معرض لتغير محتواه فان المتغيرات هي بيانات تتغير أثناء تنفيذ البرنامج.
فعلى سبيل المثال

$X = (9)_{10} = (1\ 0\ 0\ 1)_2$

$Y = (8)_{10} = (1\ 0\ 0\ 0)_2$

المتغير X يمثل بالعنوان **$(1000)_{10}$** في الذاكرة والذي يحتوي على القيمة $(9)_{10}$ في النظام العشري أو القيمة $(1001)_2$ في النظام الثنائي. المتغير Y يمثل بالعنوان **$(1002)_{10}$** في الذاكرة والذي يحتوي على القيمة $(8)_{10}$ في النظام العشري أو القيمة $(1000)_2$ في النظام الثنائي. الشكل رقم 3 يمثل هيكلا لجزء من ذاكرة الحاسب يتضمن محتوى وعنوان المتغيرين X and Y



شكل 3: هيكلا لجزء من ذاكرة الحاسب يتضمن محتوى وعنوان المتغيرين X and Y

الذي تمثل قيمة المتغير (Data)عند الإعلان عن المتغيرات يجب تحديد نوع البيانات

أمثلة:

variable X, Y: integer

الإعلان عن المتغيرين X, and Y من النوع integer

variable A1, B2: real

الإعلان عن المتغيرين A1, and B2 من النوع real

variable ch: character

الإعلان عن المتغير ch من النوع character

variable AM[20]: character

variable AM: string

العبارة الأولى تمثل الإعلان عن المتغير AM من النوع string (سلسلة حروف يحتوي على 20 حرف). يمكن استخدام العبارة الثانية في الإعلان عن المتغير AM من النوع string

variable BOL1: Boolean

الإعلان عن المتغير BOL1 من النوع Boolean

يمكن إعطاء قيم أولية للمتغيرات أثناء الإعلان عنها على النحو التالي

variable A = 20, B: integer

تهيئة المتغير A بالقيمة الأولية 20

variable ch ='K', ch1: character

تهيئة المتغير ch بالقيمة الأولية 'K' . لاحظ أننا وضعنا الحرف بين العلامتين ' ' للدلالة على وجود قيمة من النوع character

variable F1 = 12.5, F2: real

تهيئة المتغير F1 بالقيمة الأولية 12.5

variable M1=True: Boolean

تهيئة المتغير M1 بالقيمة الأولية True

variable X1[5] ={12, 34, 5, 13, 42}: integer

تهيئة المتغير X1 بالقيمة الأولية 12, 34, 5, 13, and 42. لاحظ أننا وضعنا الأرقام الصحيحة بين العلامتين { } للدلالة على وجود مصفوفة عناصرها من النوع integer ولاحظ أيضا أننا وضعنا الرقم 5 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X2[3] = { 12.6, 10.4, 6.0, 4.2, 19.3} : real

تهيئة المتغير X2 بالقيمة الأولية 12.6, 10.4, 6.0, 4.2, and 19.3. نلاحظ أننا وضعنا الأرقام العشرية بين العلامتين { } للدلالة على وجود مصفوفة عناصرها من النوع real ولاحظ أيضا أننا وضعنا الرقم 3 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X3[8] = "Computer" : character

تهيئة المتغير X3 بالقيمة الأولية Computer. نلاحظ أننا وضعنا الأحرف بين العلامتين " " للدلالة على وجود مصفوفة من الحروف و لاحظ أيضا أننا وضعنا الرقم 8 بين القوسين [ ] لتحديد عدد عناصر المصفوفة

variable X4[8] = {'C', 'o', 'm','p','u','t','e','r'} : character

يمكن عمل تهيئة بصورة أخرى لمصفوفة الحروف وذلك بوضع الأحرف بين العلامتين { } للدلالة على وجود مصفوفة ووضع كل حرف بين العلامتين ' '

يمكن إسناد قيمة إلى المتغير خارج عبارة الإعلان عن المتغير وذلك من خلال إدراج عبارات الإسناد التالية داخل جسم البرنامج

A = 20

ch ='K'

F1 = 12.5

M1=True

1.3.2)الثوابت Constants

هي بيانات لا تتغير قيمتها أثناء تنفيذ البرنامج

أمثلة:

constant A3=20: integer

A3 مقدار ثابت من النوع integer قيمته 20

constant AA = 12.4 real

AA مقدار ثابت من النوع real قيمته 12.5

constant name = "Ali": string,

Or

constant name[5] = "Ali": character

name مقدار ثابت من النوع string قيمته "Ali"

عند وجود الثابت داخل جسم البرنامج يقوم المترجم (compiler) بالتعويض عن قيمة الثابت بالقيمة المحددة عند الإعلان عنه. فعلى سبيل المثال في العبارة

Y = A3 + 10

يقوم المترجم بالتعويض عن الثابت A3 بالقيمة 20 المعرفة مسبقا في العبارة

: integer constant A3 = 20

وبالتالي تكون قيمة المتغير Y تساوي 30

1.4) عمليات البرمجة Software Operations

يوجد ثلاث عمليات رئيسية يتم استخدامها في البرمجة وهي:

a) عمليات حسابية Arithmetic operation
b) عمليات منطقية Logical operations
c) عمليات العلاقة Relational operations

1.4.1) العمليات الحسابية Arithmetic operations

الجدول رقم 1 يوضح العمليات الحسابية وعامل (operator) كل عملية

جدول 1: العمليات الحسابية وعامل (operator) كل عملية :

العملية	operatorالعامل
الجمع	+
الضرب	*
الطرح	-
القسمة	/
باقي القسمة	%MOD أو
الأس	** أو ^

هناك عوامل (operators) حسابية أخري تسمى عوامل التوظيف (assignment operators) تختص باللغة المستخدمة وخصائصها . الجدول رقم 2 يوضح عوامل التوظيف (assignment operators) في لغة C والمعنى المكافئ لكل عامل. معاملات العمليات الحسابية والمحصلة (operands and resultants) يكون من النوع numeric data (أرقام صحيحة integer numbers أو أرقام حقيقية real numbers) . فعلى سبيل المثال إذا كان: A = 30 and B = 4 فان :

C1 = A + B ⟶ C1 = 34

C2 = A – B ⟶ C2 = 26

C3 = A / B ⟶ C3 = 7.5

C4 = A * B ⟶ C4 = 120

C5 = A Mod B ⟶ C5 = 2

حيث أن A, and B هي أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات صحيحة (integer numbers) بينما C1, C2, C3, C4, and C5 هي أسماء متغيرات تمثل محصلات (resultants) تحتوي على بيانات صحيحة أو عشرية حسب نوع العامل المستخدم في العملية

جدول 2: عوامل التوظيف (assignment operators)

في لغة C والمعنى المكافئ لكل عامل

عوامل التوظيف والمعاملات	المعنى المكافئ
X++	استخدم القيمة الموجودة أولا للمتغير X ثم أضف 1 إلى هذه القيمة بعد ذلك
X—	استخدم القيمة الموجودة أولا للمتغير X ثم اطرح 1 من هذه القيمة بعد ذلك
++X	زد قيمة المتغير X بمقدار 1 أولا ثم استخدم القيمة الجديدة بعد ذلك
--X	اطرح 1 من قيمة المتغير X ثم استخدم القيمة الجديدة بعد ذلك
-X	اعكس إشارة المتغير X
X=Y	قيمة المتغيرX تساوي قيمة المتغير Y
X*=Y	تكافئ العبارة X=X*Y
X/=Y	تكافئ العبارة X=X/Y

X%=Y	تكافئ العبارةX%Y= X
X+=Y	تكافئ العبارة X=X+Y
X-=Y	تكافئ العبارةX-Y= X

1.4.2)عمليات العلاقة Relational operations

الجدول رقم 3 يوضح عمليات العلاقة وعامل (operator) كل عملية

المعاملات (operands) تكون بيانات عددية: صحيحة (integer) أو حقيقية (real).

ناتج عمليات العلاقة يكون بيانات من النوع logical (Boolean) data (False or True) .

فعلى سبيل المثال إذا كان A = 10 and B = 5 فان :

C1 = A > B ⟶ C1 = True (T)

C2 = A < B ⟶ C2 = False (F)

C3 = A ==B ⟶ C3 = False (F)

حيث أن A, and B هما أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات صحيحة. C1, C2, and C3 هي أسماء متغيرات تمثل محصلات (resultants) تحتوي على بيانات Boolean تأخذ قيم False (F) or True (T). إذا تحقق الشرط يكون ناتج العلاقة يساوي True (T) وإذا لم يتحقق الشرط يكون ناتج العلاقة False (F). تستخدم عمليات العلاقة في التركيبات الشرطية

جدول 3: عمليات العلاقة وعامل (operator) كل عملية

العملية	operator العامل
يساوي	==
لا يساوي	!=

اقل من	>
اقل من أو يساوي	=<
اكبر من	<
اكبر من أو يساوي	<=

1.4.3)العمليات المنطقية Logical operations

الجدول رقم 4 يوضح العمليات المنطقية وعامل (operator) كل عملية. الجدول رقم 5 يوضح ناتج العمليات المنطقية. العوامل المنطقية logical operators تستخدم لربط تعبيرات العلاقة relational expressions مثل :

A > B && A < 9

بالإضافة إلى إجراء العمليات operations على البيانات المنطقية logical (Boolean) data

جدول 4: العمليات المنطقية وعامل (operator) كل عملية

العملية	العامل operator
NOT	**!**
AND	**&&**
OR	**||**

جدول 5 : ناتج العمليات المنطقية

A	B	A&&B	A||B	!A	!B
F	F	F	F	T	T
F	T	F	T	T	F
T	F	F	T	F	T
T	T	T	T	F	F

المعاملات (operands) تكون بيانات منطقية logical (Boolean) data (T or F). ناتج العملية المنطقية يكون من النوع logical (Boolean) data (F or T). فعلى سبيل المثال إذا كان A = T, and B = F فان:

C1 = A && B ⟶ C1 = F

C2 = A || B ⟶ C2 = T

C3 = !A ⟶ C3 = F

حيث أن A, B, C1, C2, and C3 هي أسماء متغيرات تمثل معاملات (operands) تحتوي على بيانات منطقية (T or F)

1.5) قواعد عامة متعلقة بأنواع البيانات

GENERAL RULES FOR DATA TYPES

1) تعريف البيانات قبل استخدامها:

قبل استخدام أي بيانات في البرنامج يجب تعريفها. يتم تعريف البيانات من ناحيتين

a) ناحية الغرض: Data object

- Variable

- Constant

b) ناحية النوع Data type

- integer
- real
- character
- string
- Boolean

2) في معظم لغات البرمجة لا يصح إجراء عمليات علي بيانات من أنواع مختلفة (different data types) فعلى سبيل المثال إذا تم الإعلان عن المتغيرات Y, F, X and K بالعبارات التالية

variable Y,F: integer

variable X: character

variable K: real

واستخدمت العبارة التالية

Y = X + F

فان هذا العبارة تكون (في معظم اللغات) خاطئة حيث أن المتغيرات من أنواع مختلفة (المتغير X من النوع character والمتغير F من النوع integer) .

بعض لغات البرمجة يمكن أن تعتبر هذه العبارة صحيحة على أساس أنه يتم تحويل البيانات الصحيحة إلى حقيقية أو العكس أو تحويل الحرف إلى قيمة ASCII code الخاص به تلقائيا

3) يجب أن تتناسب نوع العملية مع المعاملات (operands). فعلى سبيل المثال إذا تم الإعلان عن المتغيرات X1, X2, x3, A1, A2,and A3 بالعبارات التالية

variable X1, X2, X3: BOOL

constant A1, A2, A3: integer

واستخدمت العبارتين التاليتين:

X1 = X2 + X3

A1 = A2 && A3

فان العبارة الأولى تكون خاطئة حيث أنه لا يمكن إجراء العمليات الحسابية (الجمع) على متغيرات من النوع Boolean والعبارة الثانية تكون خاطئة أيضا حيث أنه لا يمكن إجراء العمليات المنطقية (&&)على متغيرات من النوع integer

4) لا يمكن خلط البيانات فعلى سبيل المثال لا يمكن وضع string data في موضع متغير داخل الذاكرة يكون معرفا على أن محتواه numeric data
5) كل نوع من البيانات يستخدم قائمة محددة من البيانات data set فعلى سبيل المثال integer data تستخدم جميع الأرقام الموجبة والسالبة و real data تستخدم الأرقام العشرية و character data تستخدم جميع الحروف والرموز الخاصة مع وضع الحرف بين العلامتين (' ') و Boolean data تستخدم أحد القيمتين true أو false . إذا تم استخدام بيانات خارج هذه القائمة يكون هناك خطأ

1.6) التعبيرات والمعادلات Expressions and Equations

التعبيرات expressions تعالج البيانات (operands) من خلال استخدام عوامل operators محددة. فعلى سبيل المثال إذا كان X معامل يمثل طول مستطيل والمعامل Y يمثل عرض المستطيل فان التعبير :

X * Y

يمثل مساحة المستطيل. ناتج التعبير لا يخزن في الذاكرة

المعادلات equations تخزن ناتج التعبير في موضع محدد داخل الذاكرة من خلال استخدام عامل الإسناد assignment operator(=) و بالتالي فان المعادلة تتكون من تعبير بالإضافة

إلى عامل الإسناد. تعليمة الحاسب computer instruction تمثل بواسطة معادلة . فعلى سبيل المثال:

Z = X * Y

تمثل معادلة equation أو تعليمة instruction حيث يتم تخزين ناتج التعبير (X * Y) في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغير Z

الجدول رقم 6 يوضح الفارق بين التعبيرات والمعادلات

جدول 6 : الفارق بين التعبيرات والمعادلات

المعادلةEquation	التعبيرExpression
D = A * C operands معاملات A and C تمثل بيانات من النوع numerical data وتخزن المحصلة resultant في موضع (numerical data) داخل الذاكرة محدد بالعنوان الذي يمثله المتغيرD	**A * C** operands معاملات A and C تمثل بيانات من النوع numerical data ولا تخزن المحصلة resultant في الذاكرة
D = A > B operands معاملات A and B تمثل بيانات من النوع , numerical string or character data وتخزن المحصلة resultant في موضع داخل الذاكرة محدد بالعنوان الذي يمثله (logical data)	A > B operands معاملات A and B تمثل بيانات من النوع , numerical ولا string or character data تخزن المحصلة resultant في الذاكرة

المتغير D	
A && B معاملات A and B operands تمثل بيانات من النوع logical data وتخزن المحصلة resultant في موضع داخل الذاكرة محدد بالعنوان الذي يمثله المتغير D	A && B معاملات A and B operands تمثل بيانات من النوع logical data ولا تخزن المحصلة resultant في الذاكرة

المعادلة equation تسمى أحيانا عبارة التوظيف assignment statement وذلك لأن قيمة التعبير على يسار عامل التوظيف يتم إسناده إلى المتغير على يسار عامل التوظيف

1.7) الأسبقيات PERIORITIES

في الجزء التالي سنناقش أسبقيات تنفيذ العمليات الحسابية والمنطقية والمختلطة

1.7.1) أسبقيات العمليات الحسابية Priorities of Arithmetic Operators

الجدول رقم 7 يوضح ترتيب أسبقيات العمليات الحسابية ونوع بيانات كل من operands and resultants

جدول 7 : ترتيب أسبقيات العمليات الحسابية

ونوع بيانات كل من operands and resultants

الترتيب	نوع بيانات operands	نوع بيانات resultant
1) الأقواس 2) الأس	Numerical	Numerical

3) باقي القسمة	Numerical	Numerical
4) الضرب والقسمة (من اليسار إلي اليمين)	Numerical	Numerical
5) الجمع والطرح (من اليسار إلي اليمين)	Numerical	Numerical
	Numerical	Numerical

مثال: 2

أحسب ناتج المعادلات التالية مع توضيح خطوات التنفيذ

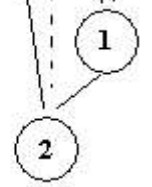
a) X = 3 + 2 * 4

b) Y = 2 * 6 + 3 – 2 * 4

c) Z = (2 * 5 ** 2) / 2 + 12

الحل:

a) X = 3 + 2 * 4

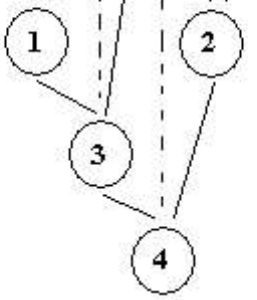


X = 3 + 2 * 4

X = 3 + 8

X = 11

b) Y = 2 * 6 + 3 – 2 * 4



Y = 2 * 6 + 3 – 2 * 4

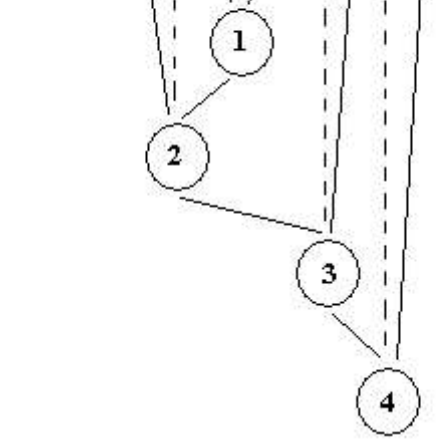
Y = 12 + 3 – 2 * 4

Y = 12 + 3 - 8

Y = 15 – 8

Y = 7

c) Z = (2 * 5 ** 2) / 2 + 12



Z = (2 * 5 ** 2) / 2 + 12

Z = (2 * 25) / 2 + 12

Z = 50 / 2 + 12

Z = 25 + 12

Z = 37

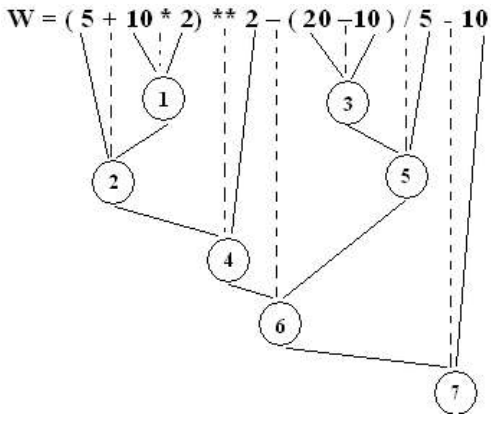
مثال: 3

إذا كان X = 5, Y = 10, Z = 20, أحسب ناتج المعادلة التالية مع توضيح خطوات التنفيذ

W = (X+Y *2) **2 – (Z- Y) /X – 10

الحل:

W = (5 + 10 * 2) ** 2 – (20 –10) / 5 – 10



W = (5 + 10 * 2) ** 2 – (20 –10) / 5 – 10

= (5 +20) **2 – (20 -10) / 5 -10

= 25 ** 2 – (20 – 10) / 5 – 10

= 25 ** 2 – 10 / 5 – 10

= 625 – 10 / 5 – 10

= 625 – 2 – 10

= 623 – 10

= 613

1.7.2)أسبقيات عمليات العلاقة Priorities of Relational Operations

الجدول رقم 8 يوضح ترتيب أسبقيات عمليات العلاقة ونوع بيانات كل من operands and resultants

جدول 8 : ترتيب أسبقيات عمليات العلاقة

ونوع بيانات كل من operands and resultants

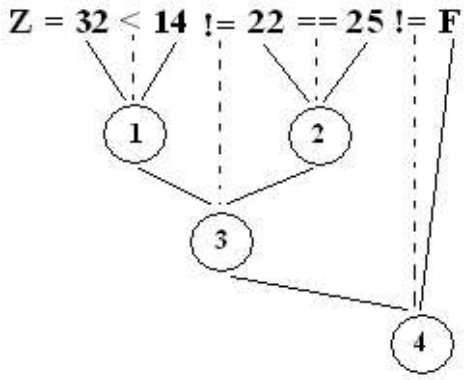
نوع بيانات resultant	نوع بيانات operands	الترتيب
logical	Numerical, strings or characters	==, <, <=, >, >=, <> (!=) من اليسار إلى اليمين

مثال: 4

احسب ناتج المعادلة التالية

Z = 32 < 14 != 22 == 25 != F

الحل



Z = 32 < 14 != 22 == 25 != F

= F != 22 == 25 != F

= F != F !F

= F != F

= F

1.7.3)أسبقيات العمليات المنطقية Priorities Of Logical Operations

الجدول رقم 9 يوضح ترتيب أسبقيات العمليات المنطقية ونوع بيانات كل من operands and resultants

جدول 9 : ترتيب أسبقيات العمليات المنطقية

ونوع بيانات كل من operands and resultants

الترتيب	نوع بيانات operands	نوع بيانات resultant
1) الأقواس	Logical	Logical
2) NOT والتي تكافئ !	Logical	Logical
3) AND والتي تكافئ &&	Logical	Logical
4) OR والتي تكافئ ||	Logical	Logical

مثال: 5

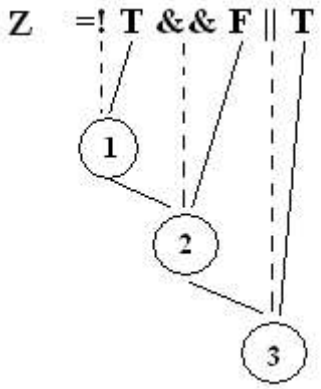
إذا كان A = T , B = F أحسب ناتج المعادلات التالية مع توضيح خطوات التنفيذ

a) Z = !A && B || A

b) Z = ! (A || B) && B

الحل:

a) Z = !A && B || A



Z = !A && B || A

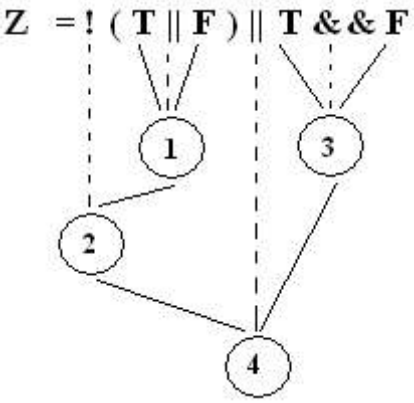
=! T && F || T

= F && F || F

= F || T =

= T

b) Z = ! (A || B) || T && B



Z = ! (T || F) || T && F

= ! T || T && F

= F || T && F

= F || F

= F

تمارين (1)

1) أوجد الصيغة المستخدمة في الحاسب لتمثيل التعبير التالي:

$$A(C+5B)+\frac{A+BC}{5B+12C}*(2X+B)$$

2) إذا كان A = 3 , B = 7 , C=12 and D=2 . أحسب قيمة التعبيرات التالية مع توضيح خطوات الحل

a) A > 3 || B < = 5

b) !(A < 5) && (A > D)

c) (A+B) < D || !(C < D)

d) A < B + 5 && C + 5 < D

e) A 2/5+2 >= 3 || 5 < D**

f) A +B / 2 * 2 < D && !(D == A)

3) إذا كان A = 3 , B = T , C = 12 and D = 2 أحسب قيمة المعادلات التالية مع توضيح خطوات الحل

F = B && A < C || A < D

F = C > D && B || A != D

4) إذا كان a = 4, b = 9, c = 8, and d = -1 احسب نتيجة التعبيرات التالية مع توضيح خطوات الحل

a) a – c / d * 2 + d*d – b MOD 3

b) (d * 2 - (a – c / 2) / (b * b – d + a)) / a + b * 2

5) إذا كان a = 5, b = 3, c = T, and d = F احسب نتيجة التعبيرات التالية مع توضيح خطوات الحل

c) a < b || !(c && b > a) && d

d) (a * 2 > 1) && d || !(a - b <2 && !c)

6) إذا كان A = T , B = F أوجد قيمة Z في المعادلات التالية مع توضيح خطوات الحل

d) Z = ! A || B && A

e) Z = ! (A && B) || B

f) Z = A < B

g) Z = (3 > 2) && A || (2 < 6)

7) إذا كان2= C = 7, and D . أوجد قيمة Y من المعادلة التالية مع توضيح خطوات الحل

Y = (C **D < 8) && (A && D == 4)

8) إذا كان A=1, B=2, and C=5. أوجد ناتج التعبير التالي

(A<C) || (C > B && C+3 > 9)

الفصل الثاني
الخوارزميات : المفهوم والخصائص وطرق الصياغة

ALGORITHMS: CONCEPTS, CHARACTERSITICS AND FORMATION METHODS

في هذا الفصل سنتعرف على المراحل الأربعة الأساسية لحل المسائل (تعريف المسألة وتحليلها والبرمجة والتوثيق) ومفهوم الخوارزميات وخصائصها وطرق صياغتها باستخدام اللغات الطبيعية والمخططات الانسيابية وكود الشفرة بالإضافة إلى شرح كامل للمخططات الانسيابية من خلال تحديد الأشكال والعناصر المستخدمة ومدلول كل منها واستخدامها في صياغة الخوارزميات وشرح كامل لاستخدام كود الشفرة في صياغة الخوارزميات

2.1) مقدمة INTRODUCTION

في هذا الفصل سنتناول المراحل الأساسية المتبعة لحل المسائل بدء من تعريف المسألة وتحليلها حتى مرحلة التوثيق مرورا بمرحلة البرمجة وذلك بالإضافة إلى تناول مفهوم الخوارزميات وخصائصها وطرق صياغتها وسنقوم بالتركيز على طريقتين شائعي الاستخدام في صياغة الخوارزميات وهما طريقة المخططات الانسيابية (flowchart) وطريقة كود الشفرة (pseudo code).

2.2) مراحل حل المسائل STEPS FOR PROBLEM SOLVING

لحل المسائل يوجد مراحل متعددة يجب تنفيذها بغية تحقيق الحل الأمثل للمسألة من الناحية الشكلية والتنفيذية ومراحل حل المسائل هي:

2.1.1)تعريف المسألة Problem Definition

يعني بتعريف المسألة دراستها وفهمها فهما جيدا وتحديد المعطيات (المدخلات) وتحديد المخرجات أو النتائج

2.1.2)تحليل المسألة Problem Analysis

وهى مرحلة تحديد العمليات والخطوات التي تؤدي إلى حل المسألة وتعتبر هذه المرحلة هي مرحلة كتابة الخوارزميات سواء كان باستخدام المخططات الانسيابية (flowchart) أو باستخدام كود الشفرة (Pseudo code)

2.1.3)البرمجة Programming

تحويل flowchart أو Pseudo code إلى برنامج لحل المسألة باستخدام إحدى لغات البرمجة . وتنقسم هذه المرحلة إلى 4 مراحل فرعية

a) Source editing

يتم كتابة البرنامج من خلال لوحة المفاتيح (keyboard) بإحدى لغات البرمجة ولتكن على سبيل المثال لغة python ثم يتم تخزين هذا البرنامج داخل ذاكرة الحاسب باسم وليكن (test.py) مع العلم أن الامتداد (.py) المرفق مع الاسم يدل على أن البرنامج تم كتابته بلغة Python . بانتهاء هذه المرحلة الفرعية يكون لدينا برنامج يسمى source program

b) الترجمة Compilation

في هذه المرحلة الفرعية يتم إجراء الوظائف التالية :

1) الكشف عن الأخطاء اللغوية (syntax error) وتعديلها حسب قواعد اللغة المستخدمة حتى يكون البرنامج خالي من الأخطاء اللغوية

2)تحويل source program إلى برنامج لغة الآلة يسمى object file بواسطة المترجم (compiler) المستخدم تبعا لنوع اللغة المستخدمة

بانتهاء هذه المرحلة الفرعية يكون لدينا ملف يسمى object file وليكن test.obj

c) الربط Linking

يتم بواسطة نظام التشغيل (operating system) ربط object file (test.obj) أو object files أخرى وذلك للحصول ملف التنفيذ (test.exe).

في نهاية هذه المرحلة الفرعية يكون لدينا ملف التنفيذ execution file والذي يتم استخدامه بواسطة المستخدم (user) لتشغيل البرنامج

d) المحاكاة Simulation

تستخدم عملية المحاكاة للتأكد من خلو البرنامج من الأخطاء المنطقية (logical errors) وذلك من خلال إدخال دخل (input) معين معروف مقدما نتائج البرنامج له (predefined output) فإذا كانت مخرجات البرنامج (results) تساوي النتائج المعرفة مقدما (predefined output = result) كان البرنامج خاليا من الأخطاء المنطقية وإلا سيقوم المبرمج بإعادة التحقق من عبارات البرنامج مرة أخرى حتى يتأكد من خلو البرنامج من جميع الأخطاء المنطقية

2.1.4)التوثيق Documentation

يعني بالتوثيق كتابة تقرير منفصل (أو تعليقات على البرنامج الأصلي (source program لأهم المعلومات عن البرنامج وطبيعة عمله بهدف الرجوع إليه عند الحاجة وللمساعدة في التطوير المستقبلي.

ونظرا لأن تحليل المسألة (problem analysis) هو الخطوة الأساسية لتحديد طريقة الحل طبقا لسرعة وسهولة ودقة الوصول إلى النتائج لذا سنتحدث في هذا الفصل عن الطرق المختلفة لتحليل المسائل

مثال: 1

أوجد قيمة المتغير Z الناتج من المعادلة :

$$Z = (X - Y)^2$$

خطوات الحل:

1) **فهم المسألة:** وهو حساب قيمة المتغير Z المعطى بالمعادلة السابقة وتحديد المدخلات وهي X and Y وتحديد المخرجات وهو إيجاد قيمة المتغير Z المحدد بالمعادلة السابقة

2) **مرحلة التحليل:** استعراض الطرق المختلفة للحل واختبار أنسبها من ناحية السرعة والسهولة وكذلك الدقة

يوجد طريقتان للحل:

الطريقة الأولى:

1) التعويض بقيمة كل من المتغيرين X and Y
2) إيجاد ناتج (X – Y)
3) إيجاد مربع الناتج السابق للحصول على قيمة المتغير Z

الطريقة الثانية:

حساب قيمة المتغير Z من خلال المعادلة

$$Z = X^2 + Y^2 - 2 * X * Y$$

1) التعويض بقيمة كل من المتغيرين X and Y
2) إيجاد مربع المتغير X^2 = X
3) إيجاد مربع المتغير Y^2 = Y
4) إيجاد قيمة 2*X*Y
5) إيجاد مجموع مربعي المتغيرين $X^2 + Y^2$= X and Y
6) إيجاد قيمة المتغير Z من خلال طرح ناتج الخطوة 4 من ناتج الخطوة الخامسة

بتحليل الطريقتين السابقتين يتضح أن الطريقة الأولى أسرع وأسهل في الوصول إلى الحل

2.3) مفهوم الخوارزميات ALGORITHMS CONCEPTS

جاءت كلمة خوارزم من اسم العالم العربي محمد بن موسى الخوارزمي وهو من علماء الرياضة العرب في القرن الثامن عشر الميلادي

2.3.1)تعريف الخوارزم Definition

هي مجموعة من الخطوات المنطقية التي يتم تنفيذها حسب ترتيب محدد يتصف بالدقة والوضوح والشمولية للوصول إلى الحل الأمثل للمسألة

2.3.2)خصائص الخوارزميات Characteristics of Algorithms

للخوارزميات خصائص عدة منها:

1) الخطوات الخوارزمية مرتبة ترتيبا منطقيا (well-ordered steps)
2) الخطوات الخوارزمية محددة (defined steps) ومنتهية (ended)
3) الخطوات الخوارزمية تنفذ عمليات بسيطة (simple operations)
4) يعرف الخوارزم تعريفا جيدا وذلك من خلال تحديد واضح لبيانات الدخل والعمليات والتعليمات والأوامر
5) طريقة عامة للحل universal solution بحيث يمكن تطبيقها لحل مسائل أخرى من نفس النوع
6) الوصول إلى الحل بطريقة مباشرة بدون تعقيد أو إطالة

2.3.3) طرق صياغة الخوارزميات Algorithms Formation Methods

يوجد طرق عديدة لصياغة الخوارزميات تختلف فيما بينها من حيث الدقة وسهولة الفهم والسرعة في الوصول إلى الحل

a) استخدام اللغات الطبيعية Using Natural Languages algorithms

هي الطريقة المباشرة للتعبير عن الخوارزم وذلك بتوضيح خطوات الحل بواسطة جمل وعبارات من خلال استخدام اللغات الطبيعية كالعربية والإنجليزية والفرنسية

مثال: 2

أوجد الخوارزمية التي تقوم بتحويل درجة الحرارة السنتغراد إلى درجة فهرنهيتية باستخدام المعادلة

$$F = 32 + 9 * \frac{C}{5}$$

الخوارزم:

1) السؤال عن إدخال درجة الحرارة السنتغراد برمز C
2) تخزين درجة الحرارة السنتغراد (C) في الذاكرة
3) حساب درجة الحرارة الفهرنهيتية المناظرة من خلال المعادلة السابقة

4)تخزين درجة الحرارة الفهرنهيتية (F) في الذاكرة

5) طباعة قيمة كل من F and C

خوارزم إعداد الرسالة البريدية و خوارزم سير النظام اليومي هي أمثلة لاستخدام اللغات الطبيعية في صياغة الخوارزميات

غالبا لا تستخدم اللغات الطبيعية في صياغة الخوارزميات لطولها وأحيانا لعدم الدقة ولذلك يتم استخدام طرق أخرى أشهرها المخططات الانسيابية (Flowcharts) أو كود الشفرة

b) استخدام المخططات الانسيابية Using Flowcharts

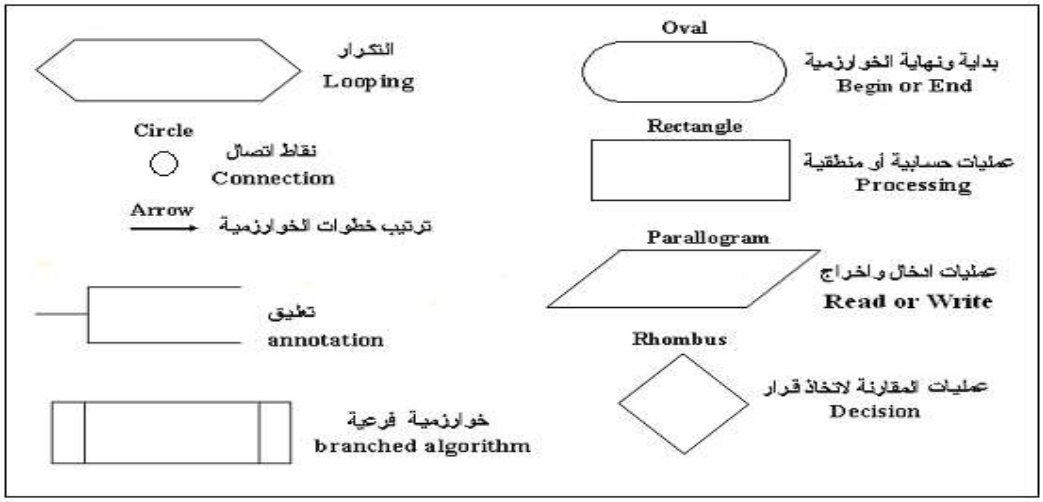
هي استخدام مجموعة من الأشكال أو العناصر الهندسية المتصلة مع بعضها بواسطة استخدام أسهم ونقاط اتصال

c) استخدام كود الشفرة Using Pseudo code

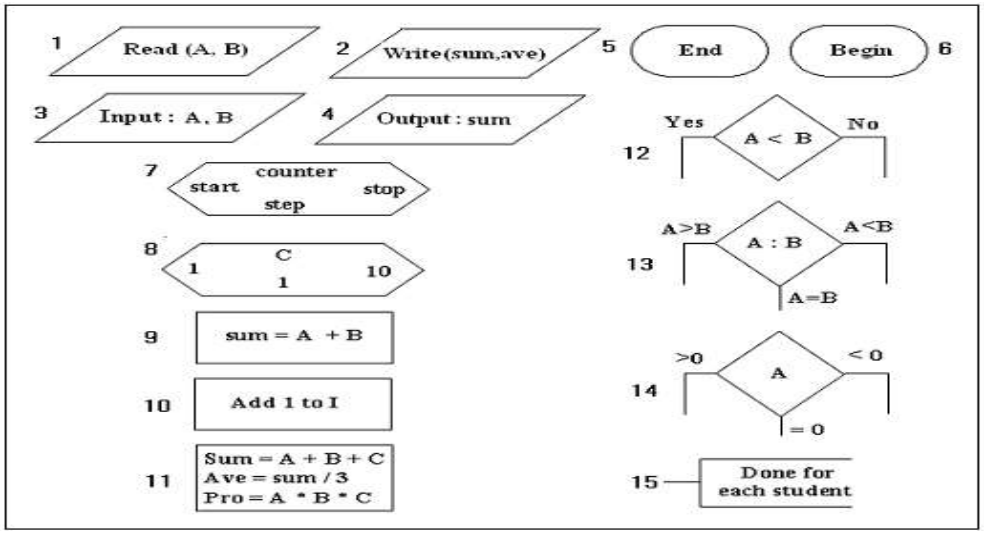
هي لغة رمزية ليس لها مترجم و تستخدم بعض العبارات القريبة من اللغات الطبيعية

2.4) المخططات الانسيابية FLOWCHARTS

هي استخدام مجموعة من العناصر أو الأشكال الهندسية المتصلة مع بعضها بواسطة استخدام أسهم ونقاط اتصال. الشكل رقم 2 يوضح العناصر المستخدمة في تكوين المخططات الانسيابية ومدلول كل عنصر. الشكل رقم 3 يوضح محتوى كل عنصر من عناصر المخططات الانسيابية



شكل 2: العناصر المستخدمة في تكوين المخططات الانسيابية ومدلول كل عنصر



شكل 3: محتوى كل عنصر من أشكال المخططات الانسيابية

ملحوظة

في الشكل رقم 3 العنصر رقم 7 والعنصر رقم 8 يمثلان العدادات أو التكرار المحدد وعند استخدام هذه العناصر يجب تحديد التالي

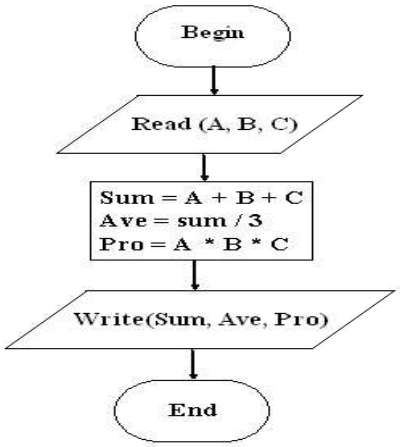
1) اسم العداد: counter = C
2) بداية العداد: start = 1
3) نهاية العداد: stop = 10
4) قيمة القفز: step = 1

العنصر رقم 8 يوضح أن العداد C يأخذ القيم (1, 2, 3, 4, 5, 6, 7, 8, 9, and 10) على التوالي

مثال 3:

أوجد المخطط الانسيابي لخوارزم يقرأ قيم المتغيرات A, B, and C ويقوم بحساب وطباعة المجموع (Sum) والمتوسط الحسابي (Ave) وحاصل الضرب (Pro)

الحل:



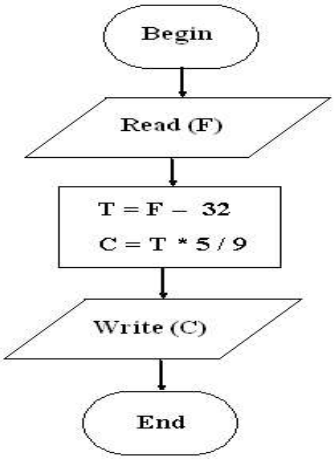
المخطط الانسيابي (Flowchart) للمثال رقم 3

مثال: 4

أوجد المخطط الانسيابي لخوارزم يقوم بتحويل درجة الحرارة من فهر نهيت (F) إلى سنتغراد (C). الخوارزم يقرأ درجة الحرارة الفهرنهيتية ويقوم بطباعة درجة الحرارة بالسنتغراد. قاعدة التحويل من درجة الحرارة الفهرنهيتية إلى درجة الحرارة السنتغراد تعطى بالمعادلة التالية:

$$C = (F - 32) * \frac{5}{9}$$

الحل:



المخطط الانسيابي (Flowchart) للمثال رقم 4

2.5) كود الشفرة PSEUDO CODE

هي لغة رمزية ليس لها compiler ولا تشكل source program وتستخدم بعض العبارات القريبة من اللغات الطبيعية.

2.5.1) خطوات إنشاء Pseudo code

1) الإعلان عن المتغيرات (variables declaration)

للإعلان عن المتغيرات يجب تحديد التالي

a) اسم المتغير والذي يحدد بواسطة المستخدم
b) نوع المتغير: (Integer, Real, Character, String, Boolean)
c) تهيئة المتغيرات بقيم أولية إن وجدت (Optional)

في بعض لغات البرمجة يكون الهدف من الإعلان عن المتغير هو إبلاغ المترجم compiler عن نوع المتغير (Variable data type) ليقوم بتحديد المساحة التخزينية المطلوبة له داخل الذاكرة والتي يمكن أن تحدد كالتالي :

a) Integer = 2 bytes
b) Real = 4 bytes
c) Character = 1 bytes

المتغير من النوع Integer يشغل 2 bytes من الذاكرة. المتغير من النوع Real يشغل 4 bytes من الذاكرة. المتغير من النوع Character يشغل 1 bytes من الذاكرة

2) كتابة كلمة Begin لبداية الخوارزم

يجب كتابة كلمة Begin لبداية الخوارزم

3) كتابة جسم الخوارزم: والذي يحتوي على التالي:

a) عبارة قراءة دخل الخوارزم Read statement
b) العبارات التنفيذية (التعليمات والعمليات والأوامر)
c) عبارة طباعة نتائج الخوارزم Write statement

4) كتابة كلمة End لإنهاء الخوارزم

يجب إنهاء الخوارزم بكتابة كلمة End في نهايته

مثال : 5

أوجد Pseudo code للمثال رقم 3

الحل:

```
Variable A, B, C, Sum, Pro: integer
Variable Ave: real
Begin
        Read (A, B, C)
        Sum = A + B + C
        Ave = Sum / 3
        Pro = A * B * C
        Write (Sum, Ave, Pro)
End
```

مثال : 6اوجد Pseudo code للمثال رقم 4

```
Variable C, F, T: real
Begin
        Read (F)
        T = F – 32
        C = T * 5 / 9
        Write (F,  C)
End
```

التمارين (2)

1) أوجد المخطط الانسيابي (Flowchart) بالإضافة إلى كود الشفرة (Pseudo code) لخوارزم يقوم بقراءة قيمة المتغير X ثم حساب وطباعة قيمة المتغير Y بواسطة المعادلة التالية:

$$Y = \sqrt{X + 5}$$

2) أوجد المخطط الانسيابي (Flowchart) بالإضافة إلى كود الشفرة (Pseudo code) لخوارزم يقوم بقراءة قيم المتغيرين A and B ثم حساب وطباعة قيم المتغيرات Y, Z, and W بواسطة المعادلات التالية

$$Y = 3 * A - 4 * B$$
$$Z = 2 * Y + 5$$
$$W = Z \; MOD \; 2$$

3) أوجد المخطط الانسيابي (Flowchart) بالإضافة إلى كود الشفرة (Pseudo code) لخوارزم يقوم بقراءة قيم المتغيرين A and B ثم حساب وطباعة قيم المتغيرات Z بواسطة المعادلة التالية

$$Z = (X - Y)^2$$

4) أوجد المخطط الانسيابي (Flowchart) بالإضافة إلى كود الشفرة (Pseudo code) لخوارزم يقوم بقراءة قيم المتغيرين A and B ثم حساب وطباعة قيم المتغيرات X, Y, and Z بواسطة المعادلات التالية :

$$X = 2 * A / (B * *2 + 2)$$
$$Y = X + 2 * B$$
$$Z = (X * Y) / 2$$

الفصل الثالث
تمثيل البيانات في الحاسب

مقدمة :

يعد استخدام الأرقام كوسيلة للعد والحساب من الإنجازات الهامة التي حققها الإنسان عبر التاريخ والتي ساهمت في تسهيل كافة العمليات الحسابية وتسريعها. فقد إستخدم الإنسان منذ القدم الكثير من الأدوات لتمثيل عمليات العد والحساب ومنها استخدامه لأصابع يده العشرة والتي كانت الأساس للنظام العددي والذي لا يزال معمول به حتى يومنا هذا والمسمى **بالنظام العشري** (Decimal System).

في المراحل الدراسية السابقة وعند دراستك للنظام العشري لابد أنك لاحظت أن القيمة الحقيقية للرقم تعتمد على قيمته المكانية في العدد , وهذا يعني أن الرقم يمكن أن يأخذ أكثر من قيمة والذي يحدد ذلك مكانه داخل العدد (والذي يسمى بالمرتبة), تزداد قيمة العدد إذا حركتهُ باتجاه اليسار وتقل قيمته إذا حركيه باتجاه اليمين. فمثلاً العدد (937) نجد أن القيمة الحقيقية للرقم 7 هي سبعة فقط أما قيمة الرقم 3 فهي (30) وقيمة الرقم 9 هي (900).

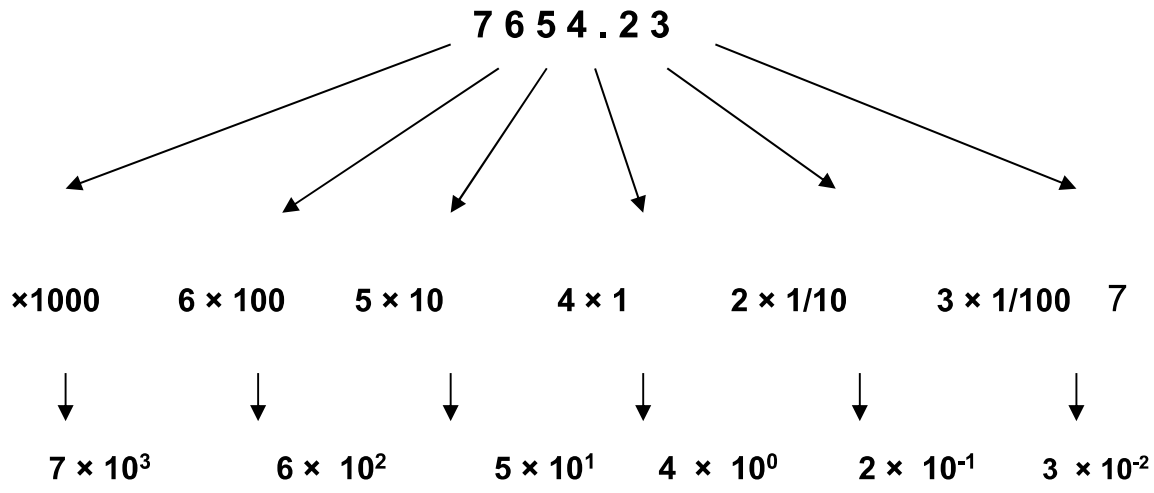
وهنالك أنظمة عددية أخرى غير النظام العشري , وأكثرها شيوعاً هي **النظام الثنائي, النظام الثماني, النظام السادس عشري**. وتكون هذه الأنظمة مفيدة في الأنظمة الرقمية مثل الحاسبات الالكترونية , المعالجات الدقيقة , وغيرها من الأنظمة الرقمية. ولهذا السبب فانه من الضروري الاطلاع على كل من هذه الأنظمة العددية لغرض استخدامها في دراستنا للأنظمة الرقمية.

النظام العشرى : Decimal System

وهو النظام العددي المتعارف عليه والمستخدم في كافة المجالات وفي كل انحاء العالم وجاءت تسمية النظام ب(العشري) لان عدد الرموز الداخلة في تركيبة أي عدد في هذا النظام هي عشرة رموز وهي (0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9) وفي حالة استخدام اكثر من رمز فان القيمة العددية تعتمد على موقع الرمز ضمن سلسلة الرموز , ان عدد الرموز الداخلة في تركيب

النظام العددي تسمى بأساس النظام , لذلك فان اساس النظام العشري هو العدد (10) وسمي بأساس العدد لان كل عدد مكتوب بهذا النظام يعتمد بالاساس على هذا العدد .

مثال : العدد العشري 7654.23 يمكن تحليله إلى المراتب التالية



النظام الثنائي: Binary System

وهو نظام عددي أساسه العدد (2) مقارنة بالنظام العشري الذي أساسه العدد (10) , أي ان عدد الرموز المستخدمة في النظام هي رمزين فقط وهي (0 , 1) لتمثيل كافة الاعداد . ويعتبر النظام الثنائي اساس اللغة التي تتعامل بها الحاسبة الالكترونية والأنظمة الرقمية , مثال على اعداد بهذا النظام :

1001 , 10111.101 , 10.1101 , 0.1011

من خلال ملاحظتنا الاعداد اعلاه نلاحظ بان الاعداد بالنظام الثنائي ولكن توجد اعداد شبيهه بها في النظام العشري , فلتمييز العدد المكتوب بالنظام المعين , تكتب الاعداد داخل اقواس مع كتابة رمز اسفل القوس يمثل اساس النظام المكتوب به العدد .

فمثلا : العدد 110 يكتب بالثنائي $(110)_2$ وبالعشري $(110)_{10}$

مثال : لتحليل العدد $(110.101)_2$ الى مراتبه :

$$(110.101)_2 = 0\times2^0 + 1\times2^1 + 1\times2^2 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$$

النظام الثماني : Octal System

وهو من الانظمة المستخدمة في الحاسبات الالكترونية أساسه العدد (8) , الرموز المستخدمة في هذا النظام هي (0 , 1 , 2 , 3 , 4 , 5 , 6 , 7) مثال على إعداد النظام الثماني

$(0.513)_8$, $(721.5)_8$, $(203.62)_8$, $(110.013)_8$

مثال : حلل العدد $(203.65)_8$ الى مراتبه

$$(203.65)_8 = 3 \times 8^0 + 0 \times 8^1 + 2 \times 8^2 + 6 \times 8^{-1} + 5 \times 8^{-2}$$

$$= 3 \times 1 + 0 \times 8 + 2 \times 64 + 6 \times 1/8 + 5 \times 1/64$$

النظام السادس عشرى : Hexadecimal System

وهو من الانظمة المهمة المستخدمة في الحاسبات الالكترونية أساسه العدد (16) أي إن عدد الرموز المستخدمة في تشكيل أعداد النظام هي 16 رمز وهي :

(F , E , D , C , B , A , 9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1, 0)

ومثال على أعداد بالنظام السادس عشري :

$(2D6.F3)_{16}$, $(10011.1)_{16}$, $(FFF)_{16}$, $(0.257)_{16}$

مثال :: حلل العدد $(3A1.7F)_{16}$ إلى مراتبه :

$$(3A1.7F)_{16} = 1 \times 16^0 + 10 \times 16^1 + 3 \times 16^2 + 7 \times 16^{-1} + 15 \times 16^{-2}$$

$$= 1 \times 1 + 10 \times 16 + 3 \times 256 + 7 \times 1/16 + 15 \times 1/256$$

ملاحظة : عند مقارنة الرموز السادس عشرية بالنظام العشري فان الرموز(A ← F) تساوي في النظام العشري (10 ← 15).

التحويلات بين الأنظمة العددية

أن عملية التحويل بين الأنظمة العددية من العمليات المهمة والتي يجب إن يتعرف عليها الشخص الذي يدرس عملية تصميم الأنظمة الرقمية . ولتسهيل عملية فهم هذه التحويلات سيتم تقسيمها إلى مجاميع كل مجموعة تتشابه بطريقة التحويل .

التحويل من الأنظمة (غير العشرية) إلى النظام العشري :

لتحويل أي عدد من أي نظام عددي إلى نظام العشري يتم تحليل العدد إلى مراتبه اعتمادا على أساس ذلك النظام ثم إيجاد ناتج جمع الحدود ، والعدد الناتج من الجمع سيكون هو العدد في النظام العشري .

مثال : حول العدد $(1101.01)_2$ إلى النظام العشري :

$$(1101.01)_2 = 1\times 2^0 + 0\times 2^1 + 1\times 2^2 + 1\times 2^3 + 0\times 2^{-1} + 1\times 2^{-2}$$

$$= 1\times 1 + 0\times 2 + 1\times 4 + 1\times 8 + 0\times 1/2 + 1\times 1/4$$

$$= 1 + 0 + 4 + 8 + 0 + 0.25$$

$$= (13.25)_{10}$$

التحويل من النظام العشري إلى الأنظمة الأخرى :

لتحويل أي عدد عشري إلى أي نظام أخر يجب تجزئته إلى جزء صحيح وجزء كسري وتحويل كل جزء بطريقة خاصة ثم جمع ناتج التحويل للجزئين للحصول على الناتج النهائي .

أولاً: تحويل الجزء الصحيح :

لتحويل الجزء الصحيح للعدد العشري لأي نظام نقوم بتقسيم العدد العشري على أساس النظام المطلوب التحويل إليه ونحتفظ بباقي القسمة ، ثم نأخذ ناتج القسمة ونقسمه مرة أخرى على أساس

النظام ونحتفظ بالباقي وهكذا نستمر بتكرار العملية إلى أن نحصل على ناتج قسمة يساوي صفر . فيكون ناتج التحويل في عمود باقي القسمة بقراته من الأسفل إلى الأعلى وكتابته من اليسار إلى اليمين

ثانياً: تحويل الجزء ألكسري :

لتحويل الجزء ألكسري من العدد العشري إلى نظيره في الأنظمة الأخرى نقوم بضرب العدد ألكسري في أساس النظام المطلوب التحويل إليه ثم اخذ الجزء ألكسري فقط من ناتج الضرب وضربه

مرة أخرى في الأساس وهكذا تستمر عملية الضرب إلى أن نتوقف في إحدى الحالات التالية :

- إما أن يكون الجزء ألكسري الناتج في الضرب **يساوي صفر .**
- تكرار الجزء ألكسري أكثر من مرة .
- تعقيد الجزء ألكسري أكثر مع استمرار عملية الضرب .

بعد توقف عملية الضرب يتم قراءة ناتج التحويل في عمود الجزء الصحيح من الضرب بقراءته من الأعلى إلى الأسفل وكتابته بعد الفارزة من اليسار إلى اليمين .

مثال: حول العدد $(34.56)_{10}$ إلى النظام الثنائي :

الجزء الصحيح

	ناتج القسمة	باقي القسمة
34÷ 2=	17	0
17÷ 2=	8	1
8÷ 2=	4	0
4÷ 2=	2	0
2÷ 2=	1	0

1÷ 2=	0	1	$(100010)_2$

الجزء العشري:

0.56× 2 =	1.12	1
0.12× 2 =	0.24	0
0.24× 2 =	0.48	0
0.48× 2 =	0.96	0
0.96× 2 =	1.92	1
0.92× 2 =	1.84	1
0.84× 2 =	1.68	1
0.68× 2 =	1.36	1
0.36× 2 =	0.72	0
0.72× 2 =	1.44	1
0.44× 2 =	0.88	0
.88× 2 =	1.76	1

$(0.100011110101\ldots)_2$

النتيجة :

$(100010.100011110101\ldots)_2$

تمارين (3)

1. حول العدد العشرى $(343)_{10}$ الى النظام الثنائى ؟
2. حول الاعداد العشرية الاتية الى الثنائى :

a) 64 b) 112 c) 257 d) 27.26

e)77.0625 f) 47.875 g) 33.125

3 . حول الاعداد الثنائية الاتية الى النظام العشرى :-

a) 11011 b) 1110101 c) 111111 d) 1110.11

e) 10101.1101 f) 1100001.11011

4. حول الاعداد من النظام الثمانى الى النظام الثنائى :-

a) 72 b) 113 c) 16.3 d) 37.6

e) 122.775 f) 417.632 g) 37.6

5. حول الاعداد من النظام الثنائى الى النظام الثمانى:-

a) 110101 b) 11110100.110101 c) 110110111.10101

d) 10001001011.1001 e) 1010111.11101

6. حول العدد العشرى $(225.625)_{10}$ الى ما يناظره بالنظام الثنائى ، النظام السادس عشر .

7. حول العدد الثنائى $(11010101)_2$ الى ما يناظره بالنظام الثمانى والسادس عشر

8. حول من النظام العشرى الى النظام السادس عشر :

a) 14 b) 80 c) 560 d) 3000

e) 62500 f) 204.125 g) 255.875 h)631.25

تمثيل البيانات داخل الحاسب الآلي:

يستخدم الحاسب نظم التشفير (Coding Systems) لتمثيل الرموز المختلفة الموجودة على لوحة المفاتيح بطريقة ثنائية ومن هذه النظم ما يلي:

١) نظام BCD :

وهي اختصار لـ Binary Coded Decimal والتي تعني نظام الأرقام العشرية الممثلة ثنائياً. يمثل هذا النظام الرموز بواسطة 6 خانات (6 Bits) .

يوضح الجدول التالي طريقة تمثيل البيانات بنظامBCD :

منطقة الدليل	التمثيل الرقمي للرمز									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
00	0	1	2	3	4	5	6	7	8	9
11		A	B	C	D	E	F	G	H	I
10		J	K	L	M	N	O	P	Q	R
01			S	T	U	V	W	X	Y	Z

يتكون هذا النظام فقط من ٦ خانات مما يعني أن عدد رموز هذا النظام هي(2^6)أي 64 رمز فقط.

مثال تمثيل KFU بنظام BCD يكون كالآتي:

100010110110010100

٢) نظام EBCDIC :

وهي اختصار لـ Extended Binary Coded Decimal Interchange والتي تعني الشفرة الموسعة للنظام العشري الممثل ثنائياً ويستخدم هذا النظام 8 خانات لتمثيل الرموز المختلفة الموجودة على لوحة المفاتيح كما مبين بالجدول التالي:

منطقة الدليل	التمثيل الرقمي للرمز									
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
1111	0	1	2	3	4	5	6	7	8	9
1100		A	B	C	D	E	F	G	H	I
1101		J	K	L	M	N	O	P	Q	R
1110			S	T	U	V	W	X	Y	Z

يستخدم هذا النظام 8 خانات لتمثيل الرموز مما يعني 2^8 احتمال أي 256 رمز .

مثال: مثلي كلمة KFU بنظام EBCDIC .

— 110100101100011011100100

٣) نظام ASCII:

وهي اختصار لـ American Standard Code for Information Interchange والتي تعني الشفرة الأمريكية القياسية لتبادل المعلومات.

ويوجد نظامان لآسكي أحدهما يستخدم 7 خانات لتمثيل الرموز الموجودة على لوحة المفاتيح بـ 2^7 =128 رمز. والآخر يستخدم8 خانات وبالتالي يمكن تمثيل 2^8 أي 256حرف.

يوضح الجدول التالي طريقة تمثيل البيانات بنظام(ASCII (7-Bit :

الدليل	التمثيل الرقمي للرمز															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
011	0	1	2	3	4	5	6	7	8	9						
100		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
101	P	Q	R	S	T	U	V	W	X	Y	Z					

مثال: مثلي KFU بنظام (ASCII (7-Bit.

1001011100011010101010101

ويوضح الجدول التالي طريقة تمثيل البيانات بنظام(ASCII (8-Bit :

الدليل	التمثيل الرقمي للرمز															
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0101	0	1	2	3	4	5	6	7	8	9						
1010		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1011	P	Q	R	S	T	U	V	W	X	Y	Z					

فيصبح تمثيل KFU بنظام (ASCII (8-Bit كالآتي:

1010101110100110101101010101

تمرين

مثلي كلمة COMPUTER

(أ) بنظام BCD.

(ب) بنظام EBCDIC.

(ج) بنظام ASCII (7-Bit).

(د) بنظام ASCII (8-Bit).

بت التدقيق Parity Bit (check digit) :

هي بت إضافية تلحق بالبايت المنقول من المعلومات وتستخدم في الكشف عن الأخطاء عند نقل المعلومات.

فالحاسب الذي يستخدم شفرة مكونة من 8 Bits مثل نظام EBCDIC سوف تكون هنالك بت تاسعة إضافية لغرض التدقيق.

إحدى طرق التأكد من أن المعلومات نقلت بصورة صحيحة هي أن يتم إرسالها مرتين والمقارنة بين الإرسالين، لكن هذه الطريقة قليلة الكفاءة لأنها تضاعف الوقت والتكلفة.

لذا تم استخدام بت التدقيق كحل بديل لتلك الطريقة.

هنالك حواسيب يكون فيها عدد خانات الرقم 1 في كل بايت مرسل عدد زوجي وتسمى حواسيب زوجية التدقيق، بينما الحواسيب التي يكون فيها عدد خانات الرقم واحد عدداً فردياً تسمى حواسيب فردية التدقيق.

في الحواسيب زوجية التدقيق إذا كان عدد خانات الرقم 1 عدداً فردياً في البايت المرسل، فان بت التدقيق يجب أن تكون1. وذلك لضمان أن يكون عدد خانات الرقم 1 عدد زوجي.

وإذا كان عدد خانات الرقم 1 زوجياً فان بت التدقيق المضافة يجب أن تكون صفر تلقائياً.

مثلاً: تمثيل الرقمين 2 و 3 على حاسب زوجي التدقيق باستخدام نظام EBCDIC هو

2 تمثيلها: | 1 | 11110010 | ← بت التدقيق (وقيمتها 1 لان عدد خانات 1 عدد فردي)

3 تمثيلها: | 0 | 11110011 | ← بت التدقيق (وقيمتها 0 لان عدد خانات 1 عدد زوجي)

عندما يتم إرسال بيانات حاسب زوجي التدقيق فان الحاسب المستقبل يفترض أن عدد خانات الرقم 1 المستقبل في كل بايت يجب أن يكون زوجي فإذا أكتشف الحاسب أن هنالك بايت به عدد فردي فانه يطلب تلقائياً إعادة الإرسال Retransmission .

أما الحواسيب فردية التدقيق فإنها تعمل بنفس المفهوم باستثناء أنها تستخدم بت التدقيق للتأكد من أن عدد خانات الرقم 1 المرسلة يجب أن يكون فردياً وإلا فانه يطلب إعادة الإرسال.

جدول تخصيص الملفات (File Allocation Table(FAT)) :

يستخدم العدد الثنائي ذو الـ 16 بت (خانة) للإشارة إلى عدد الوحدات التخزينية التي تخزن على القرص الصلب حيث تسع كل وحدة تخزينية 512 Bytes (½ KB) وهذه الوحدات تسمى القطاعات (Sectors) وبما أن العدد الثنائي 2^{16} = 65536 لذا يشير FAT 16 إلى 65536 عنوان (Address) للوحدات التخزينية أي ما يقارب

512 B * 65536 = 33554432 Byte = 32768 KB = 32 MB

وهذا يعني أن الحجم الأقصى للقرص الصلب الذي يستطيع جدول تخصيص الملفات FAT 16 التعامل معه هو 32 MB . وكان هذا هو الحد الأقصى لأول قرص صلب ولكن مع التطور وتزايد أحجام البرامج كان لابد من استخدام أحجام أكبر للقرص الصلب فكان لابد من وجود طريقة لجدولة مواقع الملفات على هذا القرص، لذلك تم استخدام تعبير جديد هو العنقود (Cluster) والذي هو عبارة عن مجموعة من القطاعات (Sectors).

على القرص الواحد حدد لكل عنقود ثمانية قطاعات فأصبح حجم العنقود 4 KB (8x ½ KB)

وهذا يعني أننا باستخدام FAT 16 يمكننا التجوال ضمن مساحة تخزينية تساوي :

4 KB x 65536 =262144 KB =256 MB

ولكن التطور في الأحجام الضخمة للبرامج لم يتوقف وكان لابد من السعي من جديد للتغلب على هذا الموضوع ، لذلك خصص لكل عنقود عدد أكبر من القطاعات وصل إلى 64 قطاع وبالتالي أصبح حجم العنقود :

64 x 512 B = 32768 B = 32 KB

وهذا ذاد من الحجم الأقصى للقرص الصلب فأصبح:

65536 x 32 KB =2097152 KB = 2048 MB = 2 GB

وهذا هو الحد الأقصى للقرص الصلب الذي نستطيع التعامل معه بنظام Dos ونظام Windows95 . وبالتالي يجب تجزئة الأقراص الصلبة ذات الأحجام أكبر من 2GB إلى عدة أجزاء باستخدام الأمر FDisk وبحد أقصى للكل جزء لا يزيد عن 2 GB .

أصبح الآن بإمكاننا تخصيص عدد أكبر من القطاعات لكل عنقود وبالتالي زيادة حجم القرص الصلب (أو جزء منه) وبمجال عناوين لا يزيد عن 65536 (2^{16}) وهذا ربح جيد.

إذا كانت لدينا ملفات صغيرة الحجم مثل بعض ملفات نظام الويندوز التي لا تزيد عن 128 byte (حرف) وحيث أننا لا نستطيع تخزين أكثر من ملف واحد في العنقود الواحد حتى ولو كان حجم هذه الملفات صغيراً جداً لأنه لا يمكن الإشارة إلى عنقود (cluster) واحد من خلال عنوانين في جدول تخصيص الملفات وبالتالي سيتم حجز أماكن إضافية فائضة لمثل هذه الملفات صغيرة الحجم .

لذلك لجأت شركة مايكروسوفت جدول مواقع الملفات FAT 32 مع نظام Windows 98 وما بعده مع المحافظة على حجم 4KB لكل عنقود وبهذه الطريقة نستطيع باستخدام FAT 32 تخصيص حجم هائل لكل جزء من أجزاء القرص الصلب يساوي:

$2^{32} \times 4\ KB = 16\ TB$

الدوائر المنطقية

Logical Circuits

ما هى البوابة المنطقية (logical gate) هى دائرة الكترونية تحتوى على (مدخل او عدة مداخل) ومخرج واحد حيث تقوم بعملية منطقية على المدخل وتنتج المخرج المطلوب . تستخدم هذه البوابات فى بناء معالجات الاجهزة الاكترونية والحواسيب لان مخرج البوابة الرقمية هو ايضا قيمة منطقية فانه يمكن استخدام مخرج احد البوابات المنطقية كمدخل لبوابة اخرى .

المنطق المستخدم غالبا هو المنطق البوليانى (Boolean Logic) وهو المنطق الذى يعمل فى الدوائر الرقمية

Digital logic circuits ⇨ electronic circuits that handle information encoded in binary form (deal with signals that have only two values, **0** and **1**)

♦ ***Digital*** …. computers, watches, controllers, telephones, cameras, ...

★ **BINARY NUMBER SYSTEM**

Number ….in whatever base	*Decimal value of the given number*
Decimal: **1,998**	$= 1x10^3+9x10^2+9x10^1+8x10^0 = 1,000+900+90+8 = \mathbf{1,998}$
Binary: **11111001110**	$= 1x2^{10}+1x2^9+1x2^8+1x2^7+1x2^6+1x2^3+1x2^2+1x2^1 =$ $1,024+512+258+128+64+8+4+2 = \mathbf{1,998}$

Powers of 2

N	2^N	*Comments*
0	1	
1	2	
2	4	
3	8	
4	16	
5	32	
6	64	
7	128	
8	256	
9	512	
10	1,024	"Kilo"as 2^{10} is the closest power of 2 to 1,000 (decimal)
11	2,048	
15	32,768	2^{15} Hz often used as clock crystal frequency in digital watches
20	1,048,576	"Mega" as 2^{20} is the closest power of 2 to 1,000,000 (decimal)
30	1,073,741,824	"Giga" as 2^{30} is the closest power of 2 to 1,000,000,000(decimal)

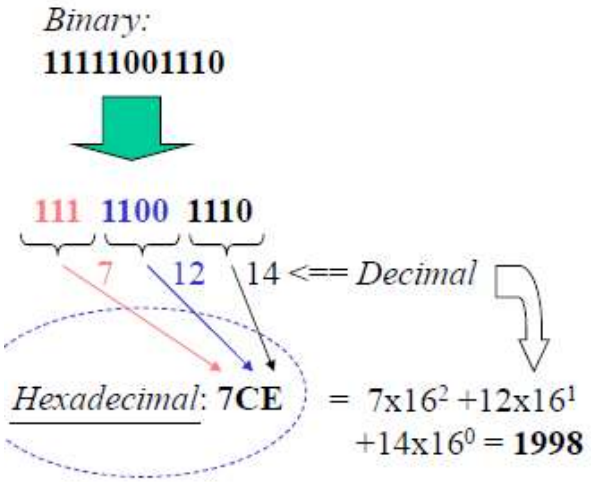
Negative Powers of 2

N <0	2^N
-1	$2^{-1} = 0.5$
-2	$2^{-2} = 0.25$
-3	$2^{-3} = 0.125$
-4	$2^{-4} = 0.0625$
-5	$2^{-5} = 0.03125$
-6	$2^{-6} = 0.015625$
-7	$2^{-7} = 0.0078125$
-8	$2^{-8} = 0.00390625$
-9	$2^{-9} = 0.001953125$
-10	$2^{-10} = 0.0009765625$
…	

Binary numbers less than 1

Binary	*Decimal value*
0.101101	$= 1x2^{-1} + 1x2^{-3} + 1x2^{-4} + 1x2^{-6} =$ **0.703125**

HEXADECIMAL



Binary	Decimal	**Hexadecimal**
0000	0	**0**
0001	1	**1**
0010	2	**2**
0011	3	**3**
0100	4	**4**
0101	5	**5**
0110	6	**6**
0111	7	**7**
1000	8	**8**
1001	9	**9**
1010	10	**A**
1011	11	**B**
1100	12	**C**
1101	13	**D**
1110	14	**E**
1111	15	**F**

LOGIC OPERATIONS AND TRUTH TABLES

Digital logic circuits handle data encoded in binary form, i.e. signals that have only two values, **0** and **1**.

Binary logic dealing with "true" and "false" comes in handy to describe the behaviour of these circuits: **0** is usually associated with "**false**" and **1** with "**true**."

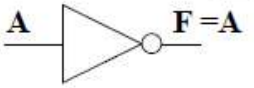
Quite complex digital logic circuits (e.g. entire computers) can be built using a few *types of basic circuits* called **gates,** each performing a single elementary logic operation : *NOT, AND, OR, **NAND**, NOR*, etc..

Boole's binary algebra is used as a formal / mathematical tool to describe and design complex binary logic circuits.

GATES

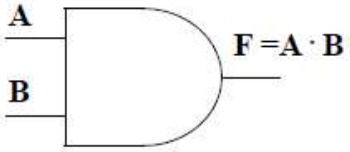
A	$\overline{A}$
0	1
1	0

NOT



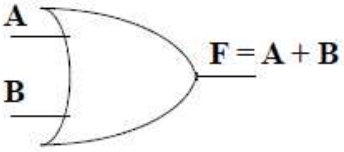
A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

AND



A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

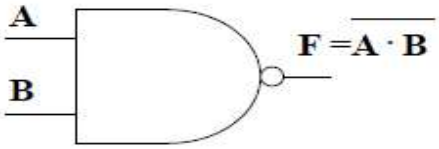
OR



... more GATES

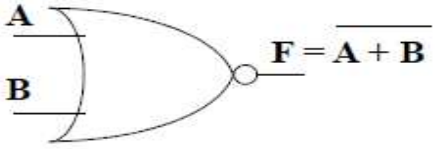
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

NAND



A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

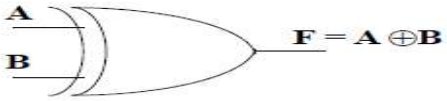
NOR



... and more GATES

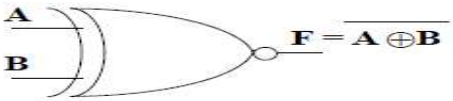
A	B	A ⊕B
0	0	0
0	1	1
1	0	1
1	1	0

XOR



A	B	$\overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

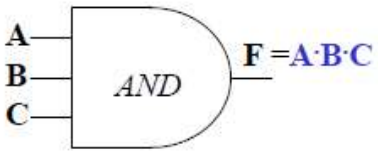
EQU or *XNOR*

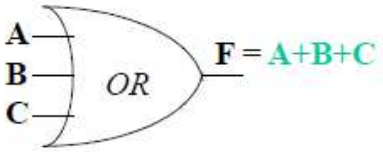


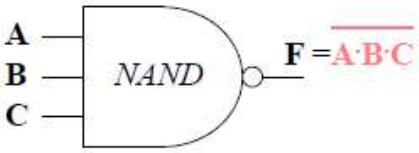
GATES ... with more inputs

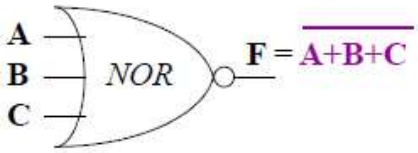
EXAMPLES OF GATES WITH THREE INPUTS

A	B	C	A·B·C	A+B+C	$\overline{A \cdot B \cdot C}$	$\overline{A+B+C}$
0	0	0	0	0	1	1
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	1	0	0



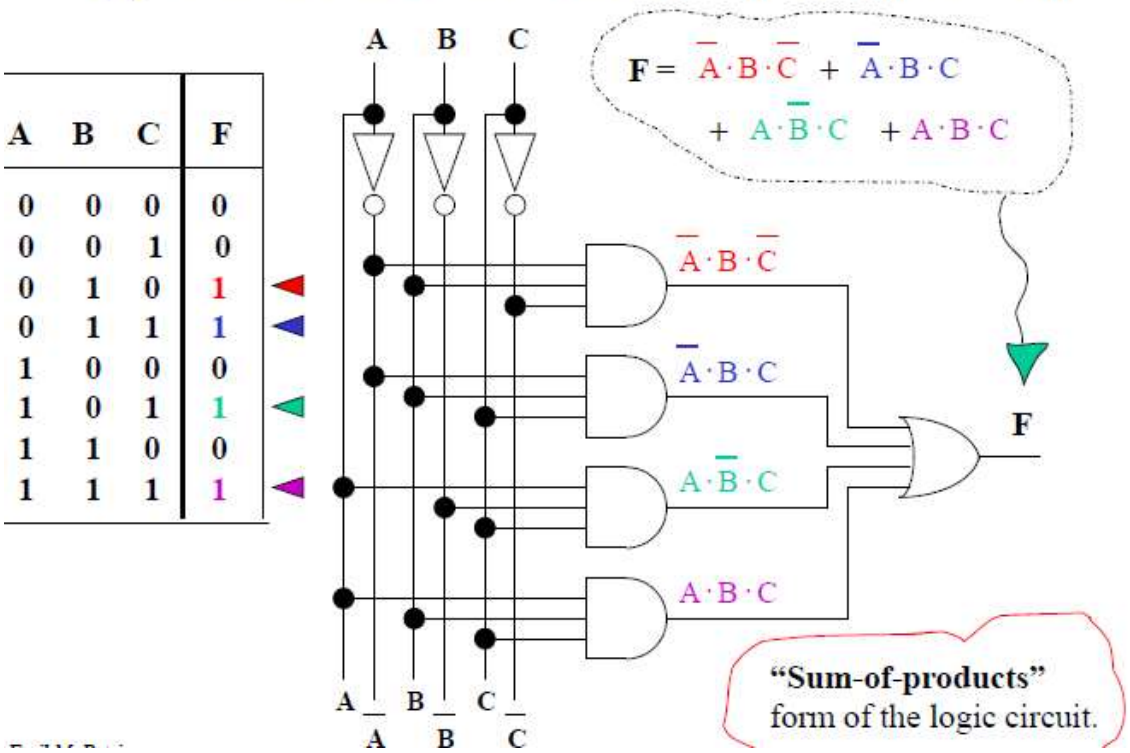






Logic Gate Array that Produces an Arbitrarily Chosen Output

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



BOOLEAN ALGEBRA

AND rules
$A\cdot A = A$
$A\cdot\bar{A} = 0$
$0\cdot A = 0$
$1\cdot A = A$
$A\cdot B = B\cdot A$
$A\cdot(B\cdot C) = (A\cdot B)\cdot C$
$A\cdot(B + C) = A\cdot B + A\cdot C$
$\overline{A\cdot B} = \bar{A} + \bar{B}$

" Proof ":

A B C	A·(B+C)	A·B+A·C
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	0	0
1 0 0	0	0
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

BOOLEAN ALGEBRA ... continued

OR rules
$A + A = A$
$A + \overline{A} = 1$
$0 + A = A$
$1 + A = 1$
$A + B = B + A$
$A + (B + C) = (A + B) + C$
$A + B \cdot C = (A + B) \cdot (A + C)$
$\overline{A} + \overline{B} = \overline{A \cdot B}$

A B C	A + B·C	(A+B) · (A+C)
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	1
1 0 0	1	1
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

DeMorgan's Theorem

$\overline{A \cdot B} = \overline{A} + \overline{B}$

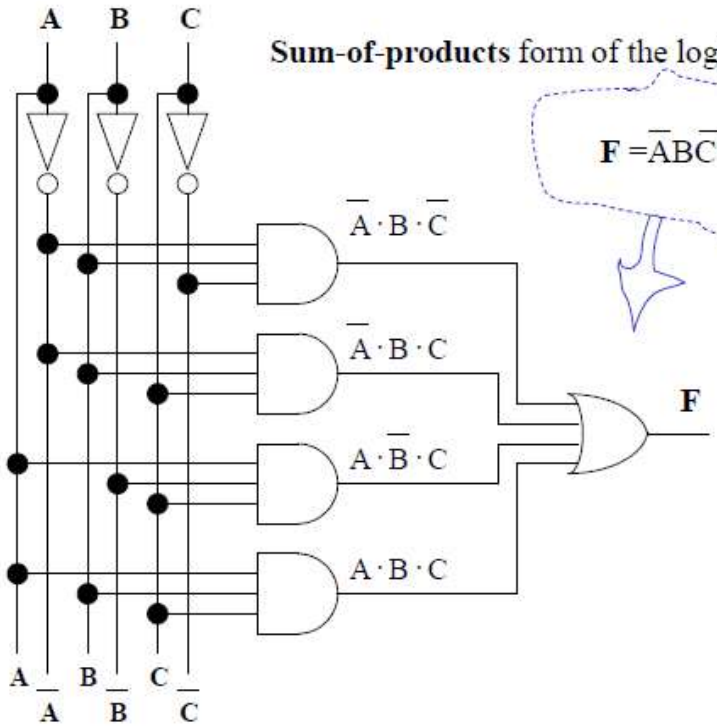
$\overline{A + B} = \overline{A} \cdot \overline{B}$

A	B	$\overline{A} \cdot \overline{B}$	$\overline{A + B}$	$\overline{A} + \overline{B}$	$\overline{A \cdot B}$
0	0	1	1	1	1
0	1	0	0	1	1
1	0	0	0	1	1
1	1	0	0	0	0

Simplifying logic functions using Boolean algebra rules

Sum-of-products form of the logic function:

$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$



Simplifying logic functions using Boolean algebra rules … continued

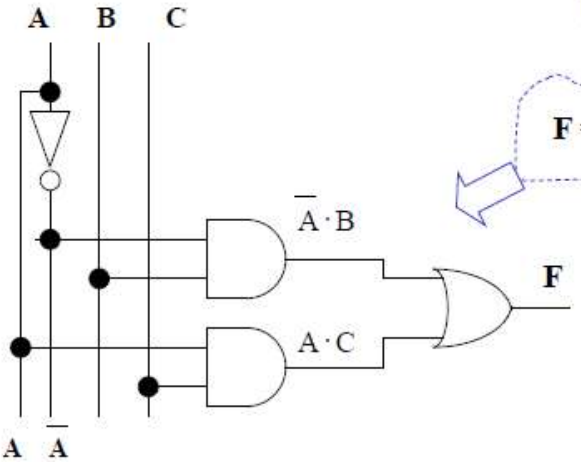
$$F = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + ABC$$

$$F = (\overline{A}B\overline{C} + \overline{A}BC) + (A\overline{B}C + ABC)$$

$$F = \overline{A}(B\overline{C} + BC) + A(\overline{B}C + BC)$$

$$F = \overline{A}B(\underbrace{\overline{C} + C}_{1}) + AC(\underbrace{\overline{B} + B}_{1})$$

$$F = \overline{A}B + AC$$



الفصل الرابع
مقدمة عن لغة البايثون

يقدم هذا الفصل انواع لغات البرمجة ومميزات وعيوب كل لغة من لغات الحاسب ويركز هذا الفصل على لغة البايثون وهى الموضوع الرئيسى للدراسة والخطوات الرئيسية لكتابة برنامج بلغة البايثون.

لغات البرمجة هى لغات خاصة بالتعامل مع الحاسب وتعتبر لغات البرمجة هى الوسيلة الوحيدة للتعامل مع الحاسب الالى وهى عبارة عن مجموعة من التعليمات والاوامر التى توجه الى الحاسب لتادية عمليات معينة، وأي لغة من هذه اللغات يجب ان تكون قادرة على تمثيل الحروف والارقام والحروف الخاصة والجدير بالذكر ان لغات البرمجة تصنف الى :-

1- لغات منخفضة المستوى Low Level Language
   1.1 لغة الالة
   1.2 لغة التجميع
2- لغات عالية المستوى High Level Language

هناك العديد من اللغات عالية المستوى مثل لغة C ، الفيجول بيزيك ، الفورتران ، الباسكال ، الجافا وغيرها وهى القريبة من لغة الانسان .

ما هى لغة البايثون ؟

هى لغة عالية المستوى، وتفسيرية (اى تحتاج مفسر للاكواد للتنفيذ) وتدعم البرمجة الكائنية . صممت لغة البايثون لتكون سهلة القراءة وتستخم الكلمات الانجليزية وهى قليلة القواعد عن اللغات الاخرى .

خصائص لغة البايثون :-

تشمل لغة البايثون الخصائص التالية :-

a) سهلة التعلم : تمتلك لغة البايثون كلمات اساسية قليلة ، وتركيبة بسيطة وقواعد سهلة وواضحة .مما يسمح للطالب تعلمها بسهولة

b) سهولة القراءة : الكود معرف بوضوح وواضح للعين .

c) سهولة تصحيح الكود

d) تحتوى على مكتبة محمولة ومتوافقة مع كل انظمة التشغيل مثل الويندوز ، اليونكس ، الماكنتوش.

e) سهولة عمل تطبيقات بواجهات رسومية GUI

تحميل برنامج البايثون :-

يتم التعرف على ما هو جديد فى لغة البايثون وايضا تحميل البرنامج من خلال الموقع التالى : http://www.python.org/.

القواعد الاساسية للغة البايثون

Basic Syntax

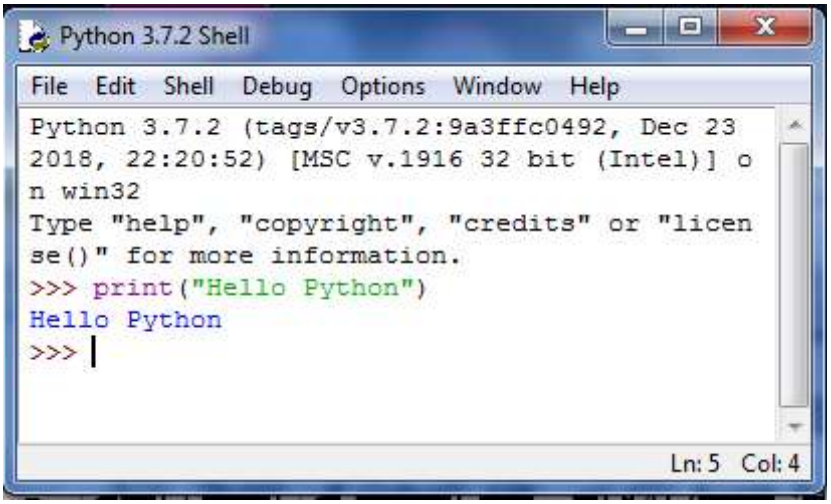
تتشابه لغة البايثون مع العديد من لغات البرمجة عالية المستوى مثل C ، Perl ، Java كما يوجد بعض الاختلافات المحددة بين اللغات .

اول برنامج بلغة البايثون :-

تنفذ برنامج لغة البايثون فى وضعين وهما :-

1- Interactive Mode Programming:-

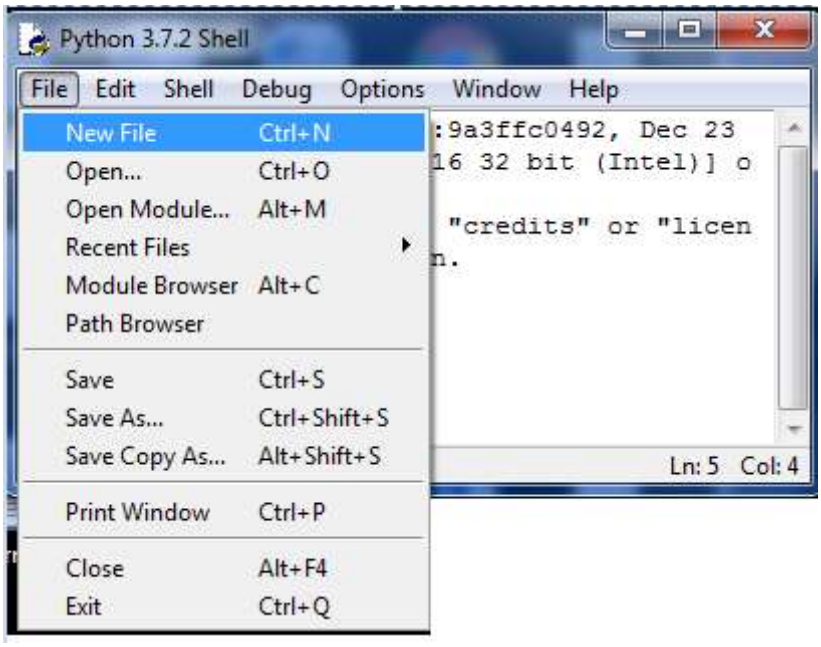
يتم كتابة جميع الاوامر عند المحث فى الشكل التالى :-

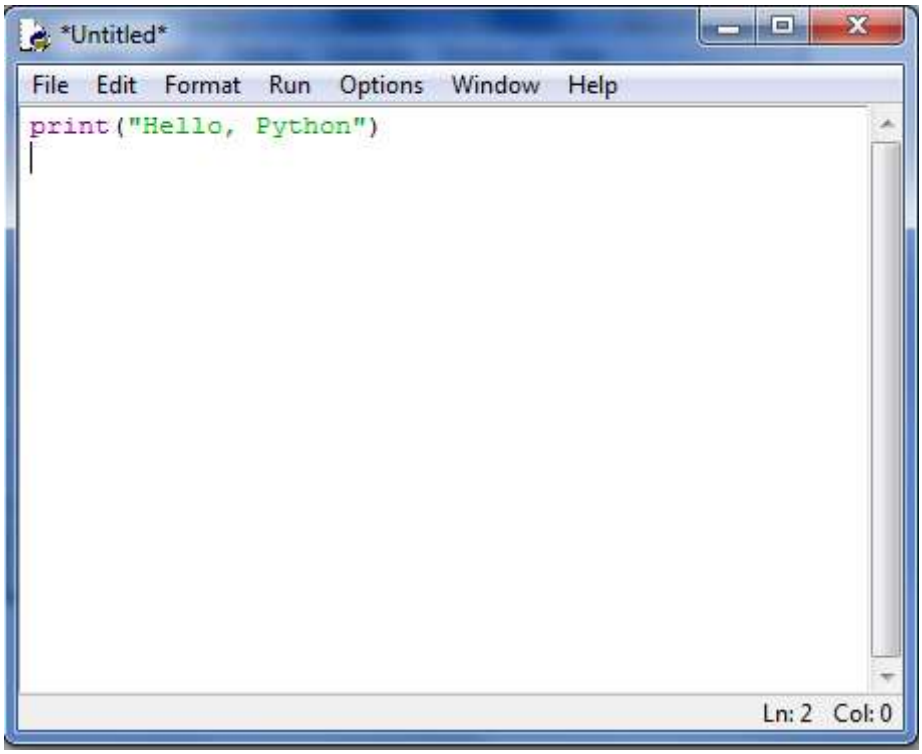


2- Script Mode Programming:

يتم هناك كتابة الاوامر والاكواد داخل ملف يتم فتح الملف كالتالى :

File→ New File





يتم حفظ الملف عن طريق File→ save as يتم اختيار اسم للملف ويتم وضع الامتداد .py

مثلا first.py

يتم حفظ الملف الخاص بالبايثون داخل مسار البرنامج مثل

C:\Programs\Python\Python7-32

المعرفات فى لغة البايثون :-

-عبارة عن اسم يستخدم لتعريف متغير ، دالة ، class ، module او اى كائن اخر . يجب ان يبدا المعرف بحرف ابجدى (Ato Z) او (a to z) او بشرطة سفلية (underscore) متبوعة بحروف وارقام (0 to 9) .

-لا يسمح باستخدام الرموز الخاصة مثل %, $, @ خلال اى معرف

- لغة البايثون حساسة بالنسبة للحروف الكبيرة والصغيرة اى Case Sensitive فمثلا المعرف First والمعرف first مختلفان .

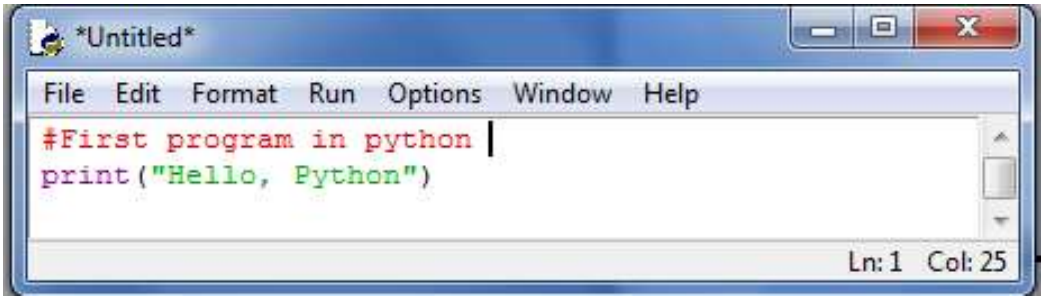
الكلمات الاساسية فى لغة البايثون Keywords:-

الجدول التالى يوضح بعض الكلمات الاساسية فى لغة البايثون والتى لا تستخدم كاسماء للمتغيرات او اى معرفات كما انها تكون حروف صغيرة

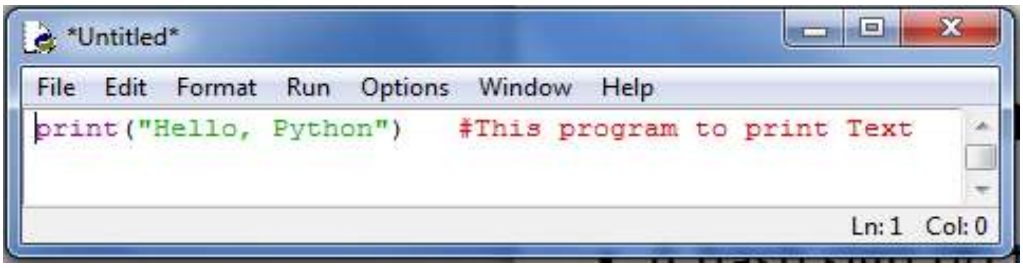
import	global	if
return	while	for
pass	try	print
raise	in	elif
or	from	del
class	finally	def

التعلقيات فى لغة البايثون :

الرمز المستخدم فى التعليق هو # حيث ان الكلمات التى تليه لا يقوم المترجم بتنفيذها مثل



يمكن ان يكون التعليق على نفس السطر لتوضيح الهدف من الامر مثل



كتابة اكثر من جملة على نفس السطر Multiple Statements on a Single Line

تسمح علامة الفاصلة المنقوطة (;) فى كتابة العديد من الجمل البرمجية على نفس السطر مثل

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

كما يمكن كتابة العديد من الجمل فى شكل بلوك يسمى suits كما بالمثال التالى :

```
if expression :
   suite
elif expression :
   suite
else :
   suite
```

أنواع المتغيرات Variable Types

ما هو المتغير ؟

عبارة عن حجز مواقع داخل الذاكرة لتخزين القيم . وهذا يعنى عندما تقوم بانشاء متغير فانك تحجز مكان له فى الذاكرة ز

بناء على نوع البيانات فى المتغير فان المفسر يقوم بحجز مساحة داخل الذاكرة ويمكن ان تكون المتغيرات من نوع اعداد صحيحة، اعداد حقيقية ، او حروف .

تخصيص قيمة لمتغير :-

يتم وضع اسم المتغير ثم علامة = ويليه قيمة المتغير مثل

```
counter = 100 # An integer assignment
miles = 1000.0 # A floating point
name = "John" # A string
print (counter)
print (miles)
print  (name)
```

النتيجة :

```
100
1000.0
John
```

يسمح البايثون بتخصيص قيمة واحدة لنفس المتغير مثل

```
a = b = c = 1
```

كما يمكن تخصيص العديد من المتغير لعديد من القيم مثل

```
a, b, c = 1, 2, "john"
```

حيث يتم وضع المتغير a يساوى 1 ، والمتغير b يساوى 2 ، والمتغير c يساوى john

انواع البيانات القياسية Standard Data Types :-

يوجد خمسة انواع من المتغيرات فى البايثون :-

1- الارقام Number: لتخزين القيم الرقمية مثل

```
var1 = 1
var2 = 10
```

تدعم لغة البايثون اربعة انواع من الارقام وهى :

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

الجدول التالى يوضح بعض الامثلة على الارقام

int	long	Float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J

2- السلاسل String

السلاسل فى لغة البايثون عبارة عن مجموعة من الحروف بين علامات التنصيص . وتسمح البايثون باستخدام علامات التنصيص المفردة او الزوجية . ويمكن تعريف جزء من السلاسل باستخدام

([ ] and [:]) كما يبدا الترقيم فى السلاسل من الدليل 0 .

تستخدم العلامة (+) لربط السلاسل الحرفية بينما تستخدم العلامة (*) لتكرار السلاسل الحرفية

المثال التالى يوضح العمليات على السلاسل الحرفية :-

```
str = 'Hello World!'
print (str )                    # Prints complete string
print (str[0] )                 # Prints first character of the string
print (str[2:5])                # Prints characters starting from 3rd to 5th
print (str[2:])                 # Prints string starting from 3rd character
print  (str * 2 )               # Prints string two times
print  (str + "TEST")           # Prints concatenated string
```

النتيجة :-

```
Hello World!
H
llo
llo World!
Hello World! Hello World!
Hello World!TEST
```

3- قائمة List

القائمة عبارة عن مجموعة من العناصر يتم الفصل بينها بالفاصلة ويتم وضع العناصر داخل اقواس مربعة [ ]وهى مشابهة للمصفوفة فى لغة C ولكن عناصر القائمة يمكن ان تكون مختلفة .

يمكن الوصول لعناصر القائمة عن طريق استخدام ([ ] and [:]) متبوعة بالدليل والذى يبدا ب صفر. يمكن استخدام الرمز * ويمكن تجميع قائمتين باستخدام الرمز + مثال :-

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print (list) # Prints complete list
print (list[0]) # Prints first element of the list
print (list[1:3]) # Prints elements starting from 2nd till 3rd
print (list[2:]) # Prints elements starting from 3rd element
print (tinylist * 2) # Prints list two times
print (list + tinylist) # Prints concatenated lists
```

النتيجة :-

```
['abcd', 786, 2.23, 'john', 70.200000000000003]
abcd
[786, 2.23]
[2.23, 'john', 70.200000000000003]
[123, 'john', 123, 'john']
```

```
['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']
```

4- Tuple : هى نوع من data type فى لغة البايثون مشابهة ل List ولكن توضع داخل اقواس دائرية ( ) الفرق الثانى بين Tuple و List لا يمكن تعديل حجم البيانات داخل Tuple

مثال :-

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )
tinytuple = (123, 'john')
print (tuple)            # Prints complete list
print (tuple[0])         # Prints first element of the list
print (tuple[1:3])       # Prints elements starting from 2nd till 3rd
print (tuple[2:])        # Prints elements starting from 3rd element
print (tinytuple * 2)    # Prints list two times
print (tuple + tinytuple) # Prints concatenated lists
```

النتيجة

```
('abcd', 786, 2.23, 'john', 70.200000000000003)
abcd
(786, 2.23)
(2.23, 'john', 70.200000000000003)
(123, 'john', 123, 'john')
('abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john')
```

التحويل بين انواع البيانات Data Type Conversion

Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
long(x [,base])	Converts x to a long integer. base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.

list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

تمارين (4)

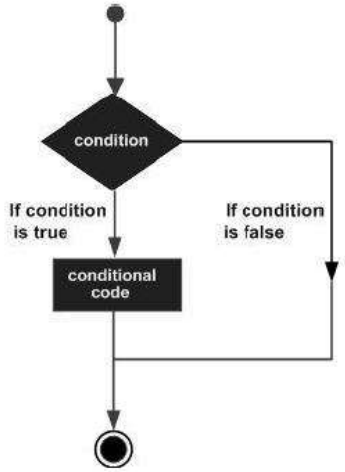
1- اكتب برنامج بلغة البايثون لحساب مساحة ومحيط دائرة نصف قطرها R
2- اكتب برنامج بلغة البايثون لحساب مساحة ومحيط مستطيل ابعاده L ، W
3- اكتب برنامج بلغة البايثون لحساب مساحة ومحيط مربع طوله L
4- اكتب برنامج بلغة البايثون لحساب اربعة متغيرات A, B, C, D
5- اكتب برنامج بلغة البايثون لحساب متوسط اربعة ارقام A, B, C, D

الفصل الخامس
جمل التحكم والتكرار
فى لغة البايثون

تؤدى البرامج فى الفصل الرابع عمليات حسابية بسيطة وطباعة الاجابات، ولكن كل جملة فى هذه البرامج تنفذ مرة واحدة فقط . غالبا البرامج المفيدة لديها خاصية تكرار مجموعة من الجمل عدد من المرات ومجموعة من الجمل المتتابعة تنفذ معتمدة على قيم البيانات المدخلة .

اتخاذ القرار (Decision Making) يقيم مجموعة من التعبيرات التى تنتج قيم True او False كمخرجات . تحتاج ان تحدد اى من الجمل تنفذ اذا كانت النتائج True او False

جمل البايثون التى تتحكم فى مجموعة من الجمل تسمى control Constructs الشكل التالى يوضح الشكل العام لتركيبة اتخاذ قرار (making Decision) الموجودة فى معظم لغات البرمجة.



الجدول التالى يوضح المعاملات العلاقية

Expression	Meaning
$x == y$	True if $x = y$ (mathematical equality, not assignment); otherwise, false
$x < y$	True if $x < y$; otherwise, false
$x <= y$	True if $x \leq y$; otherwise, false
$x > y$	True if $x > y$; otherwise, false
$x <= y$	True if $x \geq y$; otherwise, false
$x != y$	True if $x \neq y$; otherwise, false

امثلة :

Expression	Value
10 < 20	True
10 >= 20	False
x < 100	True if x is less than 100; otherwise, False
x != y	True unless x and y are equal

جملة If: فى لغة البايثون تتعدد اشكال جملة IF

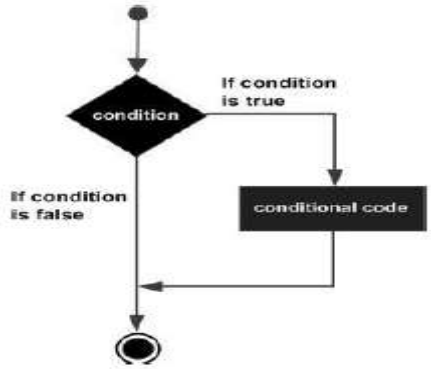
1- الشكل العام لجملة If يكون كالتالى :

```
if expression:

   statement(s)
```

اذا كان التعبير الجبرى يساوى TRUE فانه يتم تنفيذ مجموعة الجمل التى بداخل جملة If اما اذا كان التعبير الجبرى يساوى FALSE فانه يتم تنفيذ الجملة الاولى التى تاتى بعد نهاية جملة If.

الشكل التالى يعبر عن جملة If السابقة



1.py - C:/Users/Dell/AppData/Local/Programs/Python/Python37-32/1.py (3.7.2)

File Edit Format Run Options Window Help

```
var1 = 100
if var1:
      print( "1 - Got a true expression value")
      print (var1)
      var2 = 0
if var2:
      print ("2 - Got a true expression value")
      print (var2)
      print ("Good bye!")
```

Ln: 13 Col: 0

النتيجة :

```
1 - Got a true expression value

100

Good bye!
```

مثال :

```
# File betterdivision.py
# Get two integers from the user
dividend, divisor = eval(input('Please enter two numbers to divide: '))
# If possible, divide them and report the result
if divisor != 0:
  print(dividend, '/', divisor, "=", dividend/divisor)
```

النتيجة

```
Please enter two numbers to divide: 32,8
32 / 8 = 4.0
```

مثال:-

```
# Get two integers from the user
dividend, divisor = eval(input('Please enter two numbers to divide: '))
# If possible, divide them and report the result
if divisor != 0:
  quotient = dividend/divisor
  print(dividend, '/', divisor, "=", quotient)
print('Program finished')
```

ما هي نتيجة البرنامج ؟

مثال :

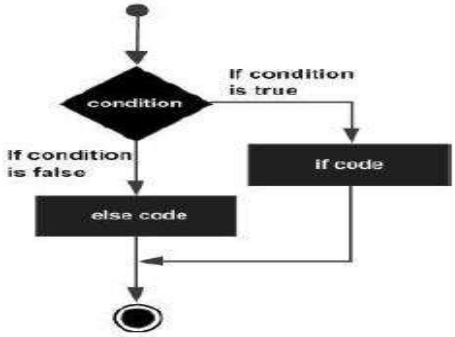
```
# Request input from the user
num = eval(input("Please enter an integer in the range 0...9999: "))
# Attenuate the number if necessary
if num < 0: # Make sure number is not too small
  num = 0
if num > 9999: # Make sure number is not too big
  num = 9999
print(end="[ ") # Print left brace
# Extract and print thousands-place digit
digit = num//1000 # Determine the thousands-place digit
print(digit, end="") # Print the thousands-place digit
num %= 1000 # Discard thousands-place digit
# Extract and print hundreds-place digit
digit = num//100 # Determine the hundreds-place digit
print(digit, end="") # Print the hundreds-place digit
num %= 100 # Discard hundreds-place digit
# Extract and print tens-place digit
digit = num//10 # Determine the tens-place digit
print(digit, end="") # Print the tens-place digit
num %= 10 # Discard tens-place digit
# Remainder is the one-place digit
print(num, end="") # Print the ones-place digit
print("]") # Print right brace
```

ما هى نتيجة البرنامج ؟

جملة if……else

```
if condition :
   if block
else:
   else block
```

فى هذه الحالة اذا كان الشرط TRUE فانه يتم تنفيذ الجمل التى تلى If مباشرة اما اذا كان الشرط غير صحيح False فانه يتم تنفيذ الجمل التى تلى else الشكل التالى يعبر عن حالة if……else



مثال

```
2.py - C:/Users/Dell/AppData/Local/Programs/Python/Python37-32/2...
File  Edit  Format  Run  Options  Window  Help
var1 = 100
if var1:
      print ("1 - Got a true expression value")
      print (var1)
else:
      print ("1 - Got a false expression value")
      print (var1)
var2 = 0
if var2:
     print ("2 - Got a true expression value")
     print (var2)
else:
     print ("2 - Got a false expression value")
     print (var2)
     print ("Good bye!")
                                                    Ln: 16  Col: 0
```

مثال :-

```
#  Get two integers from the user
dividend, divisor = eval(input('Please enter two numbers to divide: '))
#  If possible, divide them and report the result
if divisor != 0:
    print(dividend, '/', divisor, "=", dividend/divisor)
else:
    print('Division by zero is not allowed')
```

النتيجة :-

```
Please enter two integers to divide: 32, 0
Division by zero is not allowed
```

مثال:-

```
d1 = 1.11 - 1.10
d2 = 2.11 - 2.10
print('d1 =', d1, ' d2 =', d2)
if d1 == d2:
    print('Same')
else:
    print('Different')
```

النتيجة :-

```
d1 = 0.010000000000000009  d2 = 0.009999999999999787
```

التعبيرات المركبة :-Compound Boolean Expressions

التعبيرات الشرطية البسيطة تحتوى على معامل علاقة واحدة (relation operator)ويمكن ان تجمع لتكون تعبيرات شرطية مركبة وفقا للجدول التالى :-

e_1	e_2	e_1 and e_2	e_1 or e_2	not e_1
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

مثال :

```
x = 10
y = 20
b = (x == 10)              #  assigns True to b
b = (x != 10)              #  assigns False to b
b = (x == 10 and y == 20)  #  assigns True to b
b = (x != 10 and y == 20)  #  assigns False to b
b = (x == 10 and y != 20)  #  assigns False to b
b = (x != 10 and y != 20)  #  assigns False to b
b = (x == 10 or y == 20)   #  assigns True to b
b = (x != 10 or y == 20)   #  assigns True to b
b = (x == 10 or y != 20)   #  assigns True to b
b = (x != 10 or y != 20)   #  assigns False to b
```

الشروط المتداخلة Nested Conditionals:-

يمكن ان تتداخل جمل if معا مثال

```
value = eval(input("Please enter an integer value in the range 0...10: ")
if value >= 0:          #  First check
    if value <= 10:     #  Second check
        print("In range")
print("Done")
```

مثال :

```
value = eval(input("Please enter an integer value in the range 0...10: ")
if value >= 0:          #  First check
    if value <= 10:     #  Second check
        print(value, "is in range")
    else:
        print(value, "is too large")
else:
    print(value, "is too small")
print("Done")
```

مثال : فى هذا المثال تتداخل جمل If-else

```
value = eval(input("Please enter an integer in the range 0...5: "))
if value < 0:
    print("Too small")
else:
    if value == 0:
        print("zero")
    else:
        if value == 1:
            print("one")
        else:
            if value == 2:
                print("two")
            else:
                if value == 3:
                    print("three")
                else:
                    if value == 4:
                        print("four")
                    else:
                        if value == 5:
                            print("five")
                        else:
                            print("Too large")
print("Done")
```

تمارين (5)

1- بين نتيجة تنفيذ الكود التالى :

```
# i, j, and k are numbers
if  i < j:
    if j < k:
        i = j
    else:
        j = k
else:
    if j > k:
        j = i
    else:
        i = k
print("i =", i, " j =", j, " k =", k)
```

What will the code print if the variables i, j, and k have the following values?

(a) i is 3, j is 5, and k is 7
(b) i is 3, j is 7, and k is 5
(c) i is 5, j is 3, and k is 7
(d) i is 5, j is 7, and k is 3
(e) i is 7, j is 3, and k is 5
(f) i is 7, j is 5, and k is 3

2- اكتب برنامج بايثون يطلب من المستخدم خمسة قيم صحيحة . يقوم البرنامج بطباعة اكبر واصغر قيمة مدخلة اذا قام المستخدم بادخال القيم 3,2,5,0,1 فيقوم بطباعة 5 على انها اكبر قيمة ، 0 على انه اصغر قيمة

جمل التكرار

يكرر التكرار تنفيذ مجموعة من الجمل فى الكود . التتكرار مفيد لحل بعض المشاكل البرمجية . يعتبر كلا من التكرار والشرط اساس تكوين الخوارزميات

من جمل التكرار :-

1- جملة while

البرنامج التالى يقوم بطباعة الارقام من 1 الى 5

```
print(1)
print(2)
print(3)
print(4)
print(5)
```

نتيجة

```
1
2
3
4
5
```

الشكل العام :

```
while condition :
    block
```

يتم تكرار مجموعة الجمل ما دام الشرط الذى يلى while صحيح

مثال :-

```
count = 1              # Initialize counter
while count <= 5:      # Should we continue?
    print(count)       # Display counter, then
    count += 1         # Increment counter
```

مثال :-

```
#Allow the user to enter a sequence of non-negative
# numbers. The user ends the list with a negative
# number. At the end the sum of the non-negative
# numbers entered is displayed. The program prints
# zero if the user provides no non-negative numbers.

entry = 0 # Ensure the loop is entered
sum = 0 # Initialize sum

# Request input from the user
print("Enter numbers to sum, negative number ends list:")

while entry >= 0: # A negative number exits the loop
    entry = eval(input()) # Get the value
    if entry >= 0: # Is number non-negative?
        sum += entry # Only add it if it is non-negative
print("Sum =", sum) # Display the sum
```

النتيجة :-

```
Enter numbers to sum, negative number ends list:
1
2
3
4
5
6
7
-1
Sum = 28
```

التكرار النهائى ، التكرار غير النهائى :

الامثلة التالية توضح التكرار المنتهى

```
n = 1
while n <= 10:
    print(n)
    n += 1
```

النتيجة :- البرنامج السابق يبع الارقام من 1 الى 10

```
1
2
3
4
5
6
7
8
9
10
```

مثال :

```
n = 1
stop = int(input())
while n <= stop:
     print(n)
    n += 1
```

المثال التالى يوضح تكرار غير نهائى

```
done = False            # Enter the loop at least once
while not done:
   entry = eval(input()) # Get value from user
   if entry == 999:      # Did user provide the magic number?
     done = True         # If so, get out
   else:
    print(entry)         # If not, print it and continue
```

2- جملة for

تستخدم جملة for لتكرار جملة او مجموعة جمل عدد من المرات (على مدى قيم محددة)

الشكل العام :

range (*begin, end, step*)

حيث ان

Begin : اول قيمة فى المدى واذا حذفت فان القيمة الافتراصية 0

end: هى اخر قيمة فى المدى ولا تحذف

Change: هى قيمة الزيادة او النقص فاذا حذفت يتم وضع 0 كقيمة افتراضية

يجب ان تكون قيم begin ، end ، step كلها قيم صحيحة فقط

مثال :

```
for n in range(1, 11):
    print(n)
```

مثال

```
for n in range(21, 0, -3):
    print(n, '', end='')
```

النتيجة :

```
21 18 15 12 9 6 3
```

مثال : يقوم البرنامج التالى يوضح جمع الاعداد الصحيحة الاقل من 100

```
#program to calculate the sum of numbers less than  100
sum = 0    # Initialize sum
for i in range(1, 100):
    sum += i
print(sum)
```

الامثلة التالية توضح امثلة مختلفة

```
range(10)! 0,1,2,3,4,5,6,7,8,9
range(1, 10)! 1,2,3,4,5,6,7,8,9
range(1, 10, 2)! 1,3,5,7,9
range(10, 0, -1)! 10,9,8,7,6,5,4,3,2,1
range(10, 0, -2)! 10,8,6,4,2
range(2, 11, 2)! 2,4,6,8,10
range(-5, 5)! −5,−4,−3,−2,−1,0,1,2,3,4
range(1, 2)! 1
range(1, 1)! (empty)
range(1, -1)! (empty)
range(1, -1, -1)! 1,0
range(0)! (empty)
```

3- جمل التكرار المتداخلة Nested Loops

تسمح لغة البايثون باستخدام جملة تكرار بداخل جملة تكرار

الشكل العام :

تداخل جملة for

```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```

تداخل جملة While

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

مثال :

```
# Print a multiplication table to 10 x 10
# Print column heading
print("  1  2  3  4  5  6  7  8  9  10")
print("  +--------------------------------------")
for row in range(1, 11): # 1 <= row <= 10, table has 10 rows
  if row < 10: # Need to add space?
    print(" ", end="")
    print(row, "| ", end="") # Print heading for this row.
  for column in range(1, 11): # Table has 10 columns.
    product = row*column; # Compute product
    if product < 100: # Need to add space?
        print(end=" ")
    if product < 10: # Need to add another space?
        print(end=" ")
        print(product, end=" ") # Display product
  print()
```

النتيجة :-

```
     1   2   3   4   5   6   7   8   9  10
   +--------------------------------------
 1 |   1   2   3   4   5   6   7   8   9  10
 2 |   2   4   6   8  10  12  14  16  18  20
 3 |   3   6   9  12  15  18  21  24  27  30
 4 |   4   8  12  16  20  24  28  32  36  40
 5 |   5  10  15  20  25  30  35  40  45  50
 6 |   6  12  18  24  30  36  42  48  54  60
 7 |   7  14  21  28  35  42  49  56  63  70
 8 |   8  16  24  32  40  48  56  64  72  80
 9 |   9  18  27  36  45  54  63  72  81  90
10 |  10  20  30  40  50  60  70  80  90 100
```

الفصل السادس
الدوال الجاهزة فى لغة البايثون

تحتوى لغة البايثون على العديد من الدوال الجاهزة مثل :-

Mathematical Functions

Python includes following functions that perform mathematical calculations.

1- الدوال الرياضية Mathematical Functions

الجدول التالى يوضح الدوال الرياضية الجاهزة

Function	Returns (description)
abs(x)	The absolute value of x: the (positive) distance between x and zero.
ceil(x)	The ceiling of x: the smallest integer not less than x
cmp(x, y)	-1 if x < y, 0 if x == y, or 1 if x > y
exp(x)	The exponential of x: e^x
fabs(x)	The absolute value of x.
floor(x)	The floor of x: the largest integer not greater than x
log(x)	The natural logarithm of x, for x> 0
log10(x)	The base-10 logarithm of x for x> 0.
max(x1, x2,...)	The largest of its arguments: the value closest to positive infinity
min(x1, x2,...)	The smallest of its arguments: the value closest to negative infinity
modf(x)	The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float.
pow(x, y)	The value of x**y.

round(x [,n])	x rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0.
sqrt(x)	The square root of x for x > 0

الجدول التالى يوضح بعض الدوال داخل حزمة **math**

mathfunctions Module
sqrt Computes the square root of a number: sqrt(x) = $\sqrt{x}$
exp Computes *e* raised a power: exp(x) = e^x
log Computes the natural logarithm of a number: log(x) = $\log_e x = \ln x$
log10 Computes the common logarithm of a number: log(x) = $\log_{10} x$
cos Computes the cosine of a value specified in radians: cos(x) = cos*x*; other trigonometric functions include sine, tangent, arc cosine, arc sine, arc tangent, hyperbolic cosine, hyperbolic sine, and hyperbolic tangent
pow Raises one number to a power of another: pow(x,y) = x^y
degrees Converts a value in radians to degrees: degrees(x) = $\frac{\pi}{180}x$
radians Converts a value in degrees to radians: radians(x) = $\frac{180}{\pi}x$
fabs Computes the absolute value of a number: fabs(x) = $\lvert x \rvert$

مثال : اكتب برنامج لايجاد الجذر التربيعى لاى رقم باستخدام الدوال الجاهزة

```
# To makes the sqrt function available for use in the program
from math import sqrt
# Get value from the user
num = eval(input("Enter number: "))
# Compute the square root
root = sqrt(num);
# Report result
print("Square root of", num, "=", root)
```

النتيجة

```
Enter number: 144
Square root of 144 = 12.0
```

مثال : المثال التالي يوضح العديد من استخدامات الدالة الجاهزة sqrt

```
# This program shows the various ways the
# sqrt function can be used.
from math import sqrt
x = 16
# Pass a literal value and display the result
print(sqrt(16.0))
# Pass a variable and display the result
print(sqrt(x))
# Pass an expression
print(sqrt(2 * x - 5))
# Assign result to variable
y = sqrt(x)
print(y)
# Use result in an expression
y = 2 * sqrt(x + 16) - 4
print(y)
# Use result as argument to a function call
y = sqrt(sqrt(256.0))
print(y)
print(sqrt(int('45')))
```

النتيجة:-

```
4.0
4.0
5.196152422706632
4.0
7.313708498984761
4.0
6.708203932499369
```

2-الدوال المثلثية Trigonometric Function:

تحتوى البايثون على الدوال المثلثية التالية

Function	Description
acos(x)	Return the arc cosine of x, in radians.
asin(x)	Return the arc sine of x, in radians.
atan(x)	Return the arc tangent of x, in radians.
atan2(y, x)	Return atan(y / x), in radians.
cos(x)	Return the cosine of x radians.
hypot(x, y)	Return the Euclidean norm, sqrt(x*x + y*y).
sin(x)	Return the sine of x radians.
tan(x)	Return the tangent of x radians.
degrees(x)	Converts angle x from radians to degrees.
radians(x)	Converts angle x from degrees to radians.

الثوابت الرياضية Mathematical Constant

Constants	Description
pi	The mathematical constant pi.
e	The mathematical constant e.

كتابة دالة

Writing Function

تسمح لنا لغة البايثون بتكوين دالة

الشكل العام :-

```
def  name ( parameter_list ) :
    block
```

حيث ان:

def كلمة محجوزة تبين بداية تعريف الدالة

Name هى اسم الدالة

parameter_list هى مجموعة من المعاملات التى تستخدمها الدالة مفصولة بفاصلة

block هى مجموعة الجمل التى تنفذها الدالة (body of function)

مثال: الكود التالى يوضح تعريف دالة لايجاد greatest common factor

```
def gcd(num1, num2):
    #  Determine the smaller of num1 and num2
    min = num1 if num1 < num2 else num2
    #  1 is definitely a common factor to all ints
    largestFactor = 1
    for i in range(1, min + 1):
        if num1 % i == 0 and num2 % i == 0:
            largestFactor = i   # Found larger factor
    return largestFactor
```

الدالة الرئيسية Main Function

الدوال تساعد فى تنيم البرنامج . يوجد فى لغة البايثون دالة main هى التى يبدا من عندها تنفيذ البرنامج المثال التالى يوضح ذلك

```
#  Computes the greatest common divisor of m and n
def gcd(m, n):
    #  Determine the smaller of m and n
    min = m if m < n else n
    #  1 is definitely a common factor to all ints
    largestFactor = 1
    for i in range(1, min + 1):
        if m % i == 0 and n % i == 0:
            largestFactor = i   # Found larger factor
    return largestFactor
```

```
#  Get an integer from the user
def get_int():
    return int(input("Please enter an integer: "))

#  Main code to execute
def main():
    n1 = get_int()
    n2 = get_int()
    print("gcd(", n1, ",",  n2, ") = ", gcd(n1, n2), sep="")

# Run the program
main()
```

مثال :

```
def increment(x):
   print("Beginning execution of increment, x =", x)
   x += 1   #  Increment x
   print("Ending execution of increment, x =", x)

def main():
    x = 5
    print("Before increment, x =", x)
    increment(x)
    print("After increment, x =", x)

main()
```

النتيجة :

```
Before increment, x = 5
Beginning execution of increment, x = 5
Ending execution of increment, x = 6
After increment, x = 5
```

تمارين(6)

1. Is the following a legal Python program?

```
def proc(x):
    return x + 2

def proc(n):
    return 2*n + 1

def main():
    x = proc(5)

main()
```

2. Is the following a legal Python program?

```
def proc(x):
    return x + 2

def main():
    x = proc(5)
    y = proc(4)

main()
```

3. Is the following a legal Python program?

```
def proc(x):
    print(x + 2)

def main():
    x = proc(5)

main()
```

4. Is the following a legal Python program?

```
def proc(x):
    print(x + 2)

def main():
    proc(5)

main()
```

5. Is the following a legal Python program?

```
def proc(x, y):
    return 2*x + y*y

def main():
    print(proc(5, 4))

main()
```

6. Is the following a legal Python program?

```
def proc(x, y):
    return 2*x + y*y

def main():
    print(proc(5))

main()
```

7. Is the following a legal Python program?

```
def proc(x):
    return 2*x

def main():
    print(proc(5, 4))

main()
```

8. Is the following a legal Python program?

```
def proc(x):
    print(2*x*x)


def main():
    proc(5)

main()
```

9. The programmer was expecting the following program to print 200. What does it print instead? Why does it print what it does?

```
def proc(x):
    x = 2*x*x

def main():
    num = 10
    proc(num)
    print(num)

main()
```

10. Is the following program legal since the variable `x` is used in two different places (`proc` and `main`)? Why or why not?

```
def proc(x):
    return 2*x*x

def main():
    x = 10
    print(proc(x))

main()
```

11. Is the following program legal since the actual parameter has a different name from the formal parameter (`y` vs. `x`)? Why or why not?

```
def proc(x):
    return 2*x*x

def main():
    y = 10
    print(proc(y))

main()
```

12. Complete the following `distance` function that computes the distance between two geometric points (x_1, y_1) and (x_2, y_2):

```
def distance(x1, y1, x2, y2):
    ...
```

Test it with several points to convince yourself that is correct.

الفصل السابع
تطبيقات عملية على لغة البايثون

مثال(1) اكتب برنامج لحساب مجموع وحاصل ضرب ومتوسط ثلاثة متغيرات A,B, C

```
#program to calculate the summation , average and
#product of three  values A, B, C
A=eval(input('enter the value of A'))
B=eval(input('enter the value of B'))
C=eval(input('enter the value of C'))
S=A+B+C
Avr=S/3
pro=A*B*C
print(S)
print(Avr)
print(pro)
```

النتيجة

```
enter the value of A10
enter the value of B20
enter the value of C30
60
20.0
6000
```

مثال(2): اكتب برنامج يقوم بتحويل درجة الحرارة من فهر نهيت (F) إلى سيليزية (C). البرنامج يقرأ درجة الحرارة الفهرنهيتية ويقوم بطباعة درجة الحرارة بالسنتغراد. قاعدة التحويل من درجة الحرارة الفهرنهيتية إلى درجة الحرارة السنتغراد تعطى بالمعادلة التالية:

$$C = (F - 32) * \frac{5}{9}$$

```
F=eval(input('enter the value of F'))
C=(F-32)*(5/9)
print(C)
```

النتيجة :-

```
enter the value of F100
37.77777777777778
```

مثال(3) اكتب برنامج لطباعة الارقام من 1 الى 5

```
for n in range(1,6):
        print (n)
print(' n contains: ',n)
```

النتيجة :-

```
1
2
3
4
5
n contains: 5
```

مثال (4)

اكتب برنامج لايجاد مضروب اى عدد N

```
#python program to find the factorial of a number provided by the user.
#take  input  from the user
num=eval(input("Enter a number :"))
factorial =1
#check if the number is negative , positive or zero
if num<0:
    print("sorry, factorial does not exist for negative numbers")
elif num== 0:
    print("The factorial of 0 is 1")
else:
      for  i in range(1,num+1):
           factorial=factorial*i

      print("The factorial of ", num, "is", factorial)
```

النتيجة

```
Enter a number :5
The factorial of  5 is 120
```

مثال (5) اكتب برنامج لعمل الة حاسبة بسيطة تقوم بعمليات الجمع ، الطرح

```
# help_screen
# Displays information about how the program works
# Accepts no parameters
# Returns nothing
def help_screen():
    print("Add: Adds two numbers")
    print("Subtract: Subtracts two numbers")
    print("Print: Displays the result of the latest operation")
    print("Help: Displays this help screen")
    print("Quit: Exits the program")
# menu
# Display a menu
# Accepts no parameters
# Returns the string entered by the user.
def menu():
# Display a menu
   return input("=== A)dd S)ubtract P)rint H)elp Q)uit ===")
```

```
# main
# Runs a command loop that allows users to
# perform simple arithmetic.
def main():
   result = 0.0
   done = False; # Initially not done
   while not done:
      choice = menu() # Get user's choice
      if choice == "A" or choice == "a": # Addition
        arg1 = float(input("Enter arg 1: "))
        arg2 = float(input("Enter arg 2: "))
        result = arg1 + arg2
        print(result)
      elif choice == "S" or choice == "s": # Subtraction
       arg1 = float(input("Enter arg 1: "))
       arg2 = float(input("Enter arg 2: "))
       result = arg1 - arg2
       print(result)
      elif choice == "P" or choice == "p": # Print
        print(result)
      elif choice == "H" or choice == "h": # Help
         help_screen()
      elif choice == "Q" or choice == "q": # Quit
          done = True
main()
```

النتيجة :-

```
=== A)dd S)ubtract P)rint H)elp Q)uit ===A
Enter arg 1: 2
Enter arg 2: 3
5.0
=== A)dd S)ubtract P)rint H)elp Q)uit ===s
Enter arg 1: 3
Enter arg 2: 2
1.0
=== A)dd S)ubtract P)rint H)elp Q)uit ===p
1.0
=== A)dd S)ubtract P)rint H)elp Q)uit ===H
Add: Adds two numbers
Subtract: Subtracts two numbers
Print: Displays the result of the latest operation
Help: Displays this help screen
Quit: Exits the program
=== A)dd S)ubtract P)rint H)elp Q)uit ===Q
```

مثال (6) اكتب برنامج لاختبار اذا كان الرقم N يكون Prime

```
#python program to check if the input number is prime or not
#take input from the user
num=eval(input("Enter a number :"))

#prime numbers are greater  than 1
if num>1 :
    #check for factors
  for i in range (2,num):
      if (num % i)==0:
          print(num, "is not a prime number")
          print(i, "times",num //i, "is ", num)
          break
      else :
          print(num, "is a prime number")
#if input number is less than
#or equal to 1, it is not prime
else:
     print(num, "is not a prime ")
```

النتيجة

```
Enter a number:2
2 is a prime
```

تمرين : اشرح البرنامج التالى

```
from math import sqrt

# is_prime(n)
#    Determines the primality of a given value
#    n an integer to test for primality
#    Returns true if n is prime; otherwise, returns false
def is_prime(n):
    result = True  #  Provisionally, n is prime
    root = sqrt(n)
    #  Try all potential factors from 2 to the square root of n
    trial_factor = 2
    while result and trial_factor <= root:
        result = (n % trial_factor != 0 )  #  Is it a factor?
        trial_factor += 1                  #  Try next candidate
    return result

# main
#    Tests for primality each integer from 2
#    up to a value provided by the user.
#    If an integer is prime, it prints it;
#    otherwise, the number is not printed.
def main():
    max_value = int(input("Display primes up to what value? "))
    for value in range(2, max_value + 1):
        if is_prime(value):          #  See if value is prime
            print(value, end=" ")    #  Display the prime number
    print()  #  Move cursor down to next line

main()   #  Run the program
```

تمرين : اشرح البرنامج التالى ووضح مخرجاته

```
#    help_screen
#       Displays information about how the program works
#       Accepts no parameters
#       Returns nothing
def help_screen():
    print("Add:  Adds two numbers")
```

```
    print("Subtract:  Subtracts two numbers")
    print("Print:  Displays the result of the latest operation")
    print("Help:  Displays this help screen")
    print("Quit:  Exits the program")

! menu
!     Display a menu
!     Accepts no parameters
!     Returns the string entered by the user.
def menu():
    #  Display a menu
    return input("=== A)dd S)ubtract P)rint H)elp Q)uit ===")


!    main
!       Runs a command loop that allows users to
!       perform simple arithmetic.
def main():
    result = 0.0
    done = False;  #  Initially not done
    while not done:
        choice = menu()      # Get user's choice

        if choice == "A" or choice == "a":   # Addition
            arg1 = float(input("Enter arg 1: "))
            arg2 = float(input("Enter arg 2: "))
            result = arg1 + arg2
            print(result)
        elif choice == "S" or choice == "s": # Subtraction
            arg1 = float(input("Enter arg 1: "))
            arg2 = float(input("Enter arg 2: "))
            result = arg1 - arg2
            print(result)
        elif choice == "P" or choice == "p": # Print
            print(result)
        elif choice == "H" or choice == "h": # Help
            help_screen()
         elif choice == "Q" or choice == "q": # Quit
             done = True

main()
```

تمرين : ما هى مخرجات البرنامج التالى

```
#  tree(height)
#     Draws a tree of a given height
#     height is the height of the displayed tree
def tree(height):
    row = 0              #  First row, from the top, to draw
    while row < height:  # Draw one row for every unit of height
        #  Print leading spaces
        count = 0
        while count < height - row:
            print(end=" ")
            count += 1
        #  Print out stars, twice the current row plus one:
        #    1. number of stars on left side of tree
        #       = current row value
        #    2. exactly one star in the center of tree
        #    3. number of stars on right side of tree
        #       = current row value
        count = 0
        while count < 2*row + 1:
            print(end="*")
            count += 1
        #  Move cursor down to next line
        print()
        #  Change to the next row
        row += 1

#  main
#     Allows users to draw trees of various heights
def main():
    height = int(input("Enter height of tree: "))
    tree(height)

main()
```

المراجع:

- Mark Lutz, “Programming Python: Powerful Object-Oriented Programming”, 4th Edition, O'Reilly Media; 2011.
- Mike McGrath, “Python in Easy Steps”, In Easy Steps Ltd (February 15, 2015).
- David Beazley and Brian K. Jones, “Python Cookbook: Recipes for Mastering Python 3”, 3rd Edition, O'Reilly Media; 2013.
- Luciano Ramalho, “Fluent Python: Clear, Concise, and Effective Programming”, 1st Edition, O'Reilly Media; 2015.
- Allen Downey, Jeff Elkner and Chris Meyers, “Learning with Python How to Think Like a Computer Scientist”, 1st Edition, Green Tea Press, Wellesley, Massachusetts; 2002.
- https://docs.python.org/3/
- http://tutorialspoint.com/python
- https://www.w3schools.com/python

- إلبرمجة بلغة بايثون ,أكاديميّة حسوب , **Online: shorturl.at/ioqG3**
- أبو الوليد بن رشد القرطبي "احترف البايثون الآن Learn python now" shorturl.at/gCY15,