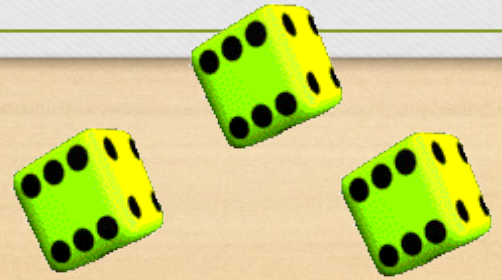
تقنية حلول المشاكل

ITGS113

مقدمة مختصرة حول البرمجة بلغة البايثون

إعداد : أ. منار سامي عريف

المحاضرة الثانية



البرنامج:

هو مجموعة من التعليمات instructions أو الأوامر commands أو الجمل statements المكتوبة بإحدى لغات البرمجة، ويتم تنفيذها من قبل الحاسوب لتنفيذ مهمة ما. كحل حل معادلة رياضية، أو معالجة بيانات، أو . . . وغيره.

لغة البرمجة:

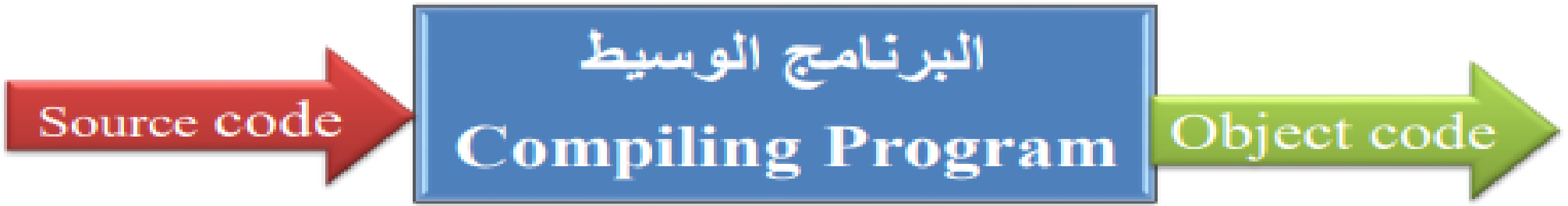
تتكون أي لغة من عنصرين أساسيين هما الكلمات words والقواعد grammars، هذه القواعد تُستخدم لترتيب وتنسيق الكلمات مع بعضها البعض في صورة منطقية لتُعطي معنى، وهذا ينطبق على لغات الحاسب التي بدورها تتكون من مجموعة من التعلميات والأوامر والتي هي مزيج من الكلمات المحجوزة reserved words المرتبة بصورة منطقية وفقًا لمجموعة من القواعد الخاصة بها.

لغات البرمجة: هي اللغات المُستخدمة لكتابة التعليمات والاوامر الموجهة للحاسب لتنفيذ عملية ما أو أداء مهمة محددة. وهناك العديد من لغات البرمجة، حيث تتنوع حسب الأسلوب أو النهج البرمجي المتبع فيها.

المبرمج Programmer

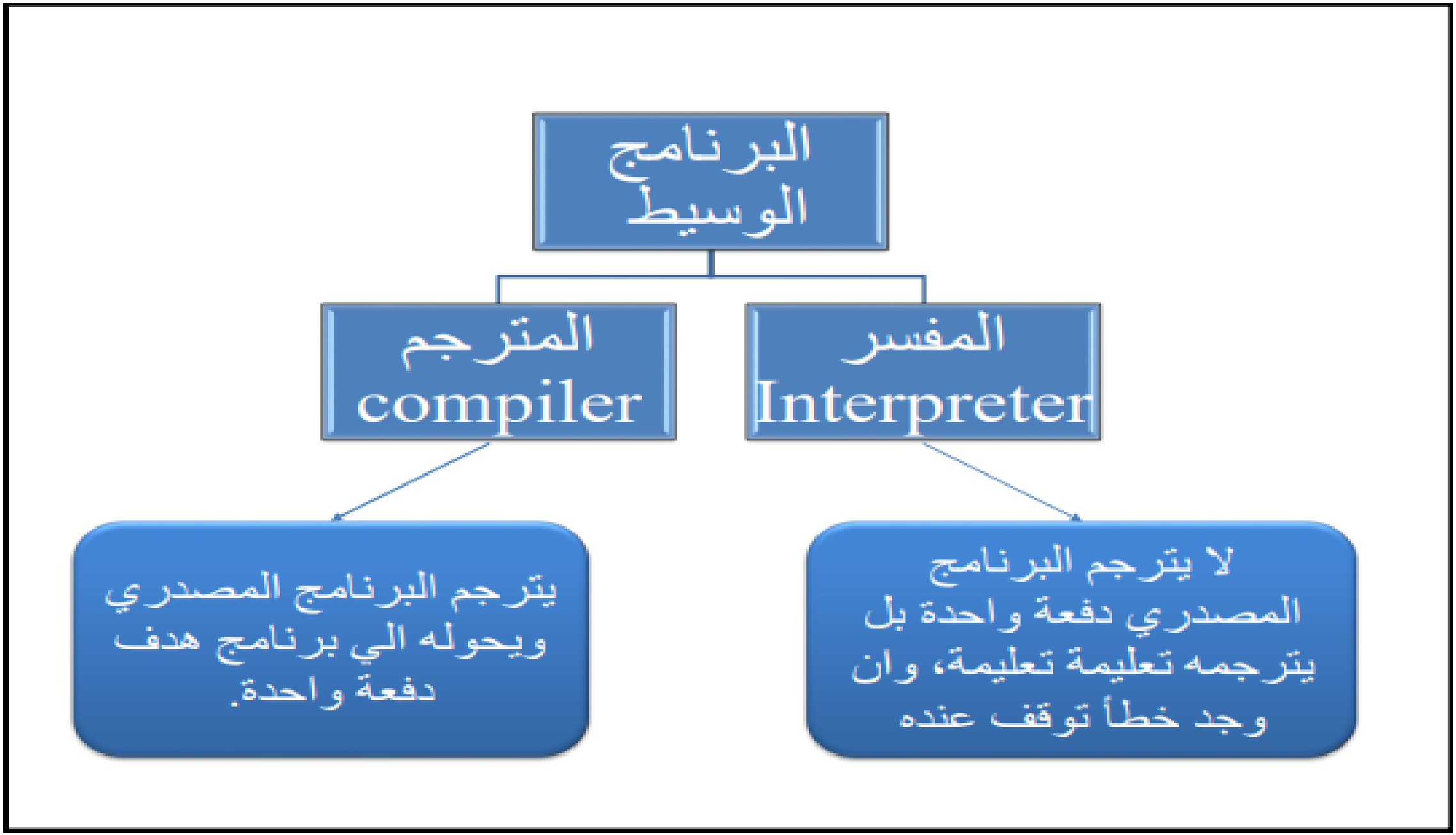
- الشخص الذي يُعد البرنامج و يكتبه بطريقة مناسبة، و ذلك من أجل حل مشكلة ما حاسوبياً.
- يقوم المبرمج بكتابة البرنامج في بيئة برمجة مناسبة IDE و هي اختصار للكلمات integrated development environment و معناها بيئة تطوير متكاملة مثل NetBeans.
- يتم كتابة البرنامج في مكان يُسمى بمحرر النصوص editor، ثم ترجمته من لغة البرمجة المكتوب بها إلى لغة الآلة باستخدام المترجم Compiler وتنفيذه run على البيئة المناسبة.

في اللغات عالية المستوى



البرنامج المصدري: هو عبارة عن مجموعة من الأوامر والتعليمات المكتوبة بشكل منطقي و متسلسل بإحدى لغات البرمجة عالية المستوى

البرنامج الهدف: هو البرنامج الخالي من الأخطاء ومكتوب بلغة الآلة والقابل للتنفيذ على الحاسوب


البرنامج الوسيط
المترجم compiler
المفسر Interpreter
يترجم البرنامج المصدري ويحوله الي برنامج هدف دفعة واحدة.
لا يترجم البرنامج المصدري دفعة واحدة بل يترجمه تعليمة تعليمة، وان وجد خطأ توقف عنده

مكونات البرنامج

جمل الادخال (Input) : البيانات التي يتم استقبالها من المستخدم

جمل الاخراج (Output) : عرض النتائج والمعلومات

جمل معالجة : رياضية، علائقية، منطقية، نصية.

جمل شرطية : وهي جمل للتحكم في مسار تنفيذ الجمل بناء على شروط معينة.

جمل التكرار : وتختص بتنفيذ الجمل أكثر من مرة إلى أن يتحقق شرط معين أو التنفيذ أكثر من مرة طالما تحقق شرط معين.

لتبسيط الامر وتنفيذ الخوارزميات على جهاز الحاسوب، تحول إلى برنامج بإحدى لغات البرمجة عالية المستوى. وفي هذا المقرر تم اختيار لغة بايثون python لبساطتها وسهولة تعلمها بالإضافة لكونها:

- سهلة الاستعمال لبساطة الجمل والتراكيب بها.
- تعتمد على التنسيق والمحادة في الجمل الشرطية، الحلقات، والوظائف.
- لغة مفسرة (Interpreted)، أي لها القدرة على تجربة جمل لوحدها دون الحاجة لكتابة وتخزين وترجمة برنامج كامل؛ مما يقلل من الجهد والوقت اللازم لتطوير البرنامج.
- مفتوحة المصدر (open source) مما يسمح للعديد من المهتمين بإبداً الملاحظات والإسهامات لتطوير وتحسين اللغة.
- متوافقة مع عدة أنظمة تشغيل مختلفة مما يجعل نقل وتبادل البرامج بين أنظمة التشغيل المختلفة سهلا ولا يحتاج إلى تغيير يُذكر.
- لها جمل لمعالجة الأخطاء الإستثنائية (Exception Handling).
- تدعم الـ(Object Oriented Programming).
- تدعم الاسلوب البنائي (Structured Approach) (مع عدم وجود ***GOTO*** التي تجعل البرنامج صعب التتبع).

www.python.org

أساسيات لغة البايثون:

أولا: الكلمات المحجوزة:

لغة Python كمعظم لغات البرمجة، لها عدد من الكلمات المحجوزة، والتي لا يُسمح باستخدامها كأسماء معرفات (متغيرات أو دوال). هذه الكلمات مخزنة في اللغة **ويجب أن تُكتب بطريقة صحيحة وبنفس حالة الأحرف؛ حتى يُمكن التعرف عليها.** في حال استعمالها كمتغيرات أو كتابتها بأحرف حالتها مغايرة لحالة الحروف المخزنة بها ينتج خطأ لغوي. والكلمات هي:

class	finally	is	return	continue	if
None	True	False	for	lambda	try
def	from	not	while	nonlocal	and
del	global	with	elif	yield	or
assert	else	in	import	except	break
pass	raise	as			

لاحظ أن:

معظم الكلمات المجوزة مكتوب بحروف صغيرة إلا أن القليل منها يبدأ بحرف كبير

أمثلة:

- if = 10 ← لا يجوز استخدام if كاسم متغير وسينتج خطأ لغوي.

- true = 10 ← لا ينتج عنها خطأ لغوي فـ true ليست كلمة محجوزة.
- أما True = 10 ← سينتج عنها خطأ لغوي. لأن True كلمة محجوزة

لاحظ true و True كلاهما نفس الكلمة ولكن حالة الحرف الأول مختلفة ولغة بايثون Python حساسة لحالة الحرف فهي تميز بين الحرف الصغير والحرف الكبير، وبالتالي فإن المتغير t لا يكافئ المتغير T

أي أن true ≠ True

لذلك يجب أن تُكتب الكلمات المحجوزة بطريقة صحيحة وبنفس حالة الأحرف؛ حتى يُمكن التعرف عليها.

القيم Values : تنقسم القيم بلغة Python إلى الأنواع التالية :

- قيم رقمية صحيحة (Integer) مثل 502، 140-، ... أي اعداد بدون الفاصلة العشرية.
- قيمة رقمية كسرية (Float) مثل 0.134 ,.45- ,3.145 ،... أي اعداد بالفاصلة العشرية.
- قيم رقمية تخيلية مثل 45j- ,5j إي بإضافة الحرف اللاتيني **j** يمين العدد.
- قيم نصية مثل "برمجة"، "Programming"، "لغات برمجة"، "113"،... أي مجموعة من الرموز محاطة بعلمتي التنصيص المفردة (') أو المزدوجة (").
- قيم منطقية وتشمل القيمتين True, False فقط. لاحظ شكل الحرف الأول في الكلمتين.

المتغيرات Variables: المتغير هو اسم يشير إلى عنوان لتخزين قيمة من القيم المذكورة أعلاه.

- يتكون إسم المتغير من الحروف اللاتينية والأرقام 9،8،7،6،5،4،3،2،1،0 والرمز (_) على أن يبدأ بحرف.
- عادة تستعمل الرمز (_) لربط كلمتين أو أكثر.
- يمكننا تخيل المتغير على أنه مكان (صندوق) توضع فيه قيمة معينة وقابلة للتغير.

ملاحظات يجب أخذها بعين الاعتبار :

- ضرورة التمييز بين الشرطة السفلية (_) وإشارة الناقص (-) فالشرطة السفلية مسموح بها ضمن اسم المتغير أما الناقص فلا.
- يُفضل عند اختيار اسم متغير أن يكون دالا على محتواه، لتسهيل فهم وتتبع البرنامج بكل بساطة فمثلا :
  - نستخدم المتغير name ليدل على أن ما سيُخزن فيه هو الاسم.
  - نستخدم المتغير phone_no ليدل على أن ما سيُخزن فيه هو رقم الهاتف.
  - نستخدم المتغير course_no ليدل على أن ما سيُخزن فيه هو رمز المقرر.

أمثلة لأسماء جائزة كأسماء متغيرات:

birth_date, myname, itgs113, phone_no, student_id

أمثلة لأسماء لايجوز استخدامها كأسماء متغيرات:

birth-date, birth date, 5A, student.id, it+gs

- لاحظ استخدام الأحرف الصغيرة في أسماء المتغيرات بدلا من الأحرف الكبيرة، وذلك تمييزا لأسماء المتغيرات من أسماء الثوابت. هذه القاعدة ليست ملزمة ولكن مُتفق عليها في كتابة البرامج. ونحن سنلتزم بها.
- لغة Python حساسة لحالة الحرف فهي تميز بين الحرف الصغير والحرف الكبير، وبالتالي فإن المتغير ab لا يكافئ المتغير AB

المخرجات والمدخلات
Inputs and Outputs:

المخرجاتOutputs :

- **لعرض نص** على الشاشة: نستخدم علامة التنصيص المزدوجة. او المفردة. مثلا:

استخدام علامة التنصيص المزدوجة.	***print*** ("هذا مثال")
استخدام علامة التنصيص المفردة.	***print*** (' هذا مثال)
عند تنفيذ هذه الجملة يظهر على الشاشة النص التالي: **هذا مثال**	

مثال :اكتب وظيفة تقوم بعرض الجملة Hello world!
لتنفيذ البرنامج يتم كتابته كما في المثال: باستخدم الأمر ***print*** لعرض النص الموجود بين علامات التنصيص " " على الشاشة.

```
print ("Hello, World!")  ← البرنامج
----------
Hello, World!  ← ناتج البرنامج
```

المخرجات Outputs :

- **لعرض قيمة (قيم) عددية** على الشاشة: أيضا نستخدم الجملة ***print*** كالآتي:

print(year)	استخدام ***print*** لعرض قيمة عددية

مثال :اكتب برنامج لعرض قيمتي المتغيرين x و y إذا كانت:

$$x = 750 \text{ و } y = 3.14$$

```
print(x,y)
```

← البرنامج

```
750 3.14
```

← ناتج البرنامج

المدخلات Inputs :

كيف يتم إسناد قيمة لمتغير ؟

أولا : عن طريق إدخال القيمة من لوحة المفاتيح، باستخدام الوظيفة

input (“prompt")

عند التنفيذ تقوم الوظيفة ***input*** بعرض النص الموجود بين علامتي التنصيص " " على الشاشة (إن وجد) لمساعدة المُدخل على معرفة المطلوب إدخاله كما في المثال التالي:

student_name = ***input*** ("Enter your name : ")

هذه رسالة للمستخدم لإدخال اسمه. حيث أن في هذه الحالة الكلمةprompt أخذت القيمة النصية: Enter your name. الجملة ***input***('prompot') تقراء القيمة المدخلة من لوحة المفاتيح كقيمة نصية وإسنادها إلى المتغير المكتوب معها في نفس السطر كالأتي:

student_number = ***input***('enter_number:')

مثلا لو أن المستخدم ادخل القيمة 12345 من لوحة المفاتيح فإن الجملة ***input*** ستسند هذه القيمة إلى المتغير student_number كقيمة نصية أي أن هذا المتغير سيحتوي على النص التالي'12345' وليس الرقم 12345

ملاحظات:

- من ضمن جملة النص بجملة ***print*** يوجد الرمز \n ويعني الانتقال إلى سطر جديد.
- تنسيق الجمل بداية من اليسار من عمود واحد وفي حالة الإخلال بالقاعدة يحدث خطأ لغوي (Syntax Error).
- الخط العريض الداكن هو النص الموجود في جملة ***input***
- والبيانات ذات الخط غير الداكن تم إدخالها من قبل المستخدم
- مابين علامات التنصيص المزدوجة الثلاثية (""" ..... """) يعتبر جمل توضحية لقارئ البرنامج وليس جمل يقوم الحاسوب بتنفذها وبالإمكان استخدام اللغة العربية كما هو موضح بالمثال.
- كما يمكن استخدام الرمز # في حالة أن الجملة التوضيحية لا تتجاوز سطر

This comment is written in a single line

ثانيا : إسناد قيم للمتغيرات باستخدام جملة التعيين (Assignment statement)

- مثل جملة لإسناد القيمة النصية "programming language" للمتغير course_name نكتب الجملة التالية

```
course_name = 'programming language'
```

- مثلا لإسناد القيمة الرقمية 2023 إلى المتغير year نكتب الجملة التالية

```
year  = 2023
```

تنبيه:

ما نوع القيمة التي تم تخزينها بالجملة التالية؟

```
student_id = "123456789"
```

تم تخزينها كقيمة نصية ولا يمكن استخدامها في جمل حسابية إلا بعد تحويلها إلى قيمة رقمية من النوع الصحيح ويتم ذلك باستخدام الوظيفة ***int*** (student_id) وإعادة إسنادها إلىنفس المتغير أو متغير آخر.

```
Student_id =int(student_id)
```

العمليات التي تجرى على البيانات:

1) **المؤثرات الحسابية:** رموز خاصة تمثل العمليات الحسابية كالجمع، والطرح والضرب، والقسمة وغيرها ...).

المؤثر	العملية الحسابية
+	الجمع addition
-	الطرح subtraction
*	الضرب multiplication
/	القسمة division
//	القسمة مع حذف الجزء الكسري من الناتج ان وجد
%	باقي القسمة remainder
**	الأس exponentiation

اسبقيه تنفيذ المؤثرات الحسابية في التعابير الحسابية

المؤثر	التنفيذ في حالة تساوي الأولوية
( )	من اليسار إلى اليمين
**	من اليمين إلى اليسار
اشارة العدد -	من اليسار إلى اليمين
* , / , // , %	من اليسار إلى اليمين
+ , -	من اليسار إلى اليمين

العمليات التي تجرى على البيانات:

2) المؤثرات العلائقية: رموز خاصة تمثل العمليات العلائقية من مقارنة (أكبر من، أصغر من، وغيرها .....).

الرمز	العملية العلائقية
==	يساوي
!=	لا يساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي

3) المؤثرات المنطقية: رموز خاصة تمثل العمليات المنطقية

المؤثر	ناتج العملية المنطقية
and	إذا كانت قيمتي طرفي المؤثرTrue ← الناتج سيكون True ، عدا ذلك سيكون False
or	إذا كانت قيمتي طرفي المؤثرFalse ← الناتج سيكون False، عدا ذلك سيكون True
not	الناتج عكس المُدخل : إذا True ← False والعكس إذا False ← True

- **التعابير الرياضية**

جمل تتكون من خليط من القيم والمتغيرات والمؤثرات (حسابية وعلائقية ومنطقية) بصيغة معينة للحصول على قيمة معينة.
مثلا من التعابير الحسابية جمع مقدران رقميان أو مجموعة مقادير تحتوي على العديد من العمليات الحسابية.

فيما يلي توضيح لكيفية تمثيل تعبير رياضي بلغة بايثون.

التعبير الرياضي	التعبير بلغة Python
$Q + P$	Q + P
$Q - P$	Q – P
$Q \times P$	Q * P
$\frac{Q}{P}$	Q / P
Q^P	Q ** P
$\sqrt{Q}$	Q ** 0.5

- أمثلة لتحويل تعابير جبرية إلى تعابير مكافئة بلغة بايثون.

Algebra: $m = \frac{a + b + c + d + e}{5}$

Python:
```
m = ( a + b + c + d + e ) / 5
```

Algebra: $y = mx + b$

Python:
```
y = m * x + b
```

$$area = \sqrt[2]{s * (s - a) * (s - b) * (s - c)}$$

```
area = ( s * (s – a) * ( s – b) * (s – c)) ** 0.5
```

التعابير الرياضية (إيجاد قيمة التعبير الرياضي)

أمثلة لحساب النتيجة النهائية لتعابير حسابية :

التعبير الحسابي	النتيجة
5 * 6 – 24	6
(12 + 23) / 5	7.0
6 / 4	1.5
6. / 4.	1.5
6 // 4	1
6. // 4.	1.
2 ** 2 ** 3	256
-3 ** 2	-9
-(3 ** 2)	-9
(-3) ** 2	9
2.75 % 0.5	0.25

4) المؤثرات النصية: وتشمل " + و * "

نلاحظ أن المؤثران + و * (الجمع والضرب) يستخدمان للتعامل مع القيم النصية، إلا أنهما لن يقوما بعمليتي الجمع والضرب. بل + لربط قيمتين نصيتين معا، أما * فلتكرار قيمة نصية عدد محدد من المرات كما يلي:

المؤثر	مثال	الناتج
+ لربط قيمتين نصيتين	'Hello, ' + 'World!'	'Hello, World!'
* لتكرار قيمة نصية عدد من المرات	'Hello, ' * 3	'Hello, Hello, Hello'
	3 * 'Hello, '	'Hello, Hello, Hello'

المؤثرات النصية: وتشمل " + و * "

```
fname = input ('Enter your first name : ')
lname = input ('Enter your last name : ')
full_name = fname + ' ' + lname
print ('Hello, ' , full_name)
```

← البرنامج

Enter your first name : Omar
Enter in your last name : Osama
Hello Omar Osama

← ناتج البرنامج

```
print ('ITGS113' * 5)
```

← البرنامج

ITGS113ITGS113ITGS113ITGS113ITGS113

← ناتج البرنامج