المحور الأول: مبادئ البرمجة وأساسياتها

مدخل إلى البرمجة:

1. فهم أهمية التفكير الحاسوبي في العلوم السياسية والاجتماعية

عرفت الرابطة الأمريكية لمعلمي علوم الحاسوب CSTA، بالتعاون مع الجمعية الدولية للتقنية في التعليم ISTE التفكير الحاسوبي على أنه: "عملية حل المشكلات، وتتضمن هذه العملية عددا من العناصر التي من خلالها يتم صياغة المشكلة بطريقة تمكن من استخدام الحاسوب للمساعدة على حلها وتعميم الفائدة من عملية حل المشكلة، وتطبيق الحل على مجموعة أكبر من المشكلات".

كما عرف التفكير الحاسوبي: بأنه عمليات عقلية متمثلة في التصميم الخوارزمي، والتحليل، والتفكير المنطقي، المتضمنة في صياغة المشكلات والتوصل لحلول لها، بشكل قابل للتمثيل والتطبيق، بواسطة الحاسوب بكل كفاءة وفاعلية.

تتمثل مهارات التفكير الحاسوبي فيما يلي:

- **التجريد Abstraction:** الهدف من التجريد هو التبسيط، ويتطلب تحديد أبعاد المشكلة وإخفاء باقي التفاصيل، ويُعرف التجريد على أنه عملية تكوين شيء ما يتسم بالبساطة من شيء آخر معقد وذلك من خلال عزل أو إقصاء التفاصيل غير الوثيقة الصلة، فعلى سبيل المثال: تعد الخوارزمية تجريداً لعملية تتضمن مدخلات، وتنفيذ سلسلة من الخطوات، والوصول إلى نواتج أو مخرجات تعمل على تحقيق هدف منشود، وتعد مهارة التجريد بمثابة عملية التفكير الأكثر أهمية والأعلى مستوى في التفكير الحاسوبي.
- **التحليل Decomposition:** التحليل هو طريقة للتفكير بشأن الأجزاء المكونة للمشكلات، وهو ما يساعد الفرد على فهم ما تتضمنه من أجزاء ومكونات، وحلها، وتطويرها وتقويمها كل على حده، كما يجعل المشكلات المعقدة أسهل في الحل ويتضمن التحليل قدرة الفرد على تحديد الجوانب الهامة للمشكلة الحاسوبية والتركيز عليها، والقدرة على تقسيم المشكلة إلى مشكلات فرعية، والقدرة على تحديد العمليات، التي يمكن استخدامها في حل المشكلة، والتكامل بين هذه العمليات لتصميم الخوارزميات، أي أنه يُركز على مهارة تجزئة المشكلات إلى أجزاء صغيرة، بحيث يُمكن التعامل معها والتركيز على حل كل جزء من أجزاء المشكلات، فيمكننا تجزئة مشكلة معقدة إلى أجزاء أصغر وأسهل كي نبني حلاً للمشكلة الكبيرة التي بدأنا بها، مما يساعد على جعل المشكلات أقل تعقيداً.
- **التصميم الخوارزمي Algorithmic Design:** أهم مهارات التفكير الحاسوبي، ويُعرف بأنه طريقة للوصول إلى حل لمشكلة ما، من خلال التحديد الواضح للخطوات اللازمة وذلك يكون بتحديد المدخلات والعمليات والمخرجات، حيث تتضمن كتابة تعليمات محددة وواضحة مرتبة خطوة بخطوة لتنفيذ عملية ما.

❖ **التقييم :Evaluation**التقييم يعني تحديد الحلول الممكنة لمشـــكلة ما وتقرير أي هذه الحلول هو الأفضـــل، وأيها يمكن أن يكون مفيداً في بعض الحالات وليس كلها وكيف يمكن تحســـينها؟ وعندما نحكم على الحلول التي اخترناها، علينا أن نفكر في عدة عوامل مثل الوقت اللازم، وهل يمكن الوثوق في قدرتها على حل المشكلة؟

هناك طرق مختلفة يمكننا من خلالها تقييم الحلول، فيمكننا أن نختبر سـرعتها بتطبيقها على الحاسـوب، أو يمكننا أن نحللها بالعد أو بحســـاب عدد الخطوات التي ســـتحتاجها غالباً، ويمكننا أن نختبر ما إذا كانت الحلول تعمل بصورة صحيحة بإعطائها الكثير من المدخلات المختلفة لمعرفة إن كانت تعمل كما يجب.

❖ **التعميم :Generalizing**تضـــمن مهارات التعميم الاســـتفادة من العمليات المســـتخدمة في حل مشكلة حاسوبية معينة وتطبيقها على مجموعة متنوعة من المشكلات، بمعنى حل المشكلات الجديدة بشكل سريع استناداً إلى المشكلات السابقة التي قام الفرد بحلها.

❖ **التفكير المنطقي :Logical thinking**عندما نحاول حل المشـــكلات، علينا أن نفكر بصـــورة منطقية، التفكير المنطقي يعني محاولة فهم الأمور منطقياً من خلال الملاحظة، وجمع البيانات والتفكر في الحقائق التي نعرفها ثم اسـتنتاج الأشـياء بناءً على ما نعرفه مسـبقاً. يسـاعدنا التفكير المنطقي في اسـتخدام معرفتنا السابقة لوضع القواعد واختبار الحقائق، وعلى مستوًى أعمق تعتمد الحواسيب كلياً على المنطق، وتسـتخدم قيم "صـواب" و"خطأ"، وتسـتخدم ما يُعرف باسـم التعبير المنطقيBoolean Expression ، لاتخاذ قرارات في برامج الحاسوب.

2. **التعرف على بيئة العمل مثل Jupyter Notebooks**

هي بيئة عمل **تفاعلية** .تسـتخدم في مجالات مثل **علوم البيانات** وتطوير البرمجيات. توفر طريقة فعالة لتنفيذ التجارب وتحليل البيانات.

وهي تطبيق ويب يســـمح بإنشـــاء وتحرير .*notebooks*تحتوي على كود برمجي ونتائج تنفيذه. تدعم لغات برمجة متعددة، مما يجعلها أداة مفيدة.

بيئة Jupyter تتكون من عدة مكونات:

❖ **Notebook** واجهة المستخدم لكتابة وتنفيذ الكود

❖ **Kernel** المحرك الذي ينفذ الكود البرمجي.

➢ **مجالات استخدامه:**

❖ **تحليل البيانات وعلوم البيانات:**

Jupyter Notebooksتســـتخدم كثيرًا في **تحليل البيانات** وعلوم البيانات. تتيح بيئة تفاعلية للمحللين والباحثين. يستطيعون استكشاف وتحليل البيانات بسهولة. تستخدم لغات برمجة مثل Python و .Rتسهل تصور البيانات وتقديم النتائج بطريقة واضحة.

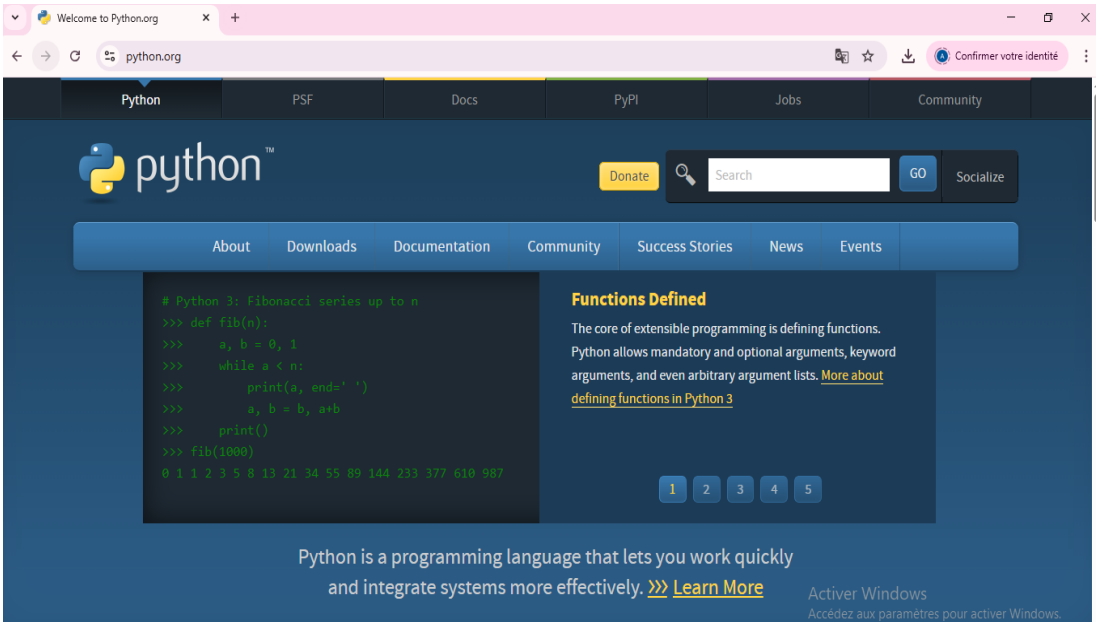
❖ **التعليم والتدريب البرمجي:** في مجال *التعليم*، Jupyter Notebooks أداة تعليمية فعالة. يمكن للمعلمين إنشـــاء دروس تفاعلية. يســـهل ذلك عملية التعلم ويزيد من تفاعل الطلاب. تســـتخدم أيضًـــا في الدورات التدريبية، تعليم البرمجة وتحليل البيانات أصبح أسهل.

❖ **البحث العلمي والأكاديمي:** في **البحث العلمي** والأكاديمي، Jupyter Notebooks تلعب دورًا هامًا. تسـهل عملية البحث للباحثين. يمكنهم اسـتخدامها لتحليل البيانات وتقديم النتائج، تسـهل إنشـاء نماذج تفاعلية. تسهل مشاركة النتائج مع الزملاء والتعاون في المشاريع البحثية.

❖ **تطوير نماذج الذكاء الاصطناعي والتعلم الآلي:** Jupyter Notebooks تستخدم أيضًا في تطوير نماذج الذكاء الاصـــطناعي والتعلم الآلي. تتيح للمطورين تصـــميم وتدريب النماذج. يمكنهم اختبارها وتحسين أدائها، تسهل تصور البيانات المستخدمة في التدريب. تقدم النتائج بطريقة واضحة.

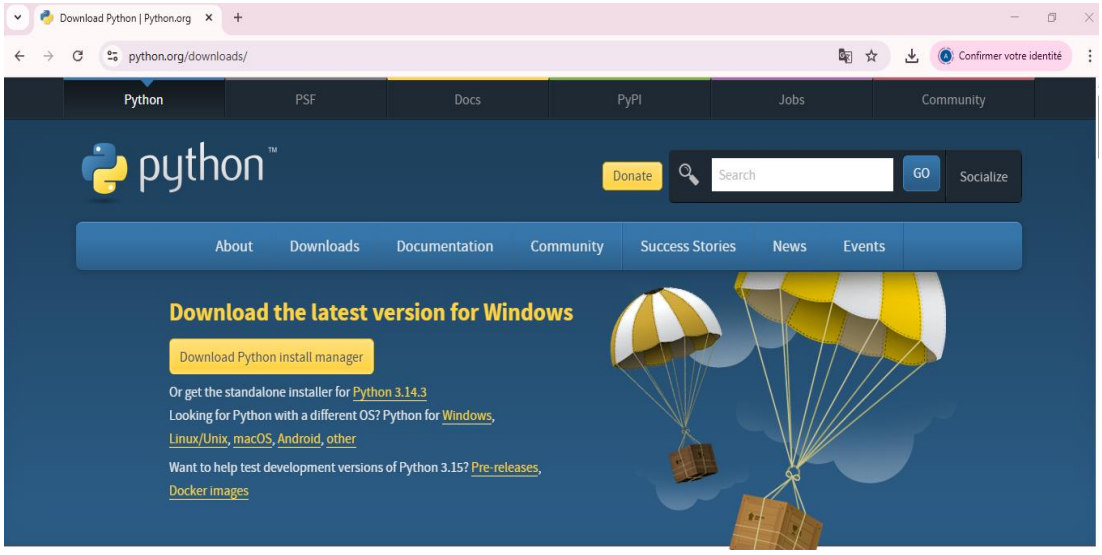
3. **<u>اللبنات الأساسية للغة Python:</u>**

- **اختيار الويندوز**: اخترت في التثبيت ويندوز 10 - 11
- **تثبيت بايثون:** لتحميل نسـخة بايثون، نقوم بزيارة متصـفح قوقل وكتابة Python، تظهر لنا نافذة نختار الموقع الرسمي : https://www.python.org/، تظهر لنا شاشة البرنامج

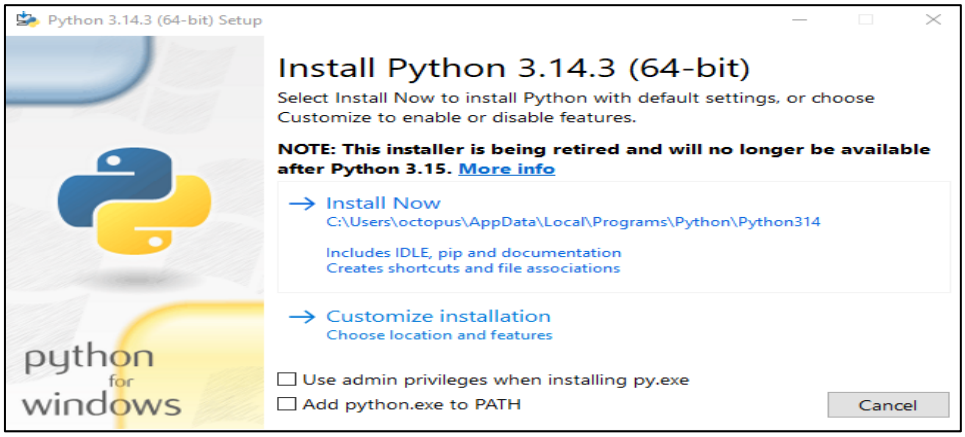


المصدر: اعداد الأستاذة أسماء صالحي

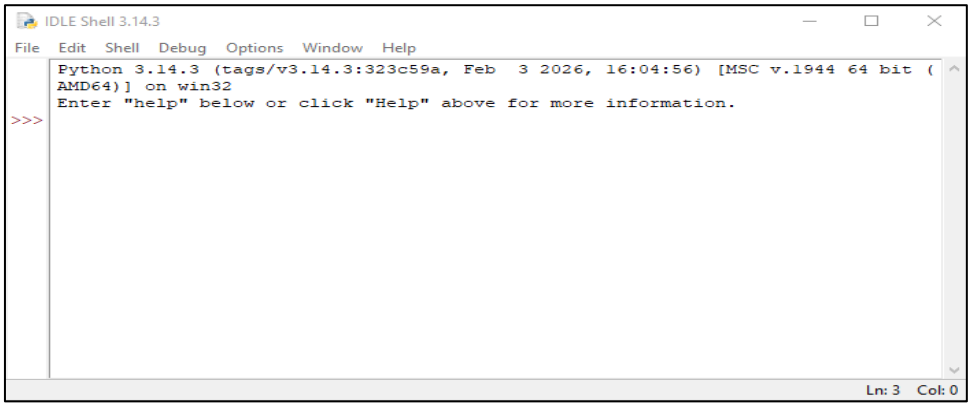
- الخطوة الثانية هي تنصيب البايثون وذلك بالنقر على ايقونة Downloads، ثم نثبت بايثون



المصدر: اعداد الأستاذة أسماء صالحي

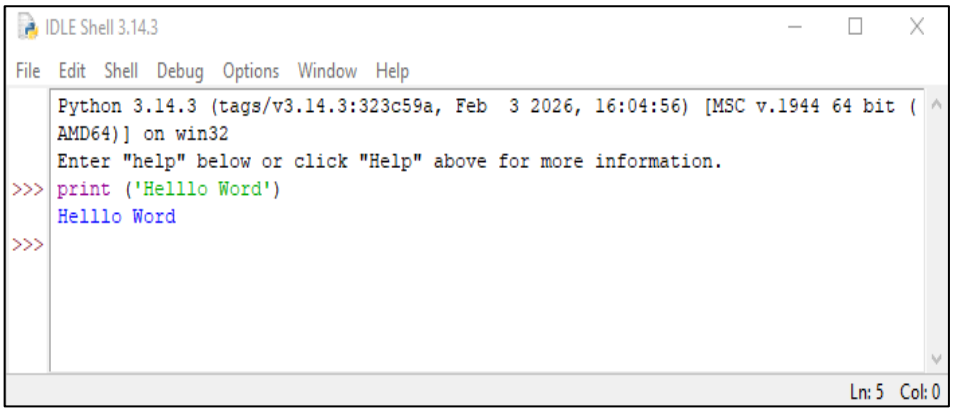


المصدر: اعداد الأستاذة أسماء صالحي



المصدر: اعداد الأستاذة أسماء صالحي

- بعد تثبيت برنامج بايثون، نبدأ بكتابة «برنامج الترحيب **(Hello Word)** وتكون الشفرة بالشكل الآتي:



المصدر: اعداد الأستاذة أسماء صالحي

❖ **دراسة المتغيرات:**

تمثِّل المتغيرات في أيّ لغة برمجية أماكن في ذاكرة الحاسوب مخصصةً لتخزين البيانات، وذلك يعني أنك عندما تعرِّف متغيرًا ما حسب قواعد المعرِّفات التي ذكرناها في المقال السابق، فأنت بذلك تحجز مكانًا في الذاكرة لتخزين ذلك المتغير وما يحتويه.

❖ **أنواع البيانات الأساسية (الأرقام، النصوص، القيم المنطقية، العمليات الحسابية):**

في البرمجة، يقوم كل متغير بتخزين **قيمة** معينة من **نوع** محدد. على سبيل المثال، يمكن أن تكون أنواع البيانات :**رقماً، أو حرفاً، أو نصاً (سلسلة نصية)، أو تاريخاً، أو لوناً، أو صورة، أو قائمة،** وغيرها.

بعض الأمثلة على أنواع البيانات وقيمها:

- **int عدد صحيح :** (مثل 1، 2، 3، 4، 5...)
- **float عدد عشري/نقطة عائمة) :** مثل 0.5، 3.14، 1.5...-)
- **Str سلسلة نصية :**(وهي نصوص مكونة من رموز، مثل'a' ، 'Hello'، 'Hi'، 'Beer'...)
- **datetime التاريخ والوقت :**(مثل 2017-12-21، 1995/07/25...)

أما في لغة **بايثون(Python)** ، فإن نوع البيانات يتحدد بناءً على القيمة التي يتم تعيينها للمتغير، ولا يتم تحديده بشكل صريح عند الإعلان عن المتغيرات) على عكس لغات مثل C#، C++، Java[1].

- **الأرقام:**

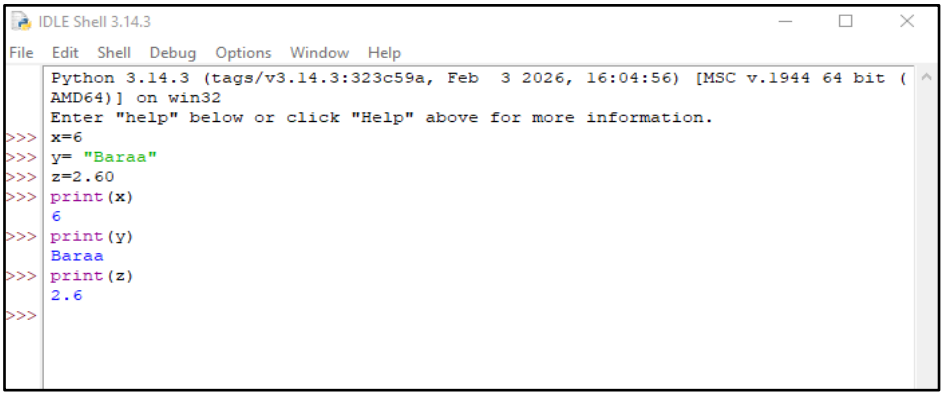
مثال: x = 6 إذن لدي المتغير x يحمل القيمة 6

لدينا y = Baraa متغير نصي اسمه براء أو يحمل القيمة "براء"

ولدينا z = 2.60 وهو قيمة كسرية

[1] Chapter 2.1. Simple Calculations.
https://python-book.softuni.org/chapter-02-simple-calculations.html (08-04-2026)

نقوم بكتابة كل متغير على حدى ونكتب: print سيقوم بإعطاء نتيجة المتغيرات كما هو موضح:



المصدر: اعداد الأستاذة أسماء صالحي

وإذا أردنا تغيير قيمة x إلى قيمة كسرية نكتب x ="string " ، ثم نكتب print تكون النتيجة معرفة ب x.

تحويل المتغيرات:

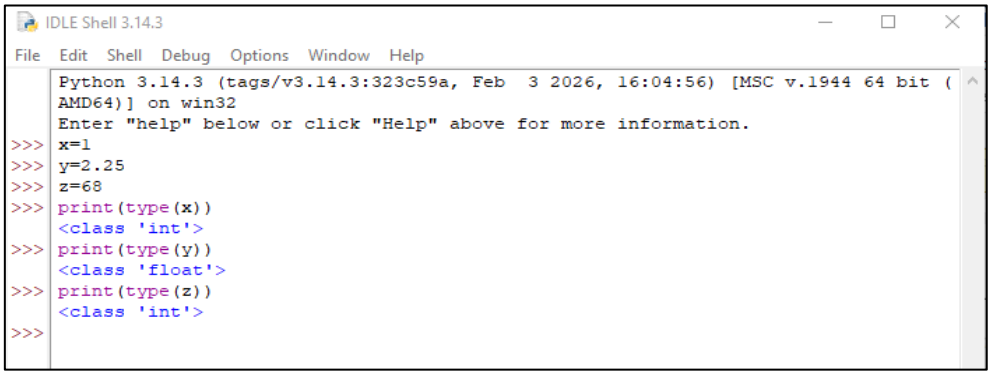
نستطيع تحويل المتغيرات بعدة طرق:

مثال: x=1 عدد صحيح

و y= 2.25 وهو عدد عشري

و z= "68" متغير نصي

نقوم بطباعة أنواع هذه المتغيرات في برنامج باثون، ثم تظهر النتيجة كما هي موضحة في الصورة التطبيقية.

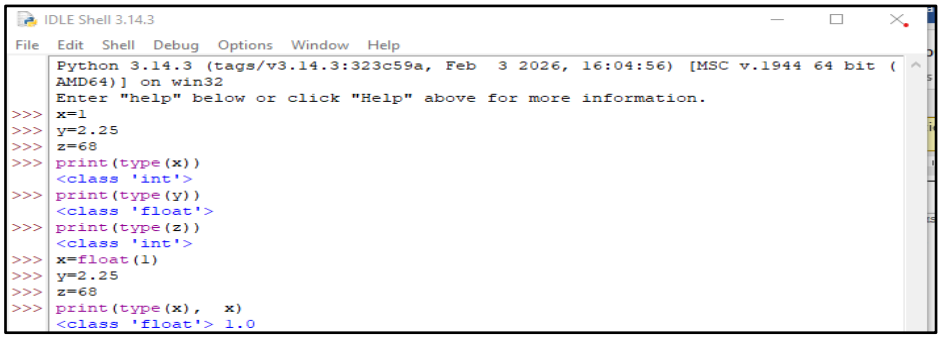


المصدر: اعداد الأستاذة أسماء صالحي

ونستطيع تحويل المتغير x من عدد صحيح إلى عدد كسري مثال:

x= float

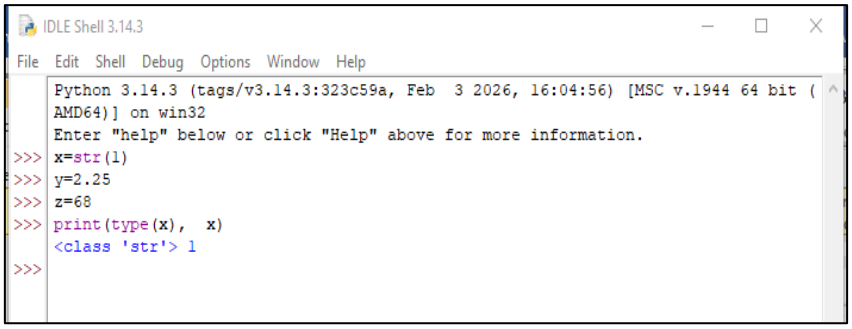
تكون نتيجة:



المصدر: اعداد الأستاذة أسماء صالحي

إذن تحول العدد الصحيح إلى عدد كسري مع إضافة عدد0 من نوع float.

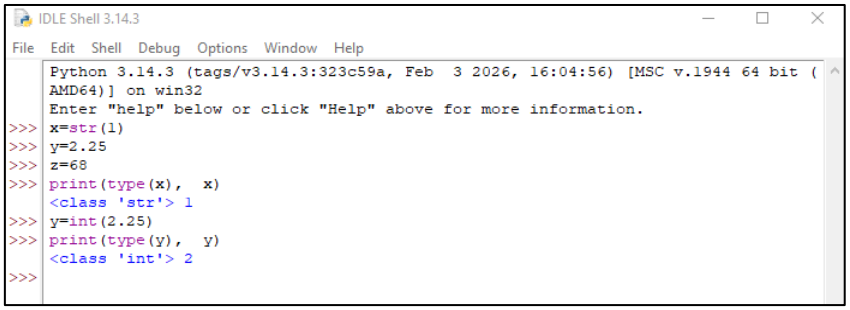
وإذا أردنا تحويل x=1 إلى عدد نصي x=**str**(1) فإن النتيجة تكون كالتالي:



المصدر: اعداد الأستاذة أسماء صالحي

كما أننا نستطيع تغيير النوع الكسري إلى عدد صحيح مثال:

y=2.25 ، نقوم بكتابة y=int (2.25)، تكون النتيجة كما هي موضحة :

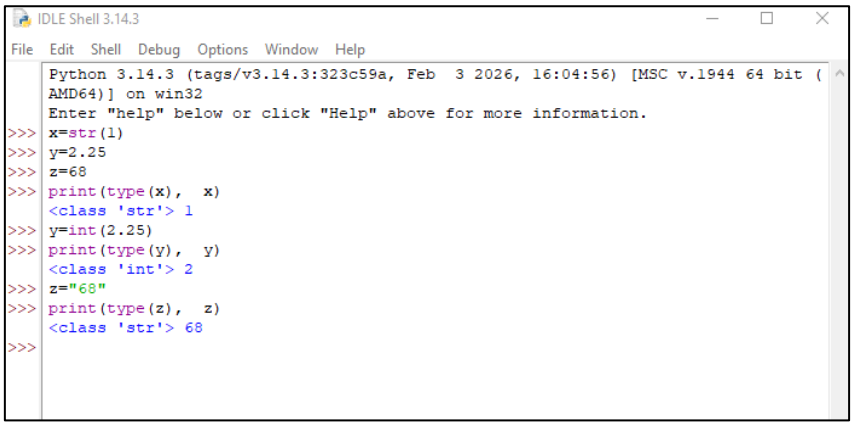


المصدر: اعداد الأستاذة أسماء صالحي

إذن تم إلغاء الكسر وتحويله إلى عدد صحيح.

كما نستطيع تحويل النوع النصي إلى متغير صحيح مثال:

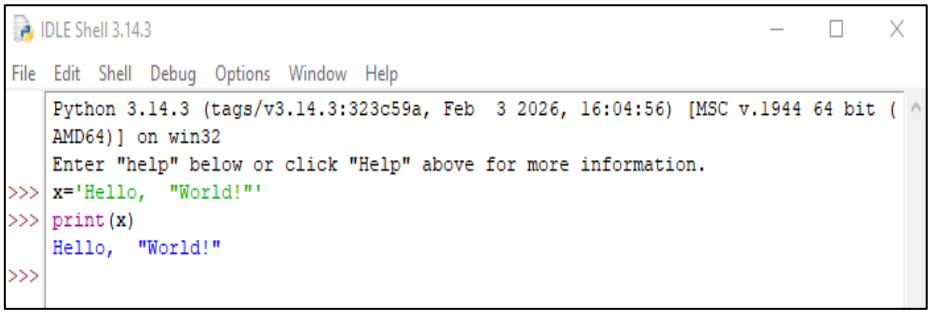
z="68"، نكتب z=int"68"، تكون النتيجة كالتالي:



المصدر: اعداد الأستاذة أسماء صالحي

- **المتغيرات النصيةStrings :**

مثال: " Hello, World! "=xونكتب المتغير النصـــي بين علامتين مزدوجتين و هذا بهدف الكتابة النصية المفردة والمزدوجة.

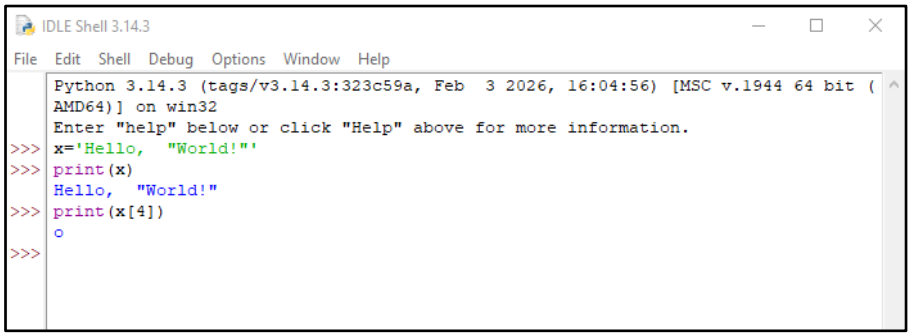


المصدر: اعداد الأستاذة أسماء صالحي

النتيجة كتابة النص كاملا.

وإذا أردنا كتابة الحروف على حدى، فإن لكل حرف موقع خاص به أو عنوان خاص به، وترقم من **0** إلى آخر حرف، مع حساب المسافة بين الحروف، والفاصلة بين الكلمات،

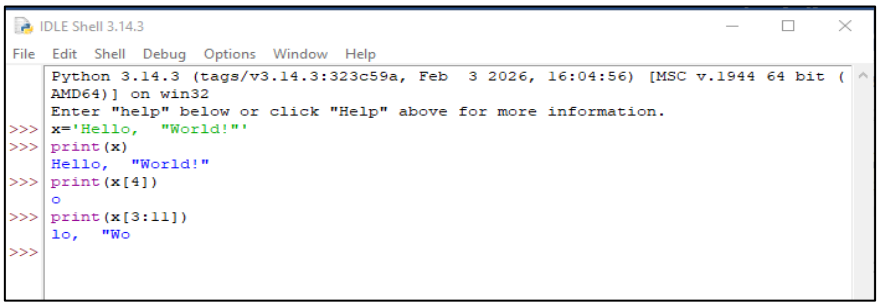
مثال: إذا أردنا كتابة حرف o، نبدأ بحســاب من عدد 0 وصــولا إلى حرف o وهو يمثل العدد4، وهو موضح في الصورة التطبيقية.



المصدر: اعداد الأستاذة أسماء صالحي

مثال 2: إذا أردنا كتابة من حرف lo إلى Wo، نبدأ من حرف H الذي يمثل عدد 0 وصولا إلى Wo

إذن l=3 و o=4 وصولا إلى o=10 مع حساب المسافة والفاصلة، يعني نكتب من 3 إلى 11.



المصدر: اعداد الأستاذة أسماء صالحي

النتيجة: Wo" , 10.

- **القيم المنطقية:**

القيمتان المنطقيتان البوليانيتان، (Boolean) اللتان تدعمهما بايثون هما **True** و **Flase** وهما كائنان ثابتان (Constant objects) يعبران عن صحّة تعبير ما، فإمّا أن يكون صحيحًا True أو خطأً False. تعدّ القيم المنطقية نوعًا فرعيًا (subtype) من **الأعداد الصحيحة**، وتسلك القيمتان **True** و **Flase** سلوك القيمتين 0 و1 على التوالي في معظم السياقات تقريبًا، ويستثنى من ذلك تحويل القيم المنطقية إلى **سلاسل نصية**، فتعاد حينئذ السلسلتان النصيتان **True** و **Flase** على التوالي.

```
>>> foo = True
>>> bar = False
```

مثال تطبيقي:

```
>>> bool(2)
True
>>> bool("Python")
True
>>> bool(0)
False
>>> bool()
False
>>> bool(None)
False
```

يمكن اختبار أي كائن لمعرفة القيمة الصحيحة وذلك لاستخدامه كشرط في عبارة **if** أو **while** أو كمعامل للعوامل المنطقية.

تعدّ بايثون الكائن صحيحًا بصورة افتراضية إلا إذا كان صنفه يعرّف التابع __bool__() الذي يعيد القيمة False أو تابع __len__() يعيد القيمة "صفر" عند استدعائه مع الكائن.

فيما يلي قائمة بالكائنات الداخلية التي تعدّها اللغة **False**:

- الثوابت المعرّفة على أنّها خطأ: وهي **None** و**False**.
- الصفر في أي نوع عددي**0**: ، أو **0.0**، أو **Fraction(0,1)** ، أو **Decimal(0)**، أو **0j**.
- التسلسلات والمجموعات الفارغة**''**: ، أو **()**، أو **[]**، أو **{}**، أو **range(0)**، أو **set().**
- تعيد العمليات والدوالّ الداخلية التي تمتلك نتائج منطقية (بوليانية) دائمًا القيمة **0**أو **False**للخطأ، وتعيد **1**أو **True**للقيم الصحيحة، إلا إذا صُرّح بغير ذلك (باستثناء العمليتان المنطقيتان **or** و **and**اللتان تعيدان دائمًا واحدًا من عواملهما فقط، وتعرف هذه الحالة بالدارة القصيرة short-circuit. .

يمكن استخدام الدالة الداخلية **bool()**لتحويل أي قيمة إلى قيمة منطقية إذا كان بالإمكان التحقق من صحّة القيمة.

- **العمليات الحسابية:**

الوظيفة(Function)	الصيغة البرمجية (Syntax)	الصيغة البديلة (Alternative Syntax)
الجمع(Addition)	x + y	add(x, y)
الطرح(Subtraction)	x - y	subtract(x, y)
الضرب (Multiplication)	x * y	multiply(x, y)
القسمة(Division)	x / y	divide(x, y)
باقي القسمة (Remainder)	x % y	remainder(x, y)
اختبار التساوي(Test of equality)	x == y	equals(x, y)
اختبارات عدم التساوي	x > y; x < y	x >= y; x <= y
القيمة المطلقة (Modulus/Absolute)	abs(x)	–

الأس/القوة(Power)	x ** y	pow(x, y)
الدالة الأسية (Exponential)	exp(x)	e ** x

المصدر: اعداد الأستاذة أسماء صالحي اعتماد على المرجع:

PX1224: Computational Skills for Problem Solving, Python as a calculator.

https://physics-python.astro.cf.ac.uk/Week1/ (09-04-2026)

4. هياكل التحكم:

❖ استخدام العبارات الشرطية if/ elife/ else

جميع لغات البرمجة التي نعرفها تحتوي على تعبير خاص بالجملة الشرطية، وتكاد تتفق أغلب اللغات على الهيكل او الترتيب العام للجملة الشـــرطية والذي يكون if/else if/else. البايثون تحتوي على نفس الأمر. من المهم أن نذكر هنا أن بعض لغات البرمجة لديها شـــكل مشـــهور للجملة الشـــرطية وهو تعبير case/switch، ولكن لغة البرمجة البايثون لا تحتوي على هذا التعبير.

كل ما تقوم به الجُملة الشرطية هو أنها تفحص تحقق شرط ما، وبناءً على ذلك يتم تنفيذ شيفرة برمجية ما.

مثال تطبيقي:

يُوضح لنا كيفية كتابة الجُملة الشرطية والتي تُعتبر أحد جُمل التحكم في بايثون كما يلي:

```
some_var = 11

if some_var > 10:
    print("some_var is totally bigger than 10.")
```

المثال السابق يحتوي على جملة شرطية تفحص إذا كان المتغير المُسمى بـ some_var أن قيمته أكبر من 10 ام لا، وإذا كان كذلك يتم طباعة الجملة some_var is totally bigger than 10. ، ولكن ماذا لو كانت قيمة المتغير أقل من 10؟ عندها يُمكننا اضـــافة جملة شـــرطية ثانية تتبع الأولى عن طريق تعبير elifكما يلي:

```
some_var = 5

if some_var > 10:
    print("some_var is totally bigger than 10.")
elif some_var < 10:    # هذه الجملة اختيارية
```

إذا لم تتحقق أي جملة شرطية، يُمكن لنا تنفيذ شيفرة برمجية باستخدام تعبير else والذي سيتضمن الشيفرة التي ستقوم بايثون بتنفيذها عندما لا تتحقق أي جملة شرطية كما في المثال التالي:

```
some_var = 10

if some_var > 10:
    print("some_var is totally bigger than 10.")
elif some_var < 10:    # هذه الجملة اختيارية
    print("some_var is smaller than 10.")
else:                  # هذه الجملة اختيارية
    print("some_var is indeed 10.")
```

❖ **الحلقات التكرارية while /for**

حلقة **For** جملة برمجية أخرى تُمكّن من التكرار، وهي من الأساليب المُستعملة للدوران حول قيم القوائم كذلك، وتتميّز بتعريف مُتغيّر العدّ في بداية الحلقة، أي أنّك على خلاف حلقة While لا تحتاج إلى تعريف المُتغيّر قبل البدء بكتابة الحلقة، كما أنّ حلقة for لا تحتاج إلى شرط معيّن بل تُحدّد عدد مرّات التّكرار حسب عدد عناصر القائمة المُسندة.

<u>مثال تطبيقي:</u>

لاحظ بأنّ الجمل التّابعة للسّطر for counter in range1, 11 مُزاحةٌ بأربع مسافات بيضاء.

```
>>> for counter in range(1, 11):
...     print counter, 'Hello World!'
...

1 Hello World!
2 Hello World!
3 Hello World!
4 Hello World!
5 Hello World!
6 Hello World!
7 Hello World!
8 Hello World!
9 Hello World!
10 Hello World!
```

الدّالة range تقوم بإنشاء قائمة أعداد مرتّبة من الأصغر إلى الأكبر حسب عاملين الأول رقم البداية والثّاني يُمثّل النّهاية مع ملاحظة بأنّ الرّقم النّهائي في السّلسلة لا يحتسب (في المثال التّالي 1 هو العدد الأول والعدد النّهائي هو العدد 11).

```
>>> range(1,11)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

أما حلقة While تقوم بتكرار شيفرة ما عندما يكون الشّرط محقّقا ولا تتوقّف إلا عندما يكون الشّرط خاطئا، ولإنشاء حلقة while يجب تحديد عدد المرّات التّي تتكرّر فيها ووضعها كشرط ثم زيادة قيمة

مُتغيّر بواحد، بحيث يزداد إلى أن يصل إلى العدد المُحدّد في الشّرط فيتوقّف. بإمكانك أيضًا تحديد شرط من نوع آخر بحيث لا ترتبط الحلقة بتنفيذ الشيفرة بعدد مُعيّن من المرّات وإنما بتحقق شرط مُعيّن (أن تصبح قيمة مُتغيّر مُخالفة لقيمة نقوم بتحديدها) .

مثال تطبيقي:

يُمكننا طباعة "Hello" عشر مرّات بالطّريقة التّاليّة، لاحظ بأنّ الجمل التّابعة للكلمة while مُزاحةٌ بأربع مسافات بيضاء:

```
>>> i = 0
>>> while i < 10:
...     print "Hello"
...     i = i +1

Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

السّطر الأول :نقوم بوضع قيمة بدئية للمتغيّر i.

السّطر الثّاني :نقوم بإضافة الشّرط وهو أن تكون قيمة المتغير أصغر من العدد عشرة فإن كان يُساوي 10 يتوقّف البرنامج .

السّطر الثّالث :نقوم بطباعة الكلمة

السّطر الرّابع :نقوم بزيّادة قيمة المتغير

في المرّة الأولى سيقرأ المفسّر قيمة المُتغيّر فيجدها تُساوي العدد 0، ثمّ يتحقّق من أنّ شرط أن تكون القيمة أصغر من 10 صحيح، وبما أنّ الشّرط مُحقّق في المرّة الأولى (بالطّبع لأنّ 0 أصغر من 10) سيُتابع البرنامج العمل، وسيطبع جملة التّرحيب "Hello" بعد ذلك يزيد المُفسّر قيمة المتغيّر i بواحد (لتكون القيمة الجديدة هي 1) يتحقّق المُفسّر من صحّة الشّرط مُجدّدا وبما أنّه مُحقّق فقد طُبعت الجملة "Hello"مرّة ثانيّة، بعدها يزيد من قيمة المُتغيّر بواحد مجدّدا لتكون القيمة الجديدة للمُتغيّر i هي العدد 2 والذي بدوره أصغر من 10 فتُطبع الجملة مرّة ثالثة. وهكذا دواليك إلى أن تصل قيمة المُتغيّر إلى العدد 10 فيُصبح الشّرط خاطئا (لأنّ العدد 10 ليس أصغر من 10 بل مُساو له)، وبالتّالي يتوقّف التّكرار .

5. هياكل البيانات:

❖ القوائم Lists :

تتوفر في بايثون ما يُدعى بالقوائمLists ، وهي تشـــبه إلى حد كبير المصـــفوفات في لغات البرمجة الأخرى، فببساطة يمكن تعريف القائمة في بايثون بأنها متغير يحتوي على مصفوفة من البيانات والتي من الممكن أن تكون من أنواع مختلفة، كما من الممكن التعامل معها كما أوردنا في التعامل مع النصوص على أســاس مصــفوفة، إذ تُعيَّن قيمة متغير القائمة عبر كتابة القيم بصــورة متتابعة يليها فاصــلة ,بين الأقواس المعقوفة [].

❖ القواميس Dictionary :

يشــبه القاموس Dictionary في بايثون جدول Hash أي Hash-table وهو إحدى هياكل البيانات في علوم الحاسوب بحيث تُربَط القيم بمفتاح ومن ثم البحث عن القيم بسرعة كبيرة إذ أنها تعمل على أساس مصـــفوفة تحتوي على أزواج من المفاتيح والقيم، بحيث يوجد مفتاح ما لكل قيمة يتيح الوصـــول إليها، وتُعرَّف القواميس عبر الأقواس المعقوصـــة { }، كما أنّ القواميس بدورها غير مرتبة، بعكس الصـــفوف والقوائم التي تُرتَّب فيها مفاتيح كل عنصـــر بالترتيب الرقمي بدءًا من الصـــفر أي يعيّن المبرمج مفاتيح القواميس والتي من الممكن أن تكون نصيةً أو عدديةً بدون ترتيب.

❖ المجموعات Sets:

تشـــبه المجموعات Sets والتي هي هياكل البيانات في بايثون إلى حد كبير الصـــفوف، لكنها تُكتب باستخدام الأقواس المعقوصة مثل القواميس.

قد يختلف ترتيب العناصـــر عند طباعة المجموعة لأن المجموعات في الأســـاس لا ترتِّب العناصـــر؛ لذلك قد يختلف ترتيب عناصر المجموعة في كل مرة يُشغَّل فيها البرنامج، أيضًا من الضروري معرفة أنّ المجموعات لا تُذكَر كثيرًا في الدورات ومساقات التعليم، إذ أنها محدودة للغاية في استخداماتها، ولا يمكن طباعة عنصر معيّن منها إلا بطرق معقدة وهي لا تستخدَم بصورة شائعة.