

هيبا نتعلم البرمجة بلغة البايثون معا

تأليف:

عدايبية عامر

هيا نتعلم البرمجة بلغة البايثون معا

تأليف :عرايبيه عامر .

الكتاب : عبارة عن صدقة جارية.

اهداء : اهدي هذا لكل مبرمج على طريق الاحتراف.

تنويه : ان احسنت فمن الله،وان اخطأت فمن نفسي والشيطان.

كلمة عن الكتاب : هذا الكتاب يشترط خبرة برمجية بسيطة سابقة ،

وان كانت غير موجودة فلا ضرر .

الجهاز المساعد في التأليف :

بواسطة الهاتف فقط وقليل من المعلومات من محركات الذكاء الاصطناعي

لإثبات للمتعلمين أن العلم لا يحتاج الأجهزة الكبيرة والغالية، لكن هذا لا ينفي

ضرورة استخدام الحاسوب في العديد من المشاريع .

أ- التمهيد :

بالطبع! البرمجة بلغة بايثون مثيرة ومفيدة جدًا. إليك تمهيد قصير لكتابة برامج باستخدام بايثون:

أ.أ- تثبيت بايثون:

يجب أن تبدأ بتثبيت بايثون على جهاز الكمبيوتر الخاص بك. يمكنك تنزيل أحدث إصدار من موقع الويب الرسمي لـ بايثون وتثبيته

أ.ب- البداية في البرمجة:

يمكنك البدء بالتعرف على مفهوم البرمجة بإطار عام. الفهم الجيد للأساسيات مهم جدًا

أ.ج- البيئة التطويرية (IDE):

يمكنك استخدام بيئة تطويرية مثل:

IDLE,
Visual Studio Code,
PyCharm,

لكتابة وتشغيل البرامج باستخدام بايثون.

أ.د- مفاهيم بايثون الأساسية:

-المتغيرات: تُستخدم لتخزين البيانات .

-الأنواع الأساسية: مثل الأعداد الصحيحة والنصوص (السلاسل) .

-العمليات الأساسية: الجمع، الطرح، الضرب، القسمة .

-الجمل التحكمية: مثل الشروط والحلقات .

أ.و- المكتبات والوحدات:

بايثون لديها مجموعة كبيرة من المكتبات والوحدات التي يمكنك استيرادها واستخدامها لتسهيل البرمجة

أ.ه- المشروعات الصغيرة:

قم ببناء مشروعات صغيرة لتطبيق ما تعلمته. مثل برنامج لحاسبة بسيطة أو مدير مهام بسيط

أ.ة- المصادر التعليمية:

هناك العديد من الكتب والدروس على الإنترنت التي تساعدك على تعلم بايثون بشكل متقدم

أ.ي- مشاركة المعرفة:

لا تتردد في طرح الأسئلة ومشاركة معرفتك مع المجتمع المبرمجي. منتديات الإنترنت ومواقع مثل

Stack Overflow

تكون مفيدة لهذا الغرض.

هذا التمهيد مجرد مقدمة سريعة. بايثون تقدم الكثير من الإمكانيات، لذا استمتع برحلتك في عالم البرمجة باستخدامها .

ب- مقدمة

بسم الله الرحمن الرحيم

والحمد لله رب العالمين والصلاة والسلام على سيد المرسلين محمد صلى الله عليه و سلم
اما بعد:

مرحبًا بك في كتابنا حول لغة البرمجة بايثون! تعتبر بايثون واحدة من أكثر اللغات شيوعًا وشعبية في عالم البرمجة، وهذا الكتاب سيكون مرافقك المثالي لاكتشاف عالم بايثون واستفادة قصوى من قوتها ومرونتها .

تميز بايثون بسهولة تعلمها وفهمها، حيث تتميز بقوة تنفيذية مذهلة ومجموعة واسعة من و المكتبات والإطارات التي تجعلها مناسبة لمجموعة متنوعة من التطبيقات. سواء كنت مبتدئًا في عالم البرمجة أو مطورًا محترفًا، فإن بايثون ستكون أداة قوية في يديك.

في هذا الكتاب، سنستكشف تدريجيًا مفاهيم بايثون الأساسية، بدءًا من المتغيرات والتوجيهات وصولًا إلى المواضيع المتقدمة مثل البرمجة الكائنية ومعالجة البيانات والواجهات الرسومية. سنقوم أيضًا بتوجيهك في كيفية استخدام بايثون في مجالات مختلفة مثل تطوير الويب والذكاء الاصطناعي وعلم البيانات .

سوف نقدم لك أمثلة عملية وتحديات لحلها، بالإضافة إلى نصائح واستراتيجيات لتحسين مهارات البرمجة الخاصة بك بغض النظر عن هدفك في تعلم بايثون، سيكون هذا الكتاب دليلك الموثوق والمصدر المفيد لاكتساب المعرفة والخبرة. دعونا نبدأ في استكشاف عالم بايثون ونبني معًا تطبيقات مذهلة !

وفي النهاية انا ايضا ساتعلم معك وسانقل تجربتي الصغيرة إليك مع عالم البايثون وكلما أتطور في هذه اللغة سأزيد من تطورك ايضا باذن الله.

ج -البايثون؟ ماهي لغة البايثون؟

ج.أ- تعريف مختصر للبايثون:

البايثون هو لغة برمجة عالية المستوى ومفتوحة المصدر تم تطويرها في أوائل التسعينيات. إنها تشتهر بقراءة الكود بسهولة وببساطة الاستخدام، مما يجعلها مثالية للمبتدئين ولأغراض متنوعة. البايثون تعتبر متعددة الاستخدامات وتدعم العديد من التطبيقات بما في ذلك تطوير الويب والذكاء الاصطناعي، وتحليل البيانات، والأتمتة، والألعاب، وأكثر من ذلك. إنها معروفة أيضًا ببيئتها البرمجية البسيطة والواضحة وبمجموعة كبيرة من المكتبات المتاحة للمطورين لتسهيل عمليات البرمجة.

ج.ب- تعريف مفصل للبايثون:

البايثون هو لغة برمجة عالية المستوى ومفتوحة المصدر تم تطويرها أصلاً بواسطة

Guido van Rossum

في أوائل التسعينيات. إليك تعريف مفصل للبايثون:

ج.ب.أ- سهولة القراءة والكتابة : البايثون تشتهر بقواعدها البسيطة والواضحة التي تجعل من السهل قراءة الكود وكتابته. هذا يساهم في جعلها مثالية للمبتدئين وتعزيز إنتاجية المطورين

ج.ب.ب- متعددة الاستخدامات: تستخدم البايثون في مجموعة متنوعة من التطبيقات بما في ذلك تطوير الويب عبر إطارات مثل :

Django,
Flask,

وعلم البيانات باستخدام مكتبات مثل :

NumPy,
Pandas,

والذكاء الاصطناعي وتعلم الآلة بواسطة :

TensorFlow,
PyTorch,

وأتمتة الأنظمة، وتطوير الألعاب، وغيرها .

ج.ب.ج- مفتوحة المصدر: البايثون متاحة كبرمجية مفتوحة المصدر، مما يعني أنها مجانية ويمكن للمطورين تعديلها وتوسيعها واستخدامها بحرية

ج.ب.د- بيئة تطوير واسعة: توفر البايثون مجموعة كبيرة من المكتبات والإطارات التي تسهل على المطورين إنشاء تطبيقات معقدة بسرعة. هذه المكتبات تشمل مكتبات للعمل مع البيانات، ومكتبات رسومية، ومكتبات لتطوير التطبيقات النقالة، والمزيد

ج.ب.د- دعم مجتمعي قوي: يتمتع البايثون بمجتمع نشط من المطورين حول العالم، مما يعني وجود مصادر ومساعدة ودعم واسعين للمطورين الجدد متوفرة على معظم الأنظمة، يمكن تشغيل البايثون على معظم أنظمة التشغيل الشائعة بما في ذلك:

Windows,
macOS,
Linux,

مما يجعلها قابلة للتنفيذ على مجموعة متنوعة من البيئات.

باختصار، البايثون هي لغة برمجة قوية ومتعددة الاستخدامات تمتاز بسهولة الاستخدام وتوفير العديد من المكتبات والأدوات لتسهيل عملية تطوير التطبيقات والمشاريع المختلفة

د- تثبيت البايثون :

د.أ- كيفية تثبيت بايثون على ويندوز:

حتى يتمكن أي مطور أو مبرمج من استخدام لغة بايثون يجب التأكد من وجودها على جهاز الحاسوب الخاص به، فإذا كان الجهاز يعمل بنظام التشغيل لينكس أو نظام ماك ، فلا داعي لتنزيل لغة البايثون؛ لأنها مُضمنة بالفعل مع نظام التشغيل، أما إذا كان الجهاز يعمل بنظام التشغيل ويندوز، فيجب تثبيت لغة بايثون على الجهاز، وهناك عدة طرق مختلفة لتثبيت لغة بايثون على نظام التشغيل ويندوز، ولكل منها مزايا وعيوب معينة .

د.ب- طريقة التثبيت الكامل :

يمكنك تثبيت بايثون على جهاز الكمبيوتر الخاص بك باستخدام الخطوات التالية :

الخطوة رقم 1 : قم بزيارة موقع بايثون الرسمي على الويب عبر الرابط التالي

<https://www.python.org/downloads/>

الخطوة رقم 2 : انقر على رابط

Download Python

الموجود في الصفحة الرئيسية

الخطوة رقم 3 : ستظهر لك قائمة بإصدارات بايثون المتاحة للتنزيل. اختر الإصدار الذي ترغب في تثبيته عادةً ما يتم توصية باستخدام أحدث إصدار متاح

الخطوة رقم 4: بعد اختيار الإصدار، انقر على الرابط المناسب لنظام التشغيل الذي تستخدمه لينكس، ويندوز، ماك .

الخطوة رقم 5: بدأ التنزيل بعد النقر على الرابط. عند اكتمال التنزيل، افتح ملف التثبيت الذي تم تنزيله .

الخطوة رقم 6: في نافذة المثبت، تأكد من التأشير على الخيار

"Add Python X.X to PATH"

(يمثل إصدار بايثون الذي تثبته X.X).

الخطوة رقم 7: انقر على زر:

"Install Now"

واتبع الخطوات الباقية في المثبت.

الخطوة رقم 8: بعد اكتمال التثبيت، يمكنك فتح نافذة الأوامر أو الطرفية واختبار تثبيت بايثون بكتابة الأمر :

"python --version". On cmd or terminal

ستظهر إصدار بايثون على الشاشة إذا تم التثبيت بنجاح.

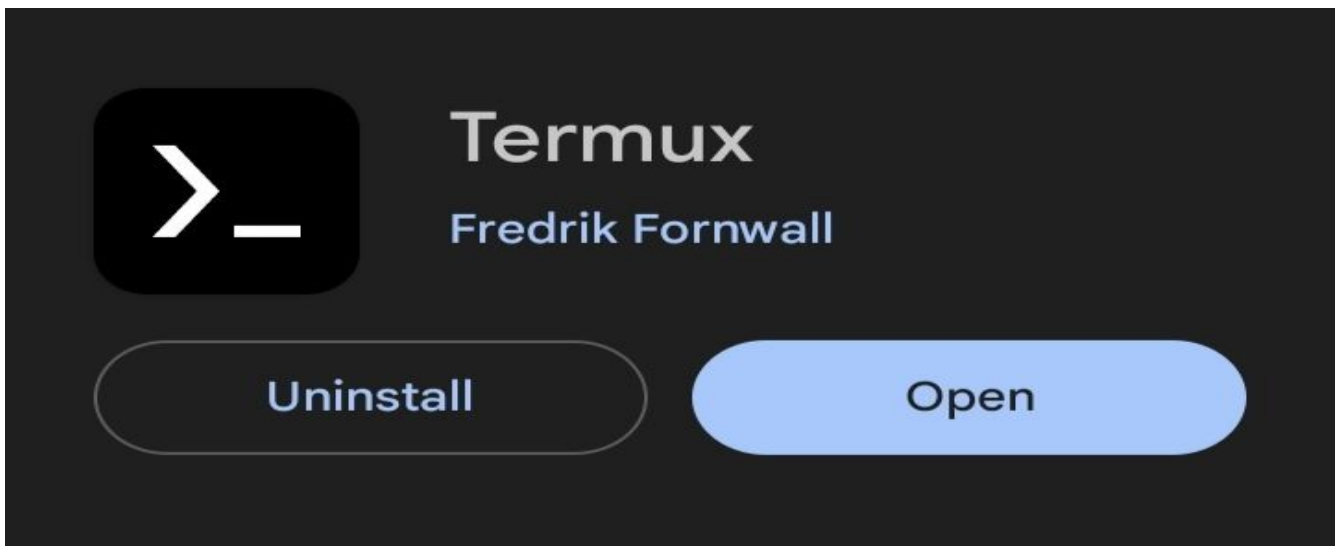
هذه الخطوات تشرح كيفية تثبيت بايثون على أنظمة ويندوز و ماك و لينكس بعد التثبيت، يمكنك بدء استخدام بايثون لتطوير وتشغيل البرامج والسكريبتات .

د.ج- كيفية تثبيت بايثون على اندرويد:

لتثبيت بايثون على جهاز اندرويد، يمكنك اتباع هذه الخطوات مع العلم هذه الطريقة التي اعمل بها انا في نظام الاندرويد.

الخطوة رقم 1: تثبيت تطبيق تيرميكس:

ابحث في متجر جوجل بلاي عن تطبيق تيرميكس وقم بتثبيته على جهازك .



الخطوة رقم 2: فتح تطبيق تيرميكس:

بعد تثبيت تيرميكس، قم بفتح التطبيق

الخطوة رقم 3: تثبيت حزمة بايثون:

استخدم هذا الأمر في تيرميكس لتثبيت حزمة بايثون :

"pkg install python"

```
Welcome to Termux!

Wiki:             https://wiki.termux.com
Community forum:  https://termux.com/community
Gitter chat:      https://gitter.im/termux/termux
IRC channel:      #termux on freenode

Working with packages:

* Search packages:  pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root:            pkg install root-repo
* Unstable:         pkg install unstable-repo
* X11:             pkg install x11-repo

Report issues at https://termux.com/issues

You are likely using a very old version of Termux,
probably installed from the Google Play Store.
There are plans in the near future to remove the
Termux apps from the Play Store so that new users
cannot install them and to **disable** them for
existing users with app updates to prevent the use
of outdated app versions. Instead, you are
encouraged to move to F-Droid or Github sources
(see [1]). You can backup all your current Termux
data before uninstallation and then restore it later
by following instructions in the wiki [2]. Check
the changelog [3] for all the new features and fixes
that you are currently missing. Check [4] for why
this is being done. Note that old versions of
multiple termux apps have privilege escalation
vulnerabilities [5], so it is **highly** advisable
that you move or update to latest versions
available on F-Droid or Github.

[1] https://github.com/termux/termux-app#installation
[2] https://wiki.termux.com/wiki/Backing_up_Termux
[3] https://github.com/termux/termux-app/releases
[4] https://github.com/termux/termux-app#google-play-store-deprecated
[5] https://termux.github.io/general/2022/02/15/termux-apps-vulnerabil
ity-disclosures.html

$ pkg install python3
```

الخطوة رقم 4: التحقق من التثبيت:

بعد الانتهاء من التثبيت، يمكنك التحقق من أن بايثون قد تم تثبيته بنجاح عن طريق إدخال الأمر التالي :

“python –version”

```
Welcome to Termux!

Wiki: https://wiki.termux.com
Community forum: https://termux.com/community
Gitter chat: https://gitter.im/termux/termux
IRC channel: #termux on freenode

Working with packages:

* Search packages: pkg search <query>
* Install a package: pkg install <package>
* Upgrade packages: pkg upgrade

Subscribing to additional repositories:

* Root: pkg install root-repo
* Unstable: pkg install unstable-repo
* X11: pkg install x11-repo

Report issues at https://termux.com/issues

You are likely using a very old version of Termux,
probably installed from the Google Play Store.
There are plans in the near future to remove the
Termux apps from the Play Store so that new users
cannot install them and to **disable** them for
existing users with app updates to prevent the use
of outdated app versions. Instead, you are
encouraged to move to F-Droid or Github sources
(see [1]). You can backup all your current Termux
data before uninstallation and then restore it later
by following instructions in the wiki [2]. Check
the changelog [3] for all the new features and fixes
that you are currently missing. Check [4] for why
this is being done. Note that old versions of
multiple termux apps have privilege escalation
vulnerabilities [5], so it is **highly** advisable
that you move or update to latest versions
available on F-Droid or Github.

[1] https://github.com/termux/termux-app#installation
[2] https://wiki.termux.com/wiki/Backing_up_Termux
[3] https://github.com/termux/termux-app/releases
[4] https://github.com/termux/termux-app#google-play-store-deprecated
[5] https://termux.github.io/general/2022/02/15/termux-apps-vulnerabil
ity-disclosures.html

$ python3 --version
Python 3.11.5
$ █
```

سيقوم هذا الأمر .

بعرض إصدار بايثون الذي تم تثبيته

الخطوة رقم 5: استخدام بايثون

يمكنك الآن استخدام بايثون على جهاز بايثون الخاص بك عبر تيرميكس. يمكنك كتابة الأوامر باستخدام محرر النص المدمج في تيرميكس أو استخدام بايثون مباشرة .

```
$ python3
Python 3.11.5 (main, Aug 30 2023, 00:13:08) [Clang 14.0.7 (https://and
roid.googlesource.com/toolchain/llvm-project 4c603efb0 on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

من الجيد أن تعلم أنه قد يكون هناك طرق أخرى لتثبيت بايثون على اندرويد باستخدام تطبيقات مثل:

QPython ,
Pydroid3,

إذا كنت تفضل واجهة أكثر تكاملاً.

د.د -ملاحظات:

-ملفات بايثون يكون امتداها :

ملف بايثون **py**

ملف بايثون مترجم **pyc**

ملف كائن لبايثون **pyo**

-تستخدم لغة البرمجة البايثون مفسراً فورياً : **interpreter**

و - أساسيات البايثون:

و.أ - التعليقات Comments:

في لغة البرمجة بايثون، يمكنك إضافة تعليقات لشرح الشيفرة المصدرية باستخدام الرمز "#". (هاشتاج).
تعليقات بايثون تُستخدم لتوضيح الأكواد وتسهيل فهمها للمطورين الآخرين أو لنفسك .
إليك بعض الأمثلة على كيفية كتابة التعليقات في بايثون:

-تعليق على سطر واحد:

```
Learn.py x
1  #/usr/bin/python3
2
3  تعليق يشرح مايفعله هذا السطر من الشيفرة#
4  x=5 # نضع القيمة 5 في المتغير x
```

-تعليق متعدد الأسطر:

يمكنك كتابة تعليقات طويلة لشرح تفاصيل الشيفرة المصدرية أو وظائفها

```
• Learn.py x
1  #/usr/bin/python3
2
3  '''
4  هذا تعليق متعدد الأسطر
5  يمكن أن يحتوي على مجموعة من النصوص
6  يمتد عبر عدة أسطر
7  '''
8  x = 10
9  y = 5
10
11  '''
12  يمكنك كتابة تعليقات طويلة لشرح تفاصيل
    الشيفرة المصدرية أو وظائفها
13  '''
```

- تجاهل الشيفرة باستخدام التعليقات:

يُمكن استخدام التعليقات لتجاهل أجزاء من الشيفرة المصدرية أثناء التنفيذ، مما يسمح بتعطيل قطع معينة من الشيفرة مؤقتًا .

```
• Learn.py ×  
  
1  #/usr/bin/python3  
2  
3  # هذا التعليق يجعل السطر التالي غير فعال  
4  # x=5
```

تذكر: أن التعليقات لا تؤثر على تنفيذ البرنامج وتُستخدم فقط للشرح والتوثيق.

و.ب -المتغيرات Variables:

في لغة البرمجة بايثون، المتغيرات تستخدم لتخزين البيانات والقيم والتعامل معها.
هنا شرح مفصل لكيفية تعريف واستخدام المتغيرات:

و.ب.أ-تعريف المتغيرات:

يمكنك تعريف متغير في بايثون بواسطة اسم للمتغير متبوعًا بعلامة
تساوي '=' و ثم القيمة التي ترغب في تخزينها في المتغير.
مثلاً:

```
Learn.py x
1  #/usr/bin/python3
2
3  age = 25
4  y = "amer"
```

في هذا المثال، تم تعريف:
age متغير لتخزين العمر
y متغير لتخزين الاسم

و.ب.ب-نوع المتغير:

بايثون هي لغة متوجهة للأوامر ولا تحتاج إلى تحديد نوع المتغير
يتم تحديد نوع المتغير تلقائياً استناداً إلى القيمة المخزنة فيه .
على سبيل المثال:

```
Learn.py
1  #/usr/bin/python3
2
3  age = 25 #int صحيح نوع
4  y = "amer" #str نص نوع
```

و.ب.ج-أنواع البيانات:

في بايثون، هناك أنواع مختلفة من البيانات التي يمكن استخدامها .
إليك شرح لبعض الأنواع الأساسية :

* أعداد صحيحة :integers:

هذا النوع يُستخدم لتمثيل الأعداد الصحيحة
على سبيل المثال:

```
>>> age = 30
```

* أعداد عائمة :floats:

تُستخدم لتمثيل الأعداد العائمة (أعداد كسرية) .
على سبيل المثال:

```
>>> price = 19.99
```

* سلاسل نصية **strings**:

تُستخدم لتخزين النصوص والمعلومات النصية .
على سبيل المثال :

```
>>> name = "amer"
```

* قوائم **lists**:

تُستخدم لتخزين مجموعة من العناصر بترتيب محدد .
على سبيل المثال:

```
>>> Willaya = ["soukahras", "Annaba", "ain defla"]
```

بالطبع، سأقدم شرحًا للقوائم في لغة البرمجة بايثون:

إنشاء القوائم:

يمكنك إنشاء قائمة في بايثون باستخدام الأقواس الزوجية `[]` ووضع العناصر بينها وفصلها بفواصل.
مثال :

```
>>> my_list = [1, 2, 3, 4, 5]
```

الوصول إلى العناصر:

```
>>> first_element = my_list[0] # يعيد العنصر الأول
```

```
>>> first_element = my_list[2] # يعيد العنصر الثاني
```

تعديل العناصر:

يمكنك تعديل قيم العناصر في القائمة بناءً على الفهرس .

مثلا :

```
>>> my_list[2] = 10 # يجعل العنصر الثالث يحتوي على القيمة 10
```

إضافة وحذف العناصر:

لإضافة عنصر إلى نهاية القائمة، يمكنك استخدام 'append()'

```
>>> my_list.append(6) # يُضيف الرقم 6 إلى نهاية القائمة
```

لحذف عنصر باستخدام قيمته، يمكنك استخدام 'remove()'

```
>>> my_list.remove(3) # يزيل الرقم 3 من القائمة
```

لحذف عنصر باستخدام الفهرس، يمكنك استخدام 'del'

```
>>> del my_list[2] # يزيل العنصر الثالث من القائمة
```

التحقق من وجود عنصر:

يمكنك التحقق من وجود عنصر في القائمة باستخدام 'in':

مثال:

```
if 3 in my_list:  
    print("العنصر 3 موجود في القائمة")
```

طول القائمة :

يمكنك معرفة عدد العناصر في القائمة باستخدام 'len()' :

```
length = len(my_list) # يعيد طول القائمة (عدد العناصر)
```

تعدد الأنواع :

يمكنك تخزين عناصر متعددة الأنواع في القائمة، مثل أرقام وسلاسل وقوائم أخرى وغيرها

التفحص الدوري iteration :

للتفحص عبر عناصر القائمة بسهولة `for` يمكنك استخدام حلقة

مثال :

```
for item in my_list:  
    print(item)
```

نسخ القائمة :

يمكنك نسخ القائمة باستخدام 'copy()'

أو ببساطة بإعادة القائمة كمتغير جديد، تجنب استخدام '=' لنسخ لأنها تشير فقط لنفس القائمة.

تجميع القوائم :

يمكنك دمج قائمتين باستخدام '+', مثل :

```
>>>combined_list = list1 + list2
```


قائمة فارغة :

يمكنك إنشاء قائمة فارغة باستخدام '[]' :

```
>>> empty_list = []
```

هذا شرح للقوائم في بايثون. تعتبر القوائم هيكلًا بيانات قويًا ومرنًا تستخدم على نطاق واسع في البرمجة لتخزين وإدارة البيانات .

*** قواميس dictionaries:**

تُستخدم لتخزين البيانات في صورة أزواج مفتاح-قيمة.
على سبيل المثال:

```
>>> person={"age":"25","name":"tayab"}
```

بالطبع، سأقدم شرحاً للقواميس في لغة البرمجة بايثون.
قواميس في لغة البايثون هي هياكل بيانات تستخدم لتخزين البيانات بشكل مفهوم وفعال يمكنك التفكير في القاموس كمجموعة من الأزواج المفتاح-القيمة حيث يتم تخزين البيانات بواسطة مفتاح فريد يُشير إلى قيمة معينة. الآن دعنا نشرح كيفية استخدام القواميس في بايثون .

إنشاء قاموس :

يمكنك إنشاء قاموس فارغ باستخدام القوسين المعكوفين '{}':

```
>>> my_dict = {}
```

إضافة عناصر إلى القاموس :

يمكنك إضافة عناصر إلى القاموس بواسطة تحديد مفتاح وقيمة :

```
>>> my_dict = {"name": "amer", "age": "19", "country": "Algeria"}
```

الوصول إلى القيم باستخدام المفاتيح :

يمكنك الوصول إلى القيم في القاموس عن طريق ذكر المفتاح :

```
>>> my_dict["name"] = اسم_الشخص
```

تعديل القيم :

يمكنك تعديل القيم في القاموس بنفس الطريقة :

```
>>> my_dict["age"] = 26
```

حذف عناصر من القاموس :

يمكنك حذف عناصر من القاموس باستخدام الدالة :

```
>>> del my_dict["country"]
```

التحقق من وجود مفتاح :

يمكنك التحقق من وجود مفتاح في القاموس باستخدام العبارة التالية :

```
if "name" in my_dict:  
    print("الشخص موجود")
```

الحصول على قائمة بجميع المفاتيح أو القيم :
يمكنك استخدام الدوال 'values()', 'keys()',
للحصول على قائمة بالمفاتيح والقيم على التوالي :

```
>>> keys_list = my_dict.keys()
```

```
>>> values_list = my_dict.values()
```

هذا هو شرح مبسط لكيفية استخدام القواميس في لغة البايثون. تُستخدم القواميس بشكل واسع في البرمجة لتخزين وإدارة البيانات والمعلومات بشكل فعال .

* مجموعات Sets :

تُستخدم لتخزين مجموعة فريدة من العناصر بدون تكرار .
على سبيل المثال :

```
>>> colors = {"red", "green", "white"}
```

في لغة البرمجة بايثون، المجموعات هي هيكل بيانات تستخدم لتخزين مجموعة فريدة من العناصر، وهي مشابهة للقوائم ولكنها تختلف في عدة نقاط .
إليك شرح مفصل للمجموعات في بايثون:

إنشاء مجموعة :

يمكنك إنشاء مجموعة ببساطة باستخدام الأقواس الزوجية '{}' ووضع العناصر بينها .
مثال :

```
>>> my_set = {1, 2, 3, 4, 5}
```

العناصر الفريدة :

المجموعات تحتفظ بالقيم الفريدة فقط. إذا قمت بإضافة عنصر متكرر، سيتم تجاهله .

مثال:

```
>>> my_set = {1, 2, 2, 3, 3, 4}
```

```
>>> print(my_set) ستظهر: #{4,3,2,1}
```

الوصول إلى العناصر :

يمكنك الوصول إلى العناصر في المجموعة باستخدام حلقة 'for'

او بالبحث عن عنصر محدد.

مثال :

```
my_set = {1, 2, 3, 4, 5}
```

```
for num in my_set:  
    print(num)
```

```
#البحث عن عنصر
```

```
if 3 in my_set:  
    print("العنصر موجود في المجموعة")
```

إضافة وحذف العناصر:

يمكنك إضافة عنصر إلى المجموعة باستخدام 'add()'

وحذفه باستخدام 'remove()' or 'discard()'

إذا حاولت حذف عنصر غير موجود 'remove()'.

سيثير استثناء

في حين أن 'discard()'

لن يثير استثناء

مثال :

```
>>> my_set = {1, 2, 3}
```

```
>>> my_set.add(4)
```

```
>>> my_set.remove(2) # يثير استثناء
```

```
>>> my_set.discard(5) # لا يثير استثناء
```

العمليات على المجموعات :

يمكنك أيضًا القيام بعمليات مثل الاتحاد والاقطاع والتمييز بين المجموعات باستخدام الأوبراتورات المناسبة ('|', '&', '-')

أو الوظائف المتاحة مثل 'union()' or 'intersection()' or 'difference()'

مثال :

```
>>> set1 = {1, 2, 3, 4}
```

```
>>> set2 = {3, 4, 5, 6}
```

```
>>> union_set = set1 | set2 # الاتحاد
```

```
>>> intersection_set = set1 & set2 # الاقطاع
```

```
>>> difference_set = set1 - set2 # الفرق
```

هذا هو ملخص مفصل للمجموعات في بايثون، وهي هيكل بيانات مفيدة عند الحاجة إلى تخزين مجموعة من العناصر الفريدة والقيام بعمليات مختلفة عليها .

*** قيم بوليانية : booleans**

تُستخدم لتمثيل القيم الصحيحة أو الخاطئة (True or False) على سبيل المثال :

```
>>> is_raining = True
```

القيم البوليانية في البايثون هي قيم تتخذ إحدى اثنتين القيم: صحيح او خطأ. يُستخدم النوع البولياني للتعبير عن الحالات الثنائية مثل التفعيل والتعطيل، أو الصحة والخطأ، أو وجود شيء معين أو عدم وجوده .

يتم استخدام القيم البوليانية في العديد من السياقات في البرمجة، بما في ذلك الشروط والتحكم في التدفق والتصفية والبحث والكثير من العمليات الأخرى .
هنا مثال بسيط على استخدام القيم البوليانية في البايثون :

```
Learn.py*  
/storage/emulat...  
  
1  #/usr/bin/python3  
2  
3  x = True  
4  y = False  
5  
6  if x:  
7      print("قيمة x صحيحة")  
8  
9  if not y:  
10     print("قيمة y خاطئة")
```

في هذا المثال، تم تعريف متغيرين 'x' and 'y' بقيمتين بوليانييتين مختلفتين. ثم تم استخدامهما في جملة شرط لفحص قيمهما وإجراء إجراءات مختلفة بناءً على قيمهما .

هذه هي بعض الأنواع الأساسية في بايثون، ويمكنك استخدامها لتخزين وتمثيل مختلف أنواع البيانات في اللغة.

ملاحظة: اعلم أنك وجدت لبس في هذا الجزء الأخير وبالتالي يمكنك تخطي هذا الجزء الخاص بأنواع البيانات لأن فيه معلومات لم نتطرق لها مثل الدوال. ويمكنك العودة إليه فيما بعد التطرق إلى المفاهيم الأساسية الشروحات المتقدمة.

و.ب.د- تغيير قيم المتغيرات :

لتغيير قيمة متغير في البايثون، يمكنك ببساطة تعيين قيمة جديدة له
إليك كيفية تغيير قيمة متغير في بايثون

* قم بإنشاء المتغير :

قم بإنشاء متغير وتعيين قيمته الأولى :
على سبيل المثال :

```
>>> my_variable = 5 # قيمة المتغير هي 5
```

* قم بتغيير قيمة المتغير :

يمكنك تغيير قيمة المتغير ببساطة عن طريق إعادة تعيينه
على سبيل المثال

```
>>> my_variable = 10 # تم تغيير قيمة المتغير من 5 الى 10
```

الآن، قيمة المتغير تم تغييرها إلى 10 بدلاً من 5 .

ملاحظة: يمكنك استخدام قيم المتغيرات في العديد من العمليات الحسابية والمنطقية.
مثلاً:

```
>>> sum = my_variable + 5 # النتيجة هي 15 لان تم جمع 5 مع قيمة المتغير الجديدة
```

هذا هو الشرح البسيط لكيفية تغيير قيمة متغير في بايثون، قم بإعادة تعيين المتغير بالقيمة الجديدة التي ترغب فيها، وسيتم تحديث قيمة المتغير.

و.ب.و -الأسماء وقواعد التسمية :

في اللغة البايثون ، هناك بعض القواعد الأساسية لتسمية المتغيرات والمعرفات:

-يجب أن يبدأ اسم المتغير بحرف أو شرطة سفلية (_).

-يمكن أن يحتوي اسم المتغير على أحرف أبجدية صغيرة وأحرف أبجدية كبيرة وأرقام وشرطات سفلية .

-الأسماء حساسة لحالة الأحرف، يعني أن

"myVariable" , "myvariable"

يُعتبران مختلفين .

-يجب تجنب استخدام كلمات محجوزة في البايثون كأسماء متغيرات .

مثل :

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

-إليك بعض الأمثلة على أسماء متغيرات صحيحة في البايثون :

```
>>> my_variable = 42
```

```
>>> name = "John"
```

```
>>> user_age = 30
```

هذه هي بعض قواعد تسمية المتغيرات في البايثون، ويجب أن تلتزم بها عند كتابة البرامج باستخدام هذه اللغة .

هذه هي مفاهيم أساسية حول المتغيرات في بايثون. يمكنك استخدام المتغيرات لتخزين والتعامل مع البيانات في برامجك بسهولة .

تذكر: أن بايثون تستخدم تطبيق قواعد صارمة للتعرف على المتغيرات وأنواع البيانات، لذا يجب أن تكون دقيقًا عند تعريف واستخدام المتغيرات .

و.ب.ه-العمليات على المتغيرات :

بالطبع، سأقدم لك شرحاً مفصلاً حول العمليات على المتغيرات في لغة البرمجة بايثون. عندما نتحدث عن العمليات على المتغيرات، نعني كيفية تنفيذ العمليات الرياضية والمنطقية والمزيد على المتغيرات. دعنا نبدأ :

و.ب.ه.أ-العمليات الرياضية :

في بايثون، يمكنك تنفيذ العمليات الرياضية الأساسية مثل الجمع والطرح والضرب والقسمة باستخدام العمليات الرياضية القياسية. إليك بعض الأمثلة :

```
Learn.py ×
1  #/usr/bin/python3
2
3  a = 5
4  b = 3
5  c = a + b  # الجمع
6  d = a - b  # الطرح
7  e = a * b  # الضرب
8  f = a / b  # القسمة
9
```

و.ب.ه.ب -العمليات المنطقية :

يمكنك أيضًا تنفيذ العمليات المنطقية مثل الشرطيات

(AND ,OR ,NOT)

على المتغيرات. إليك بعض الأمثلة:

```
Learn.py
1  #/usr/bin/python3
2
3  x = True
4  y = False
5  result_and = x and y  # العملية الشرطية AND
6  result_or = x or y    # العملية الشرطية OR
7  result_not_x = not x  # العملية الشرطية NOT
```

و.ب.ه.ج -العمليات على النصوص :

بايثون توفر أيضًا العديد من العمليات التي يمكن استخدامها على النصوص (سلاسل النصوص) .
مثل الجمع والتكرار واستخراج الأحرف والمزيد. إليك بعض الأمثلة:

```
Learn.py
1  #/usr/bin/python3
2
3  text1 = "Hello, "
4  text2 = "world!"
5  concatenated_text = text1 + text2  # الجمع
   للنصوص (الدمج)
6
7  repeated_text = text1 * 3  # التكرار
8
9  character = text1[0]  # استخراج حرف من النص
10
```

و.ب.ه.د-العمليات على القوائم :

يمكنك أيضًا تنفيذ العديد من العمليات على القوائم، مثل إضافة عناصر وإزالتها واستخراج العناصر والمزيد. إليك بعض الأمثلة :

```

1  #/usr/bin/python3
2
3  my_list = [1, 2, 3]
4  my_list.append(4)  # إضافة عنصر إلى القائمة
5  my_list.remove(2)  # إزالة عنصر من القائمة
6  element = my_list[0] # استخراج عنصر من
    القائمة
7
8  length = len(my_list) # الحصول على طول
    القائمة
9

```

هذا ملخص للعمليات الأساسية التي يمكنك تنفيذها على المتغيرات في لغة البرمجة بايثون. باستخدام هذه العمليات، يمكنك أداء مجموعة متنوعة من الأوامر على المتغيرات الرقمية والنصية والقوائم وغيرها لتنفيذ مهام متعددة في البرمجة.

وهذا جدول بدون بعض العمليات الغير مذكورة:

Expression	Description
<code>x == y</code>	x equals y.
<code>x < y</code>	x is less than y.
<code>x > y</code>	x is greater than y.
<code>x >= y</code>	x is greater than or equal to y.
<code>x <= y</code>	x is less than or equal to y.
<code>x != y</code>	x is not equal to y.
<code>x is y</code>	x and y are the same object.
<code>x is not y</code>	x and y are different objects.
<code>x in y</code>	x is a member of the container (e.g., sequence) y.
<code>x not in y</code>	x is not a member of the container (e.g., sequence) y.

و.ج - الإدخال والإخراج : input and output

في بايثون يتيح لك التفاعل مع البرنامج عن طريق إرسال البيانات إليه (الإدخال) ، واستعراض النتائج أو عرضها للمستخدم (الإخراج).
إليك شرح مفصل لكيفية القيام بذلك :

و.ج.أ - الإدخال : input

للقراءة من المستخدم، يُستخدم في بايثون دائمًا الدالة 'input()' لاستدعاء إدخال من المستخدم. يتم قراءة الإدخال كنص (نص).
مثال :

```
Learn.py
1  #!/usr/bin/python3
2
3  name = input("Enter your name : ")
4  # الامر السابق يطلب من المستخدم ادخل اسمه
5
6  print("hello : " + name + "!")
7  # الامر السابق يطبع
8
9
```

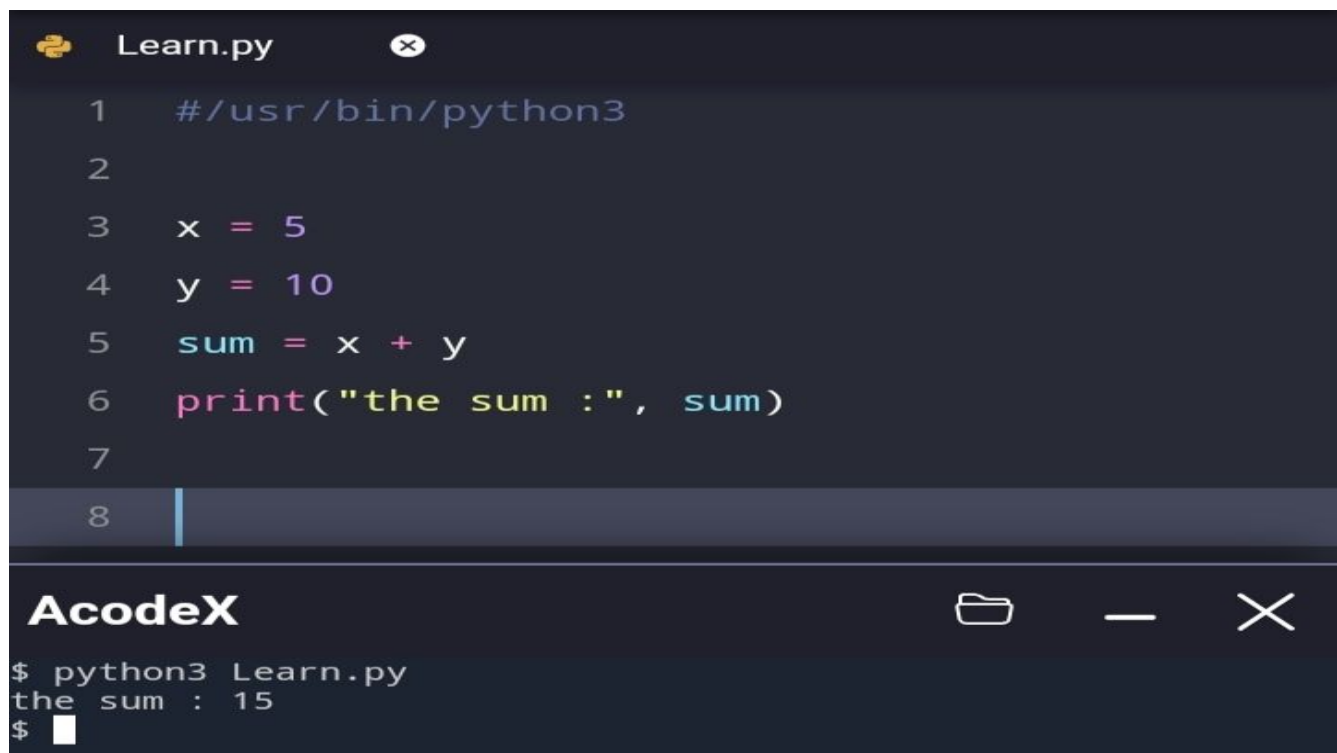
AcodeX

```
$ python3 Learn.py
Enter your name : Amer
hello : Amer!
$
```

في هذا المثال، تطلب الدالة 'input()' من المستخدم إدخال اسمه، وبعد ذلك يتم عرض رسالة ترحيب تحتوي على الاسم الذي قام المستخدم بإدخاله .

و.ج.ب -الإخراج output :
لعرض نتائج أو معلومات للمستخدم، يُستخدم الدالة 'print()'.

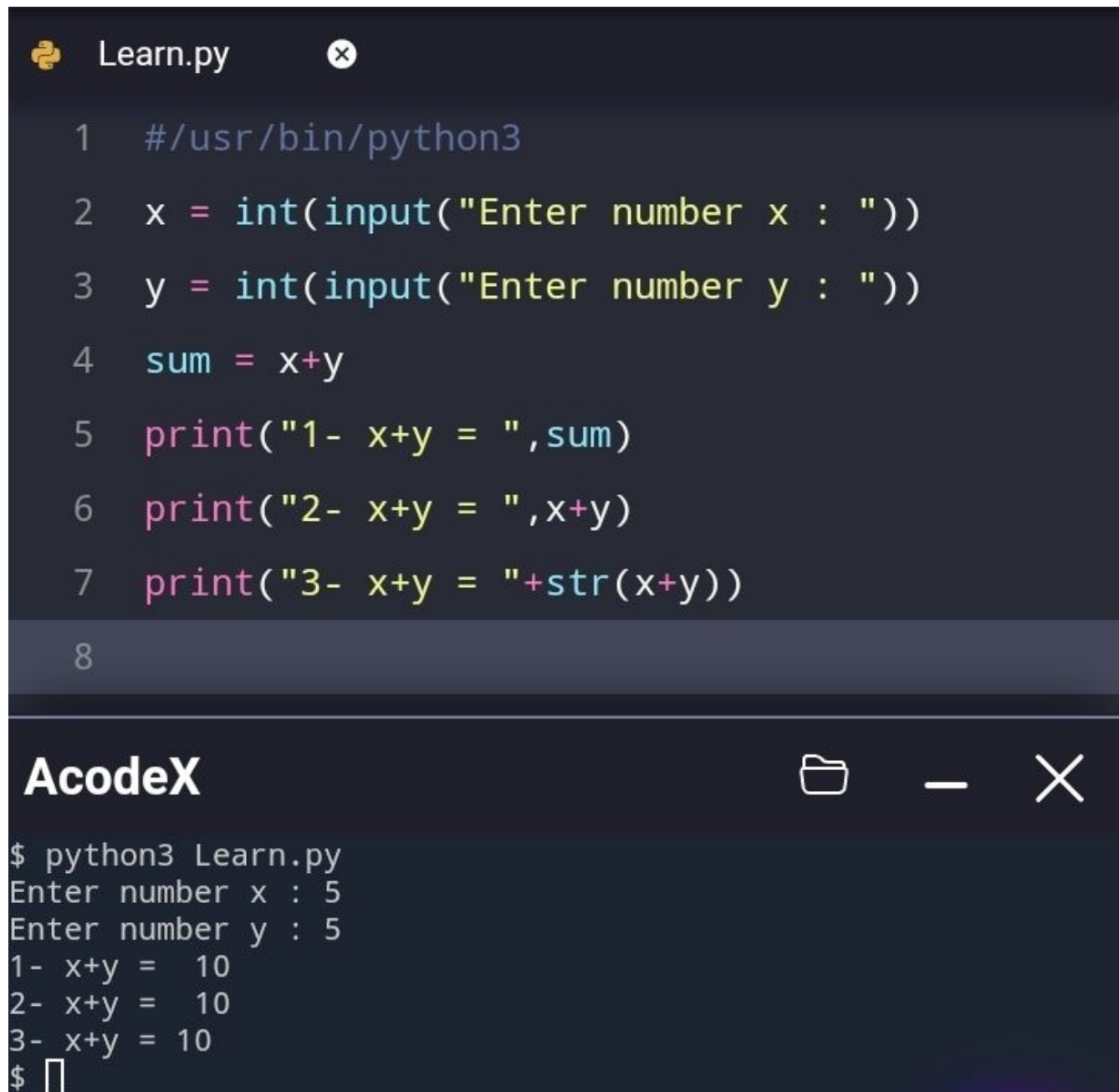
مثال 1:



```
Learn.py
1  #!/usr/bin/python3
2
3  x = 5
4  y = 10
5  sum = x + y
6  print("the sum :", sum)
7
8

AcodeX
$ python3 Learn.py
the sum : 15
$
```


مثال 2:



The image shows a code editor window titled "Learn.py" with a Python icon and a close button. The code is as follows:

```
1  #/usr/bin/python3
2  x = int(input("Enter number x : "))
3  y = int(input("Enter number y : "))
4  sum = x+y
5  print("1- x+y = ",sum)
6  print("2- x+y = ",x+y)
7  print("3- x+y = "+str(x+y))
8
```

Below the code editor is a terminal window titled "AcodeX" with folder, minus, and close icons. It shows the execution of the program:

```
$ python3 Learn.py
Enter number x : 5
Enter number y : 5
1- x+y = 10
2- x+y = 10
3- x+y = 10
$
```

في هذا المثال نفهم ان عمل الدالة

`int()` تحول المتغير الى رقم صحيح

`str()` تحول المتغير الى نص

وهناك دوال اخرى مثل

`float()` تحول المتغير الى رقم عائ

في هذه الامثلة، تم استخدام الدالة `'print()'`
لعرض نص يحتوي على نتيجة الجمع بين `x+y`

و.ج.ج-التنسيق والتحكم في الإخراج :

يمكنك تنسيق الإخراج باستخدام أمور مثل تضمين قيم متغيرة داخل النص واستخدام
العلامات التنصية ("" أو '')

لتحديد النص. يمكنك أيضًا استخدام وظائف تنسيق مثل `'format()'`
للتحكم في كيفية عرض البيانات.

مثال:

```
Learn.py
1  #!/usr/bin/python3
2
3  name = "Amine"
4  age = 12
5  print("name: {},age: {}".format(name, age))
6
7

AcodeX
$ python3 Learn.py
name: Amine,age: 12
$
```

هذا المثال يستخدم 'format()' لوضع قيم المتغيرات في النص.

باستخدام الإدخال والإخراج، يمكنك إنشاء برامج تفاعلية تتفاعل مع المستخدم وتعرض معلومات بشكل فعال .

و.د-تمرينات الفصل :

بالطبع، إليك بعض التمارين البسيطة لفهم أساسيات البرمجة بلغة بايثون:

و.د.أ-التمرين الاول المتغيرات :

قم بإنشاء متغيرات لتخزين الأسماء والأعمار والعناوين. ثم، اطبع هذه المعلومات على الشاشة .

و.د.أ.أ-الحل الاول :

```
Learn.py
1  #!/usr/bin/python3
2
3  name = input("enter your name : ")
4  age  = input("enter your age : ")
5  adr  = input("enter your address : ")
6  print("\n hello Mr,"+str(name)+"\n Age : "
        +str(age)+"\n adr : "+str(adr))

AcodeX
$ python3 Learn.py
enter your name : Amer
enter your age : 21
enter your address : Age1

hello Mr,Amer
Age : 21
adr : Age1
$
```

و.د.أ.ب -الحل الثاني :

```

1  #/usr/bin/python3
2  info = {}
3  info["name"] = input("enter your name : ")
4  info["age"] = input("enter your age : ")
5  info["adr"] = input("enter your address : ")
6  print("\n hello Mr,"+str(info["name"])+"\n
    Age : "+str(info["age"])+"\n adr : "+str
    (info["adr"]))
7

```

AcodeX

```

$ python Learn.py
enter your name : Amine
enter your age : 11
enter your address : alge

hello Mr,Amine
Age : 11
adr : alge
$ 

```

و.د.ب -التمرين الثاني العمليات الحسابية :

أكتب برنامجًا يقوم بإجراء عمليات حسابية بسيطة مثل الجمع والطرح والضرب والقسمة .

و.د.ب.أ -الحل :

```

1  #/usr/bin/python3
2
3  x = float(input("Enter number x : "))
4  y = float(input("Enter number y : "))
5
6  print("\n x+y = {}\n x-y = {}\n x*y = {}\n
    x/y = {}".format(x+y,x-y,x*y,x/y))
7

```

AcodeX

```

$ python Learn.py
Enter number x : 10
Enter number y : 5

x+y = 15.0
x-y = 5.0
x*y = 50.0
x/y = 2.0
$ 

```


ر-التحكم في التدفق :

بالطبع، سأقدم لك شرحاً مفصلاً لكيفية التحكم في تدفق البرامج في لغة البرمجة بايثون. التحكم في التدفق يشمل استخدام هياكل الشرط وحلقات التكرار. دعونا نبدأ :

ر.أ-العبارات الشرطية Conditional Statements :

في بايثون، هناك هياكل شرطية تسمح لك بتنفيذ الأوامر استناداً إلى الشروط المحددة . الهياكل الشرطية الرئيسية هي 'if', 'elif', 'else'

ر.أ.أ-العبرة الشرطية 'if' :

تُستخدم لتنفيذ كود إذا تحقق شرط معين. مثل :

```
Learn.py
1  #/usr/bin/python3
2
3  age = 18
4  if age >= 18:
5      print("you are an adult")

AcodeX
$ python3 Learn.py
you are an adult
$
```

ر.أ.ب - العبارة الشرطية 'elif' :

تُستخدم لإضافة شروط إضافية بعد 'if' في حال عدم تحقق الشرط الأول . مثل :

```
Learn.py
1  #/usr/bin/python3
2  grade = 14
3  if grade >= 15:
4      print("excellent")
5  elif grade >= 10:
6      print("successful")
7  else:
8      print("Failed")

AcodeX
$ python3 Learn.py
successful
$
```

ر.أ.ج - العبارة الشرطية 'else' :

تُستخدم لتنفيذ كود إذا لم تتحقق أي من الشروط السابقة . مثل :

```
Learn.py
1  #/usr/bin/python3
2
3  number = 7
4  if number % 2 == 0:
5      print("The number is even")
6  else:
7      print("The number is odd")

AcodeX
$ python3 Learn.py
The number is odd
$
```

ر.ب -حلقات التكرار Loops :

حلقات التكرار تُستخدم لتنفيذ كود معين مرارًا وتكرارًا .
في بايثون، هناك حلقتين رئيسيتين 'while' and 'for' :

ر.ب.أ -الحلقة 'for' :

تُستخدم لتكرار كود على عناصر في تسلسل (مثل قائمة) :

```
Learn.py
1  #!/usr/bin/python3
2
3  fruits = ["apple", "banana", "an orange"]
4  for fruit in fruits:
5      print(fruit)
6
```

```
AcodeX
$ python3 Learn.py
apple
banana
an orange
$
```

ر.ب.ب -الحلقة 'while' :

تُستخدم لتكرار كود حتى يتحقق شرط معين. مثل :

```
Learn.py
1  #!/usr/bin/python3
2
3  count = 0
4  while count < 5:
5      print(count)
6      count += 1
7
```

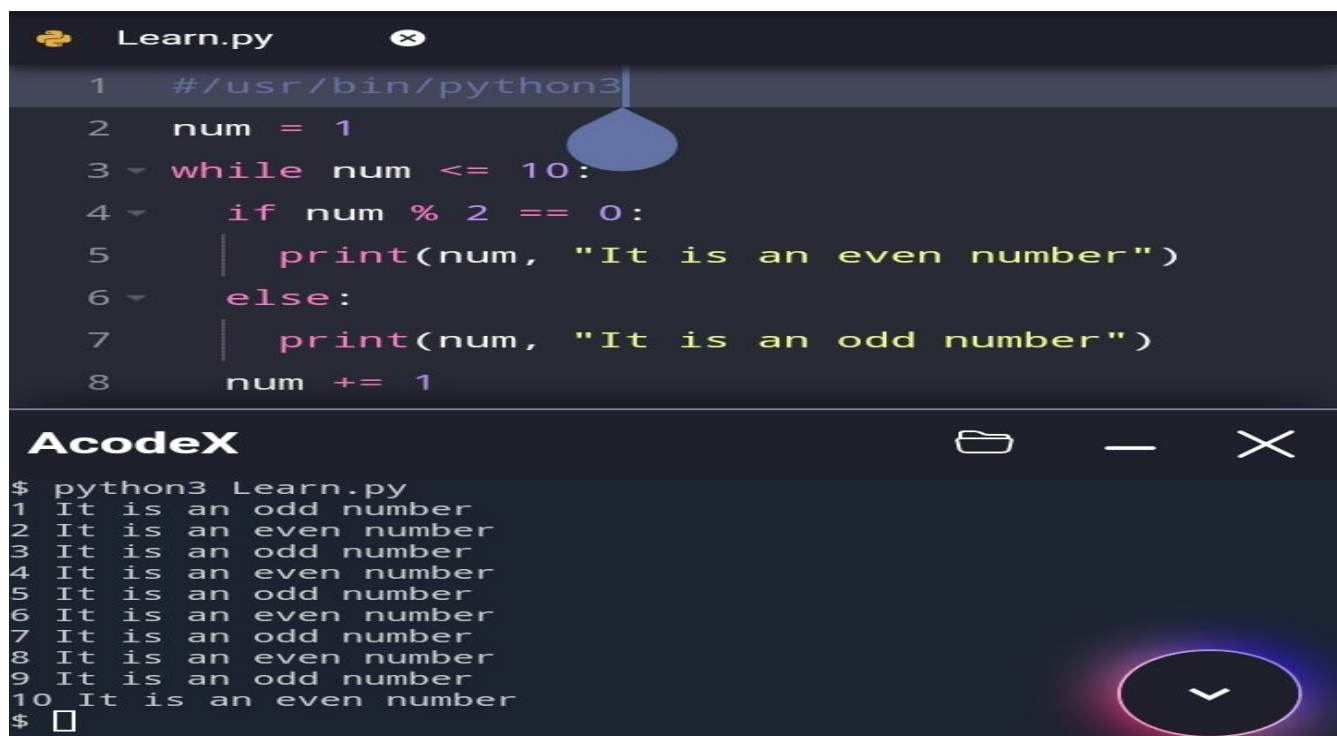
```
AcodeX
$ python3 Learn.py
0
1
2
3
4
$
```


رج-اندماج الشرط والتكرار :

يمكن دمج الشرط والتكرار في البرمجة باستخدام حلقات مع شروط

مثلا، يمكنك استخدام حلقة 'while'

لتكرار الكود حتى يتحقق شرط معين.



```
Learn.py
1  #!/usr/bin/python3
2  num = 1
3  while num <= 10:
4      if num % 2 == 0:
5          print(num, "It is an even number")
6      else:
7          print(num, "It is an odd number")
8      num += 1
```

AcodeX

```
$ python3 Learn.py
1 It is an odd number
2 It is an even number
3 It is an odd number
4 It is an even number
5 It is an odd number
6 It is an even number
7 It is an odd number
8 It is an even number
9 It is an odd number
10 It is an even number
$
```

هذا هو شرح مفصل لكيفية التحكم في تدفق البرامج في لغة البرمجة بايثون باستخدام البيانات الشرطية وحلقات التكرار. تذكر أن هذه الهياكل أساسية في البرمجة وتساعدك على تنظيم تنفيذ الأوامر وفقاً للشروط والتكرارات المطلوبة .

ر.د-تمرينات الفصل :

بالطبع! هنا بعض التمارين حول التحكم في التدفق في بايثون:

ر.د.أ-التمرين الاول هيكل التحكم الشرطي :

اكتب برنامج يطلب من المستخدم إدخال رقم ويقوم بفحص ما إذا كان الرقم موجبًا أم سالبًا .

ر.د.أ-الحل :

```
Learn.py
1  #/usr/bin/python3
2
3  number = float(input("Enter number : "))
4  if number > 0:
5      print("positive")
6  elif number < 0:
7      print("nigative")
8  else:
9      print("number = 0")
10

AcodeX
$ python3 Learn.py
Enter number : 5
positive
$
```

ر.د.ب -التمرين الثاني الحلقات :

كتابة برنامج يستخدم حلقة لطباعة الأعداد الزوجية من 1 إلى 10 .

ر.د.ب -الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  i = 1
4  while i<=10:
5      if i % 2 == 0:
6          print(i)
7          i=i+1
8
AcodeX
$ python3 Learn.py
2
4
6
8
10
$
```


س- الوظائف Function :

بالطبع! لنبدأ بشرح مفصل للوظائف في لغة البرمجة بايثون .

الوظائف هي كتلة من الشيفرة كود بلوك تقوم بأداء مهمة محددة. في بايثون، يمكنك إنشاء واستخدام الوظائف باستمرار لتنظيم وتبسيط البرنامج الخاص بك . إليك بعض المفاهيم الأساسية حول الوظائف في بايثون :

س.أ- تعريف الوظيفة :

لتعريف وظيفة في بايثون، يمكنك استخدام الكلمة الرئيسية 'def' متبوعة باسم الوظيفة وقوسين () .

مثال :

```
def hello():  
    print("Hello!")
```

س.ب- معلمات الوظيفة :

يمكنك تمرير معلمات إلى الوظائف لتمكينها من استخدام قيم معينة أثناء تنفيذها .

مثال :

```
def hello(name):  
    print("Hello ,Mr " +name)
```

س.ج -إرجاع القيم:

يمكن للوظائف في بايثون إرجاع قيم باستخدام الكلمة الرئيسية
يمكنك استخدام قيمة العودة لاستخدام نتائج الوظيفة في أماكن أخرى في البرنامج .

مثال :

```
def sum(x,y):  
    su = x+y  
    return su
```

س.د -استدعاء الوظائف :

لاستدعاء وتنفيذ وظيفة في بايثون، يجب استخدام اسم الوظيفة متبوعًا بقوسين () .

مثال :

```
hello("Amer")  
Result = sum(5, 3)
```

هذه هي مفاهيم أساسية حول الوظائف في بايثون.
يمكنك إنشاء واستخدام الوظائف. لتنظيم وتبسيط برنامجك وجعله أكثر قابلية للصيانة وفهمًا .

س.و-تمرينات الفصل :

بالطبع، إليك بعض التمارين حول الوظائف في لغة البرمجة بايثون.

س.و.أ-التمرين الاول عرف واستدعي وظيفة بسيطة.

س.و.أ-الحل:

```
Learn.py
1  #!/usr/bin/python3
2
3  def hello(name):
4      |  print("Hello ,Mr ", name)
5
6  hello("amer")

AcodeX
$ python3 Learn.py
Hello ,Mr  amer
$
```

س.و.ب - التمرين الثاني اكتب وظيفة تقوم بحساب مجموع قائمة من الأرقام.

س.و.ب - الحل:

```
Learn.py

2
3 def sum_list(list):
4     sum = 0
5     for num in list:
6         sum += num
7     return sum
8
9 list1 = [1, 2, 3, 4, 5]
10 sum1 = sum_list(list1)
11 print("The Sum : ", sum1)
12
```

AcodeX

```
$ python3 Learn.py
The Sum : 15
$
```


س.و.ج - التمرين الثالث انشاء واستدعي وظيفة تأخذ معاملات اختيارية .

س.و.ج - الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  def hello2(name,sal="Hello, "):
4      print(sal,name)
5
6      hello2("adel")
7      hello2("Amine", "hi")
8

AcodeX
$ python3 Learn.py
Hello,  adel
hi Amine
$
```

هذه أمثلة بسيطة على كيفية استخدام وتعريف الوظائف في بايثون.
يمكنك تجربة هذه التمارين وتعديلها حسب احتياجاتك لفهم أفضل لكيفية عمل الوظائف في بايثون.

س.س - امثلة عن بعض الوظائف الاساسية :
بالطبع! إليك بعض الدوال الأساسية في لغة البرمجة بايثون:

``print()``: تُستخدم لطباعة النصوص والمعلومات على الشاشة .

``input()``: تُستخدم لاستقبال إدخال من المستخدم من خلال واجهة سطر الأوامر .

``len()``: تُستخدم للعثور على طول (عدد العناصر) في مصفوفة أو سلسلة نصية .

``range()``: تُستخدم لإنشاء مجموعة من الأرقام التسلسلية .

``type()``: تُستخدم لمعرفة نوع المتغير أو القيمة .

``int()``, ``float()``, ``str()``:

تُستخدم لتحويل القيم بين أنواع مختلفة (عدد صحيح، عدد عشري، سلسلة نصية) .

``list()``, ``tuple()``, ``dict()``, ``set()``:

تُستخدم لإنشاء هياكل بيانات مثل قوائم، أزواج مرتبة، قواميس، ومجموعات على التوالي.

ش-الملفات Files :

في بايثون، الملفات تُستخدم للقراءة والكتابة على القرص الصلب
يمكنك فتح ملف باستخدام الدالة 'open()' ،
واستخدام مجموعة من الأساليب للتعامل معه. إليك شرح مفصل لكيفية التعامل
الملفات والاستدعاءات :

ش.أ-فتح الملف :

'open()'

يمكنك فتح ملف باستخدام الدالة
مع اسم الملف والوضع (القراءة، الكتابة، إلخ). مثال:

```
file = open("myfile.txt", "r") # لفتح ملف للقراءة
```

ش.ب-قراءة المحتوى :

`read()``

`readline()``

يمكنك استخدام الدالة :
لقراءة محتوى الملف بأكمله أو الدالة
لقراءة سطر واحد في كل مرة

مثال :

```
content = file.read() # لقراءة الملف بأكمله
```

```
line = file.readline() # لقراءة سطر واحد
```

ش.ج -الكتابة في الملف :

يمكنك استخدام الوضع "w" ،

للكتابة في الملف. استخدم الدالة 'write()' .

مثال لكتابة البيانات في الملف :

```
file = open("myfile.txt", "w")
```

```
file.write("Hello, World!")
```

ش.د -إغلاق الملف :

بمجرد الانتهاء من استخدام الملف، يجب إغلاقه باستخدام الدالة 'close()'

للحفاظ على استقرار البرنامج .

```
file.close()
```

ش.و -استخدام السياقات :

يمكن استخدام السياقات ('with')

لضمان إغلاق الملف تلقائياً عند الانتهاء من العمل معه، وهذا يعزز من أمان البرنامج .

```
with open("myfile.txt", "r") as file:
```

```
    content = file.read()
```

الملف سيتم إغلاقه تلقائياً هنا بغض النظر عن ما إذا تمت العملية بنجاح أم لا

هذا هو ملخص مفصل لكيفية التعامل مع الملفات في بايثون واستدعاءاتها .
تأكد دائماً من التحقق من وجود الملفات قبل الوصول إليها ومعالجة الاستثناءات
إذا لزم الأمر لتجنب الأخطاء .

ش.س -تمرينات الفصل :

بالطبع، إليك بعض التمارين حول كيفية التعامل مع الملفات في بايثون.

ش.س.أ -التمرين الاول : اقرء ملف نصي واعرض محتواه .

ش.س.أ -الحل :

```
untitled.txt  Learn.py
1  #!/usr/bin/python3
2
3  # افتح الملف للقراءة
4  with open('sample.txt', 'r') as file:
5      content = file.read()
6      print(content)
```

ش.س.ب -تمرين الثاني : اكتب في ملف نصي .

ش.س.ب -الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  # افتح ملفًا للكتابة واكتب نصًا
4  with open('output.txt', 'w') as file:
5      text_to_write = input("Enter text : ")
6      file.write(text_to_write)
```

ش.س.ج -تمرين الثالث : انسخ ملف .

ش.س.ج -الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  import shutil
4  # a 'source.txt' To 'destination.txt'
5  shutil.copy('source.txt', 'destination.txt')
6
```

ش.س.د-تمرين الرابع : قراءة وكتابة ملف بتنسيق CSV .

ش.س.د-الحل :

```
Learn.py
1  #/usr/bin/python3
2
3  import csv
4  # قراءة ملف CSV
5  with open('data.csv', 'r') as file:
6      reader = csv.reader(file)
7      for row in reader:
8          print(row)
9
10 # كتابة إلى ملف CSV
11 data_to_write = ['name', 'age'],
12                 ['bendada', 22], ['abdlmalk', 21]]
13 with open('new_data.csv', 'w', newline='')
14     as file:
15         writer = csv.writer(file)
16         writer.writerows(data_to_write)
```


ش.س.و -تمرين الخامس: عين أذونات الملف .

ش.س.و -الحل :

```
Learn.py
1  #/usr/bin/python3
2
3  import os
4
5  # تعيين أذونات الملف ليصبح قابلاً للقراءة
   #والكتابة فقط للمالك
6  os.chmod('file.txt', 0o600)
7
8
```

ش.س.س -تمرين السادس: احذف ملف .

ش.س.س -الحل :

```
Learn.py
1  #/usr/bin/python3
2
3  import os
4
5  # حذف ملف إذا كان موجودًا
6  if os.path.exists('file_to_delete.txt'):
7      os.remove('file_to_delete.txt')
8
```

تأكد من تعديل اسم الملف والمسار وفقًا للملفات التي تملكها على نظامك .
هذه التمارين تساعدك على فهم كيفية التعامل مع الملفات في بايثون وتطبيق الأمور الأساسية .

ملاحظة : شرح المكتبات في الفصل القادم .

ص - المكتبات Modules :

بالطبع! في لغة البرمجة بايثون، المكتبات تمثل مجموعة من الأوامر والوظائف التي تم تجميعها مسبقاً لتنفيذ مهام محددة. يمكنك استخدام هذه المكتبات لتوسيع إمكانيات لغة البرمجة بايثون وتبسيط عمليات البرمجة .
إليك شرح مفصل لكيفية استخدام المكتبات في بايثون :

ص.أ - استيراد المكتبات :

لاستخدام مكتبة معينة في بايثون، يجب عليك أولاً استيرادها باستخدام الأمر `'import'`.
على سبيل المثال، إذا أردت استخدام مكتبة `'math'` لأغراض حسابية، يمكنك كتابة:

```
import math
```

ص.ب - استخدام وظائف المكتبة :

بعد استيراد المكتبة، يمكنك الوصول إلى وظائفها واستخدامها. على سبيل المثال، لاستخدام وظيفة `'sqrt'` في مكتبة `'math'` لحساب جذر تربيعي:

```
x = math.sqrt(16)
```

في هذا المثال، تم استخدام وظيفة `'sqrt'` من المكتبة `'math'` لحساب جذر تربيعي للعدد 16 وتخزين النتيجة في المتغير `'x'`

ص. ج- المكتبات المدمجة والمكتبات الخارجية :

بايثون يحتوي على مكتبات مدمجة مثل : 'math', 'random'

وغيرها التي يمكنك استخدامها من دون الحاجة إلى تثبيتها بشكل منفصل .
ولكن هناك أيضًا مكتبات خارجية يمكنك تنزيلها واستخدامها حسب احتياجاتك، مثل :

`numpy`

للعمل مع البيانات العلمية

`requests`

للتفاعل مع واجهات ويب

ص. د - تثبيت المكتبات الخارجية :

لتثبيت مكتبة خارجية، يمكنك استخدام مدير الحزم الخاص بايثون مثل 'pip' .
على سبيل المثال، لتثبيت مكتبة 'requests' .
يمكنك تنفيذ الأمر التالي في سطر الأوامر :

```
$ pip install requests
```

بعد التثبيت، يمكنك استخدام هذه المكتبة في مشروعك .

ص. و - الاستفادة من الوثائق :

دائمًا يجب عليك الرجوع إلى وثائق المكتبة التي تستخدمها لفهم كيفية استخدامها بشكل صحيح .
معظم المكتبات تأتي مع وثائق مفصلة تشرح وظائفها وكيفية استخدامها .

باستخدام المكتبات في بايثون، يمكنك تسريع عملية التطوير وجعل برمجتك أكثر فعالية من خلال إعادة استخدام الأكواد والوظائف الموجودة مسبقًا .

ص.س-تمرينات الفصل :

بالطبع، إليك تمارين حول استخدام المكتبات في بايثون مع حلولها:

ص.س.أ-تمرين الاول : استخدام المكتبة الرياضية **Math Library** :

لحساب مساحة الدائرة .

ص.س.أ-الحل :

```
Learn.py ×
1  #/usr/bin/python3
2
3  import math
4  # حساب مساحة دائرة
5  radius = 5
6  area = math.pi * math.pow(radius, 2)
7  print("Circle area :", area)
```

ص.س.ب -تمرين الثاني : استخدام المكتبة العشوائية Random Library :
قيام قرعة عشوائية لاختيار 10 اشخاص.

ص.س.ب -الحل :

```
Learn.py ×  
1  #/usr/bin/python3  
2  
3  import random  
4  
5  # قرعة عشوائية لاختيار 10 أشخاص  
6  people = ["p1", "p2", "p3", "p4", "p5",  
            "p6", "p7", "p8", "p9", "p10"]  
7  winners = random.sample(people, 10)  
8  print("the people's winners :", winners)
```

ص.س.ج - تمرين الثالث استخدام مكتبة الوقت **Time Library** :
لقياس مدى الوقت لتنفيذ عملية معينة.

ص.س.ج -الحل :

```
Learn.py ×  
1  #/usr/bin/python3  
2  
3  import time  
4  
5  # قياس مدى الوقت لتنفيذ عملية معينة  
6  start_time = time.time()  
7  # العملية التي تأخذ وقتًا  
8  end_time = time.time()  
9  execution_time = end_time - start_time  
10 print("The amount of time it takes : ",  
        execution_time, "seconds")
```

ص.س.د - التمرين الرابع استخدام مكتبة الطلبات **Requests Library** :
لجلب بيانات من موقع ويب.

ص.س.د - الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  import requests
4
5  # جلب بيانات من موقع ويب
6  url = "https://www.example.com"
7  response = requests.get(url)
8  print("page content : ")
9  print(response.text)
```

ص.س.و - تمرين الخامس استخدام مكتبة التصوير **Pillow Library** :
لفتح صورة وتطبيق تأثيرات عليها.

ص.س.و - الحل :

```
Learn.py
1  #!/usr/bin/python3
2
3  from PIL import Image, ImageFilter
4
5  # فتح صورة وتطبيق تأثيرات عليها
6  image = Image.open("example.jpg")
7  blurred_image = image.filter(ImageFilter
8  .BLUR)
9  blurred_image.show()
```


تمثل هذه الأمثلة تمارين بسيطة تساعدك على فهم كيفية استخدام المكتبات في بايثون. يمكنك تجربة هذه الأمثلة ومن ثم استكشاف المزيد من المكتبات والتمارين حسب احتياجاتك.

ص.ص - بعض مكتبات البايثون:

بالطبع اليك بعض مكتبات البايثون :

ص.ص.أ - مكتبة 'os' :

في بايثون هي واحدة من المكتبات الأساسية التي تُستخدم للتفاعل مع نظام التشغيل وإدارة الملفات والمجلدات. تتيح لك هذه المكتبة الوصول إلى مجموعة متنوعة من الوظائف والخصائص للقيام بمهام مثل إنشاء وحذف الملفات، والتلاعب بالمسارات، والتحقق من وجود ملفات ومجلدات، وأكثر من ذلك .

إليك شرحًا كاملاً لمكتبة 'os' :

ص.ص.أ.أ - استيراد المكتبة :

قبل استخدام مكتبة 'os'

يجب عليك استيرادها في برنامج بايثون الخاص بك كما يلي :

```
import os
```

ص.ص.أ.ب - العمليات الأساسية :

- `os.getcwd()`: تُستخدم للحصول على المجلد الحالي (دليل العمل الحالي)
- `os.chdir(path)`: تغيير دليل العمل الحالي إلى المسار المحدد 'path'
- `os.listdir(path)`: استعراض محتوى مجلد معين
- `os.mkdir(path)`: إنشاء مجلد جديد
- `os.rmdir(path)`: حذف مجلد فارغ
- `os.remove(path)`: حذف ملف

ص.ص.أ.ج - التفاعل مع المسارات :

- `os.path.join()` : دمج مسارات لإنشاء مسار كامل
- `os.path.exists(path)` : التحقق من وجود ملف أو مجلد
- `os.path.isfile(path)` : التحقق مما إذا كان المسار يشير إلى ملف
- `os.path.isdir(path)` : التحقق مما إذا كان المسار يشير إلى مجلد

ص.ص.أ.د - التحقق من الصلاحيات :

- `os.access(path, mode)` : التحقق من صلاحيات الوصول إلى ملف أو مجلد

ص.ص.أ.و - العمليات على المسارات :

- `os.path.basename(path)` : استخراج اسم الملف أو المجلد من المسار
- `os.path.dirname(path)` : استخراج المجلد الرئيسي للمسار

ص.ص.أ.س - التعامل مع متغيرات البيئة :

- `os.environ` : الوصول إلى متغيرات البيئة

ص.ص.أ.ص - العمليات على الملفات :

- `os.rename(src, dst)` : إعادة تسمية ملف أو مجلد
- `os.replace(src, dst)` : استبدال ملف أو مجلد بآخر
- `os.stat(path)` : الحصول على معلومات مفصلة حول ملف أو مجلد

ص.ص.أ.ض - التفاعل مع النظام :

- `os.system(command)` : تنفيذ أمر النظام

ص.ص.أ.هـ -الفصل بين نظم التشغيل :

اسم نظام التشغيل : `os.name` -

(Windows لنظام التشغيل 'nt' و Unix/Linux لنظام التشغيل 'posix')

هذه هي بعض الوظائف والأساليب الأساسية التي تقدمها مكتبة 'os' ،
تتيح لك هذه المكتبة التحكم الكامل في التفاعل مع نظام التشغيل والملفات في برامجك
باستخدام بايثون .

ص.ص.ب -مكتبة 'time' :

في لغة البرمجة بايثون توفر وسائل للتعامل مع الوقت والتوقيت في البرامج .
إليك شرحاً مفصلاً لأهم الوظائف والمفاهيم المتعلقة بمكتبة 'time' :

1. `time.time()`:

- تُستخدم هذه الوظيفة للحصول على عدد الثواني المنصرفة منذ منتصف
الليل (1 يناير 1970)، وهو معروف باسم "تصريح يونكس"
- يمكن استخدام هذا التصريح لقياس الزمن وحساب فارق الوقت بين نقطتين
مختلفتين في الزمن .

2. `time.sleep(seconds)`:

- تُستخدم لإيقاف تنفيذ البرنامج لعدد محدد من الثواني المحددة في المعامل '`seconds`'
- يُفيد ذلك في تعليق تنفيذ البرنامج لفترة زمنية معينة، مثل الانتظار لحين اكتمال
عملية معينة .

3. `time.localtime([seconds])``:

- تقوم بتحويل تصريح يونكس إلى هيكل (`struct`)
- يحتوي على معلومات الوقت المفصلة مثل السنة، والشهر، واليوم، والساعة، والدقيقة، والثانية.
 - يمكن تمرير `'seconds'` ،
 - للحصول على هيكل الوقت لتصريح معين، أو يمكن تركه فارغًا للحصول على الوقت الحالي .

4. `time.strftime(format, [struct_time])``:

- تُستخدم لتنسيق الوقت كنص وفقًا لتنسيق معين محدد بواسطة المعامل `'format'` .
- يمكن استخدام هيكل الوقت الذي تم إنشاؤه بواسطة `'time.localtime()'` كمدخل لهذه الوظيفة .

5. `time.clock()`:`

- تُستخدم لقياس الزمن المنقضي منذ بدء تنفيذ البرنامج على مستوى الوحدة المركزية (CPU) .

6. `time.perf_counter()`` و time.process_time()`:`

- تُستخدم لقياس الأداء والزمن المنقضي بدقة عالية على مستوى النظام .
- تقيس الوقت الذي مر منذ بدء تشغيل البرنامج بدقة عالية وقد تتأثر بتوقيت النظام .

7. `time.monotonic()`:`

- تستخدم لقياس الوقت بدقة عالية مع تجنب تأثيرات التغييرات في توقيت النظام .

هذه هي بعض الوظائف والمفاهيم الأساسية في مكتبة `'time'` في بايثون. تستخدم هذه الوظائف على نطاق واسع في تطبيقات مختلفة مثل إجراء عمليات زمنية دقيقة، وقياس الأداء، وتنسيق الوقت .

ص.ص.ج - مكتبة 'JSON' :

في بايثون هي واحدة من المكتبات المدمجة التي تستخدم لتحليل وإنشاء بيانات

JSON (JavaScript Object Notation) ،

وهي تستخدم بشكل شائع لتبادل البيانات بين التطبيقات،

إليك شرح كامل لمكتبة 'JSON' :

ص.ص.ج.أ - استيراد المكتبة :

يمكنك استيراد مكتبة 'JSON'

في بايثون باستخدام الأمر التالي :

```
import json
```

ص.ص.ج.ج - تحويل بايثون إلى 'JSON'

يمكنك استخدام 'json.dumps()'

لتحويل كائن بايثون إلى نص 'JSON'

```
data = {"name": "John", "age": 30, "city": "New York"}  
json_data = json.dumps(data)
```

ص.ص.ج.د - تحويل 'JSON To Python' :

يمكنك استخدام 'json.loads()' .

لتحويل نص إلى كائن بايثون :

```
json_data = '{"name": "John", "age": 30, "city": "New York"}'  
data = json.loads(json_data)
```

ص.ص.ج.و - التعامل مع الملفات :

يمكنك قراءة ملف 'JSON'

وتحويله إلى كائن بايثون باستخدام :

```
with open('data.json', 'r') as file:  
    data = json.load(file)
```

ويمكنك كتابة كائن بايثون إلى ملف 'JSON' :

باستخدام :

```
data = {"name": "John", "age": 30, "city": "New York"}  
with open('data.json', 'w') as file:  
    json.dump(data, file)
```

ص.ص.ج.س - التعامل مع هياكل البيانات المعقدة :

يمكن لمكتبة 'JSON'

التعامل مع هياكل البيانات المعقدة، بما في ذلك القوائم والكائنات المتداخلة .

ص.ص.ج.ص - التحقق من صحة JSON

يمكنك استخدام :

`json.JSONEncoder` لتخصيص كيفية ترميز الكائنات JSON

`json.JSONDecoder` لتخصيص فك ترميز JSON

هذا هو ملخص لاستخدام مكتبة JSON .

تسمح لك هذه المكتبة بتبسيط عمليات تحويل البيانات بين التطبيقات التي تستخدم تنسيق كوسيلة لتبادل البيانات .

ص.ص.د - مكتبة 'random' :

هي مكتبة في لغة برمجة بايثون تستخدم لإنشاء أرقام عشوائية .

تُستخدم هذه المكتبة على نطاق واسع في البرمجة لإجراء عمليات توليد أرقام عشوائية لأغراض مختلفة مثل الاختبارات والمحاكاة والألعاب .

إليك شرح مفصل لاستخداماتها وبعض الأمثلة على كيفية استخدامها :

ص.ص.د.أ - استيراد المكتبة :

يجب أن تبدأ بالاستيراد لاستخدام المكتبة في برنامجك :

```
import random
```

ص.ص.د.ب - توليد عدد عشوائي بين قيمتين :

يمكنك استخدام 'random.randint(a,b)'

لإنشاء عدد صحيح عشوائي بين 'a'، 'b'

```
random_number = random.randint(1, 10)
```

ص.ص.د.ج - توليد عدد عشوائي فاصل عشري :
يمكنك استخدام `random.uniform(a, b)`
لإنشاء عدد فاصل عشري عشوائي بين 'a' و 'b' :

```
random_float = random.uniform(0.0, 1.0)
```

ص.ص.د.د - اختيار عنصر عشوائي من قائمة :
يمكنك استخدام `random.choice(seq)`
لاختيار عنصر عشوائي من قائمة 'seq' :

```
fruits = ["apple", "banana", "cherry"]  
random_fruit = random.choice(fruits)
```

ص.ص.د.و - اختلاط العناصر في القائمة :
يمكنك استخدام `random.shuffle(seq)`
لاختلاط العناصر في قائمة 'seq'
بشكل عشوائي :

```
random.shuffle(fruits)
```


ص.ص.د.س -توليد تسلسل عشوائي :

يمكنك استخدام `random.sample(seq, k)`

لإنشاء قائمة من 'k'

عناصر عشوائية من قائمة 'seq'

دون تكرار :

```
random_sample = random.sample(fruits, 2)
```

ص.ص.د.ص -عملية توليد عشوائية أكثر تخصيصًا :

يمكنك استخدام وظائف متقدمة أخرى في المكتبة للتحكم في عمليات التوليد بشكل أدق ،
مثل توليد أرقام عشوائية تتبع توزيع معين أو تخصيص توزيع الاحتمالات .

هذه مجرد بعض الاستخدامات الشائعة لمكتبة 'random'

تساعدك هذه المكتبة في إنشاء عناصر عشوائية في برامجك بسهولة وفعالية .

ص.ص.و -مكتبة 'requests' :

في بايثون هي مكتبة مفيدة لإجراء طلبات HTTP

إلى موارد عبر الإنترنت مثل صفحات الويب وواجهات برمجة التطبيقات (APIs).

إليك شرح مفصل لكيفية استخدامها :

ص.ص.و.أ - قبل استخدام المكتبة :

يجب تثبيتها إذا لم تكن مثبتة بالفعل يمكنك فعل ذلك باستخدام pip

عبر الأمر التالي :

```
pip install requests
```

ص.ص.و.ب - بعد تثبيت المكتبة، استيرادها في برنامجك :

```
import requests
```

ص.ص.و.ج - إجراء طلب GET :

```
response = requests.get('https://example.com')
```

هنا تم إجراء طلب GET .

إلى 'https://example.com' .

وحفظ الاستجابة في متغير 'response'

ص.ص.و.د - فحص حالة الاستجابة واستخراج المحتوى :

```
if response.status_code == 200:  
    print(response.text)
```

هذا يتحقق من أن الاستجابة كانت ناجحة (رمز الحالة 200) ويقوم بطباعة محتوى الصفحة .

يمكنك أيضًا إضافة معلمات إضافية إلى طلبك مثل الرؤوس أو البيانات 'headers' or 'data' على النحو التالي :

```
headers = {'User-Agent': 'my-app'}  
response = requests.get('https://example.com', headers=headers,  
params={'param1': 'value1'})
```

هنا تم إضافة رأس المستخدم ('User-Agent')

وبيانات الاستعلام (query parameters)

إلى الطلب .

ص.ص.و.و -إجراء طلب POST :

```
data = {'key1': 'value1', 'key2': 'value2'}  
response = requests.post('https://example.com/post-endpoint', data=data)
```

هذا يقوم بإجراء طلب **POST** وإرسال البيانات المحددة .

هذا هو ملخص مفصل لكيفية استخدام مكتبة 'requests' في بايثون لإجراء طلبات **HTTP** .
يمكنك استخدام هذه الأساسيات للتفاعل مع موارد الإنترنت بسهولة .

ص.ص.ص.ص -مكتبة 'BeautifulSoup' :

في بايثون هي أداة قوية تُستخدم لتحليل واستخراج المعلومات من صفحات الويب بشكل سهل وفعال .
تستخدم عادة مع مكتبة 'requests'
لجلب صفحات الويب ومن ثم تحليلها باستخدام **BeautifulSoup**
إليك شرح مفصل لكيفية استخدامها :

ص.ص.ص.ص.أ_تثبيت BeautifulSoup :

يمكنك تثبيت المكتبة باستخدام **pip**
بالأمر التالي :

```
pip install beautifulsoup4
```

ص.ص.ص.ب -استيراد المكتبة BeautifulSoup :

بعد التثبيت، يمكنك استيراد BeautifulSoup :

في برنامجك كما يلي :

```
from bs4 import BeautifulSoup
```

ص.ص.ص.ج -جلب صفحة الويب :

يمكنك استخدام مكتبة requests

لجلب صفحة الويب التي تريد تحليلها، مثل :

```
import requests
```

```
url = 'https://example.com'
```

```
response = requests.get(url)
```

ص.ص.ص.د -إنشاء كائن BeautifulSoup :

بعد جلب صفحة الويب، يمكنك إنشاء كائن BeautifulSoup

وتحليل صفحة HTML

بالشكل التالي :

```
soup = BeautifulSoup(response.text, 'html.parser')
```

ص.ص.ص.و -استخراج المعلومات :

يمكنك استخدام BeautifulSoup

لاستخراج المعلومات من صفحة HTML

باستخدام وسوم ال HTML

والسمات. على سبيل المثال لاستخراج جميع الروابط في الصفحة :

```
links = soup.find_all('a')
```

```
for link in links:
```

```
    print(link.get('href'))
```

ص.ص.ص.س -التنقل بين العناصر :

يمكنك الوصول إلى العناصر داخل صفحة HTML باستخدام تركيبات يايثون مثل :

```
title = soup.title.string
```

هذا ملخص مفصل لكيفية استخدام BeautifulSoup

في يايثون لتحليل واستخراج المعلومات من صفحات الويب .

يمكنك تخصيص استخداماتها بناءً على احتياجاتك الخاصة لجمع البيانات من الويب .

ص.ص.ص.ض -مكتبة 'Selenium' :

هي أداة تُستخدم لأتمتة عمليات التصفح على الويب باستخدام لغة البرمجة يايثون .

تُستخدم عادة لاختبار تطبيقات الويب أو لأتمتة المهام الروتينية على مواقع الويب.

إليك شرح مبسط لكيفية استخدام 'Selenium' :

ص.ص.ص.ض.أ -تثبيت 'Selenium' :

قبل استخدام Selenium

في يايثون، يجب تثبيتها بواسطة pip

يمكنك استخدام الأمر التالي في الطرفية

```
pip install selenium
```

ص.ص.ض.ب -تحميل مشغل متصفح :

Selenium تتعامل مع المتصفحات المختلفة مثل كروم فايرفوكس لاستخدامها مع متصفح معين، تحتاج إلى تنزيل مشغل المتصفح المناسب . على سبيل المثال إذا كنت تستخدم جوجل كروم، فستحتاج إلى تنزيل مشغل **Chrome Webdriver**

ص.ص.ض.ج -كتابة الشيفرة

الآن يمكنك كتابة الشيفرة باستخدام **Selenium** لتحكم في المتصفح والقيام بالعمليات المطلوبة. إليك مثال بسيط على كيفية فتح متصفح كروم وزيارة موقع ويب :

```
from selenium import webdriver
```

```
# تكوين مشغل المتصفح
```

```
driver = webdriver.Chrome(executable_path='مسار_Chrome_WebDriver.exe')
```

```
# فتح موقع ويب
```

```
driver.get('https://www.example.com')
```

```
# أدخل نص في مربع البحث إذا كان ذلك مطلوبًا
```

```
search_box = driver.find_element_by_name('search')
```

```
search_box.send_keys('بحث')
```

```
# إرسال النموذج أو أداء الإجراءات الأخرى على الموقع
```

```
# إغلاق المتصفح عند الانتهاء
```

```
driver.quit()
```

هذا مثال بسيط، ويمكنك توسيع استخدام **Selenium** لأتمتة العديد من الأمور على الويب، مثل النقر على روابط ، واستخراج البيانات، والتفاعل مع الصفحات والنماذج، وأكثر من ذلك . تحتاج إلى استشارة وثائق **Selenium** والمصادر الإضافية لمزيد من التفاصيل والأمثلة .

ص.ص.ط - مكتبة 'hashlib' :
في بايثون هي مكتبة تستخدم لحساب القيم الهاش (التجزئة) للبيانات .
يمكن استخدامها لإنشاء قيم هاش لأي نص أو بيانات تريدها .
هذا هو شرح مفصل لكيفية استخدامها :

ص.ص.ط.أ - استيراد المكتبة :
قبل استخدام 'hashlib'
يجب عليك استيرادها في برنامج بايثون الخاص بك باستخدام الأمر التالي :

```
import hashlib
```

ص.ص.ط.ب - إنشاء كائن هاش :
يجب عليك إنشاء كائن من فئة هاش معينة مثل (SHA-256, MD5)
باستخدام دالة 'hashlib.new()' على سبيل المثال :

```
sha256_hash = hashlib.new('sha256')
```

ص.ص.ط.ج - تمرير البيانات للكائن :

بعد إنشاء الكائن هاش، يمكنك تمرير البيانات التي ترغب في حساب هاشها باستخدام دالة `'update()'` مثل هذا :

```
data = "Hello, world!".encode('utf-8') # تحويل النص إلى بايتات
sha256_hash.update(data)
```

ص.ص.ط.د - استخراج القيمة الهاش :

بعد تمرير البيانات، يمكنك استخراج القيمة الهاش باستخدام دالة `'hexdigest()'` :

```
hash_value = sha256_hash.hexdigest()
print("SHA-256 : ", hash_value)
```

ستحصل على سلسلة نصية تمثل القيمة الهاش للبيانات التي قمت بتمريرها .

ص.ص.ط.د - تنظيف الكائن هاش (اختياري) :

بعد الانتهاء من استخدام الكائن هاش، يمكنك تنظيفه باستخدام دالة `'digest()'` لاستعادة الموارد :

```
sha256_hash.digest()
```

هذا هو ملخص مفصل لاستخدام مكتبة `'hashlib'`

في بايثون لحساب القيم الهاش. تذكر أنه يمكنك اختيار أي خوارزمية هاش تريدها مثل:

(SHA-256, MD5, SHA-1)

واستخدامها بناءً على متطلبات مشروعك .

ص.ص.ظ - مكتبة 'socket' :

في بايثون تمكّنك من إنشاء وإدارة اتصالات الشبكة عبر مختلف البروتوكولات مثل UDP, TCP إليك شرح مفصل لكيفية استخدام هذه المكتبة:

ص.ص.ظ.أ - استيراد المكتبة :

يجب أن تبدأ بالاستيراد المكتبة في البرنامج الخاص بك :

```
import socket
```

ص.ص.ظ.ب - إنشاء مأخذ 'Socket' :

يمكنك إنشاء مأخذ باستخدام الدالة 'socket()'

وتحديد نوع البروتوكول (TCP OR UDP)

ونوع العنوان (IPv4 OR IPv6)

مثال:

```
# إنشاء مأخذ TCP
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
# إنشاء مأخذ UDP
```

```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

ص.ص.ظ.ج - ربط المأخذ بعنوان ومنفذ (Binding) :

إذا كنت ترغب في استقبال الاتصالات على الجهاز الحالي ،
يجب عليك ربط المأخذ بعنوان ايبي ومنفذ محدد.

مثلا :

```
server_address = ('127.0.0.1', 12345) # ومنفذ IPv4 عنوان  
server_socket.bind(server_address)
```

ص.ص.ظ.د - الاستماع للاتصالات (Listening) :

في حالة الاتصالات TCP ،
يمكنك استخدام الدالة 'listen()'
للبدء في الاستماع للاتصالات الواردة :

```
server_socket.listen(5) # استقبال حتى 5 اتصالات متزامنة
```

ص.ص.ظ.و - قبول الاتصال (Accepting) :

لاستقبال اتصال TCP
وقبوله، يمكنك استخدام الدالة 'accept()'
ستعود هذه الدالة بمأخذ جديد يُستخدم للتواصل مع العميل :

```
client_socket, client_address = server_socket.accept()
```

ص.ص.ظ.س - إرسال واستقبال البيانات :

يمكنك استخدام مأخذ العميل والخادم لإرسال واستقبال البيانات بي الدوال 'send()' or 'recv()' .

إرسال بيانات من الخادم إلى العميل

```
client_socket.send(b'Hello, client!')
```

استقبال بيانات من العميل إلى الخادم

```
data = client_socket.recv(1024)
```

ص.ص.ظ.ص - إغلاق المأخذ :

عند الانتهاء من التواصل، يجب إغلاق المأخذ بواسطة الدالة 'close()' .

```
client_socket.close()
```

```
server_socket.close()
```

هذه هي الخطوات الأساسية لاستخدام مكتبة 'socket'

في بايثون لإنشاء تطبيقات شبكية. تذكر أنه يجب التحقق من معالجة الاستثناءات وإدارة الأخطاء أثناء تنفيذ هذه العمليات لضمان استقرار التطبيقات الخاصة بك .

ص.ص.ع - مكتبة NumPy :

هي مكتبة مفيدة للغاية في لغة البرمجة بايثون والتي تستخدم للعمل مع البيانات المتعددة الأبعاد والعمليات الرياضية عليها .
إليك شرح مفصل لمكتبة NumPy :

ص.ص.ع.أ - التثبيت :

يمكنك تثبيت المكتبة باستخدام **pip** من خلال الأمر التالي في الطرفية :

```
pip install numpy
```

ص.ص.ع.ب - استيراد NumPy :

بعد التثبيت، يجب عليك استيراد NumPy في برنامجك باستخدام الأمر التالي :

```
import numpy as np
```

ص.ص.ع.ج - إنشاء Arrays :

المكتبة تقدم هيكل البيانات المعروف باسم "أراي"، وهو عبارة عن ترتيب من العناصر من نفس النوع .
يمكنك إنشاء أراي بسهولة باستخدام 'np.array()'

```
my_array = np.array([1, 2, 3, 4, 5])
```

ص.ص.ع.د-العمليات الرياضية :

NumPy تتيح لك أداء العمليات الرياضية على الأري بسهولة. مثلاً، يمكنك جمع عناصر الأري معًا .

```
result = my_array + 2
```

ويمكنك أيضًا أداء العمليات الرياضية الأخرى مثل الضرب والقسمة والطرح .

ص.ص.ع.و-أبعاد متعددة

NumPy تدعم الأري ذات الأبعاد المتعددة. يمكنك إنشاء مصفوفات ثنائية الأبعاد وثلاثية الأبعاد وأكثر مثال على مصفوفة ثنائية الأبعاد :

```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

ص.ص.ع.س-وظائف الخاصة NumPy :

NumPy توفر العديد من الوظائف الخاصة للتعامل مع البيانات، مثل :

```
`np.sum()`
```

لحساب مجموع العناصر

```
`np.mean()`
```

لحساب المتوسط

```
`np.max()`, `np.min()`
```

للعثور على القيمة القصوى والصغرى، والمزيد

ص.ص.ع.ص-الفهرسة والاستعراض :

يمكنك الوصول إلى عناصر الأري باستخدام الفهرسة والاستعراض. على سبيل المثال :

```
element = my_array[2] # الوصول إلى العنصر الثالث في الأري
```

```
subset = matrix[0:2, 1:3] # استخراج جزء من المصفوفة
```

ص.ص.ع.ض-البث **Broadcasting** :

نيمباي تدعم البث، مما يعني أنه يمكنك أداء العمليات الرياضية على أراي ذات أشكال مختلفة بدون الحاجة إلى كتابة حلقات .

ص.ص.ع.ط-التعامل مع الملفات :

يمكنك استخدام نيمباي لقراءة وكتابة البيانات من وإلى ملفات مثل

‘TXT’, ‘CSV’

وغيرها

ص.ص.ع.ظ-مجتمع نشط :

NumPy هي جزء من البيئة البيانية البيانات الكبيرة في بايثون وتتميز بمجتمع نشط من المستخدمين والمساهمين، مما يعني وجود الكثير من الموارد والمساعدة المتاحة عبر الإنترنت .

هذا ملخص مبسط لمكتبة نيمباي في بايثون. تُستخدم نيمباي بشكل واسع في مجالات مثل العلوم البيانية والتعلم الآلي ومعالجة الصور والعديد من التطبيقات الأخرى حيث تحتاج إلى التعامل مع البيانات المتعددة الأبعاد والعمليات الرياضية .

ض -التطبيقات العملية :

هناك العديد من التطبيقات العملية للبرمجة باستخدام لغة البايثون.
إليك بعض الأمثلة على التطبيقات الشائعة للبايثون في مجموعة متنوعة من المجالات:

ض.أ -تطوير الويب (Web Development) :

إنشاء مواقع الويب باستخدام إطار عمل Flask او Django
تطبيقات الويب باستخدام إطار العمل Flask
وإضافاته المتعددة .

ض.ب -العلم بيانات وتحليل البيانات (Data Science and Analysis) :

تحليل البيانات باستخدام مكتبات مثل Numpy و Pandas .
تدريب وتقييم نماذج التعلم الآلي باستخدام مكتبات مثل PyTorch و TensorFlow .

ض.ج -التطوير المحمول والسطحي (Mobile and Desktop Development) :

تطوير تطبيقات الهاتف المحمول باستخدام إطار العمل Kivy .
أو تطبيقات سطح المكتب باستخدام PyQt .

ض.د -التصور والرسوم البيانية (Visualization and Graphics) :

إنشاء الرسوم البيانية والتصور باستخدام مكتبات مثل Seaborn و Matplotlib

ض.و -تطوير الألعاب (Game Development) :

تطوير ألعاب الكمبيوتر باستخدام مكتبات مثل Pygame .

ض.ص -التطوير الآلي والمهام المكررة (Automation and Scripting) :

كتابة البرامج النصية لتنفيذ المهام المكررة أو الأتمتة باستخدام بايثون .

ض.ض-العمليات الشبكية (Networking) :

بناء تطبيقات الشبكة باستخدام مكتبات مثل Requests و Scapy .

ض.ط-معالجة اللغة الطبيعية (Natural Language Processing) .

تطوير تطبيقات معالجة اللغة الطبيعية باستخدام مكتبات مثل spaCy و NLTK .

ض.ظ-إدارة البيانات وقواعد البيانات (Data Management and Databases) :

الاتصال بقواعد البيانات وإدارتها باستخدام مكتبات مثل Django ORM و SQLAlchemy .

ض.ع-الآتمة في إدارة النظام (System Administration and Automation) :

تنفيذ مهام إدارة النظام مثل النسخ الاحتياطي وجدولة المهام باستخدام بايثون .

ض.غ-الذكاء الاصطناعي وتعلم الآلة (Artificial Intelligence and Machine Learning) :

تطوير وتدريب نماذج الذكاء الاصطناعي باستخدام مكتبات مثل TensorFlow و Keras .

ض.ه-تطوير تطبيقات الذكاء الصناعي (Artificial Intelligence Applications) :

تطوير تطبيقات تستند إلى تقنيات الذكاء الصناعي مثل تصنيف الصور والاستدلال والتعرف على الكلام .

هذه مجرد نماذج صغيرة من التطبيقات التي يمكن تحقيقها باستخدام بايثون .
تعتبر لغة البايثون متعددة الاستخدامات ومناسبة للعديد من المجالات المختلفة ،
مما يجعلها واحدة من اللغات البرمجية الأكثر شعبية واستخدامًا حاليًا .

ط-مشاريع بسيطة :

بالطبع، إليك أمثلة على مشاريع بسيطة في بايثون مع الأكواد:

ط.أ-بناء آلة حاسبة بسيطة :

```
def add(x, y):  
    return x + y
```

```
def subtract(x, y):  
    return x - y
```

```
def multiply(x, y):  
    return x * y
```

```
def divide(x, y):  
    if y == 0:  
        return "لا يمكن قسم على الصفر"  
    return x / y
```

```
while True:  
    print("خيارات:")  
    print("1. جمع")  
    print("2. طرح")  
    print("3. ضرب")  
    print("4. قسم")  
    print("5. انتهاء")
```

```
choice = input("اختر العملية (1/2/3/4/5) : ")
```

```
if choice == '5':  
    break
```

```
num1 = float(input("أدخل العدد الأول: "))  
num2 = float(input("أدخل العدد الثاني: "))
```

```
if choice == '1':  
    print("الجواب:", add(num1, num2))  
elif choice == '2':  
    print("الجواب:", subtract(num1, num2))  
elif choice == '3':  
    print("الجواب:", multiply(num1, num2))  
elif choice == '4':  
    print("الجواب:", divide(num1, num2))  
else:  
    print("خيار غير صحيح")
```

ط.ب -مولد كلمات سر :

```
import random
import string
```

```
def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password
```

```
password_length = int(input("أدخل طول كلمة المرور: "))
```

```
print("كلمة المرور المولدة:", generate_password(password_length))
```

ط.ج -لعبة تخمين الرقم :

```
import random

secret_number = random.randint(1, 100)
attempts = 0

print("لعبة تخمين الرقم بين 1 و 100")

while True:
    guess = int(input("اختر رقماً: "))
    attempts += 1

    if guess == secret_number:
        print(f"محاولات {attempts} بعد {secret_number} أحسنت! لقد تخمنت الرقم")
        break
    elif guess < secret_number:
        print("الرقم الذي اخترته أصغر من الرقم السري")
    else:
        print("الرقم الذي اخترته أكبر من الرقم السري")
```

قم بزيارة **GitHub**

واختر مشروعًا مفتوح المصدر يثير اهتمامك واتبع الإرشادات في مستودع المشروع لتشغيله.

هذه أمثلة بسيطة تظهر كيفية بدء مشاريع في بايثون. يمكنك توسيع هذه المشاريع وإضافة المزيد من الميزات حسب رغبتك ومستوى مهاراتك في البرمجة .

ظ - البرمجة الكائنية :

في بايثون تعتمد على مفهوم الكائنات (Objects)، وهي واحدة من مفاهيم البرمجة الشيئية (Object-Oriented Programming – OOP). فيما يلي شرح مفصل للبرمجة الكائنية في بايثون :

ظ.أ - الكائنات (Objects) :

في بايثون، يعتبر كل شيء كائنًا. الأشياء الحقيقية مثل القواميس والقوائم والأعداد هي أمثلة على الكائنات .

ظ.ب - الفصل (Classes) :

الفصل هو قالب أو نموذج يستخدم لإنشاء كائنات. يمكنك تصويره على أنه مثل المخطط الهندسي لبناء.

ظ.ج - إنشاء كائن :

لإنشاء كائن من الفصل، تحتاج إلى استخدام اسم الفصل مع إضافة أقواس مستديرة. مثلاً

```
class Person:  
    pass
```

```
person1 = Person()  
person2 = Person()
```

ظ.د - السمات (Attributes) :

السمات هي المتغيرات التي تمتلكها الكائنات. يمكنك تعريف السمات داخل الفصل وتهيئتها لكل كائن على حده. مثلاً :

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

ظ.و - الأساليب (Methods) :

الأساليب هي الوظائف التي يمكن أن تعمل على الكائنات. يمكنك تعريف الأساليب داخل الفصل. مثلاً :

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.")
```


ظ.ص -الاستخدام :

يمكنك إنشاء كائنات من الفصل واستدعاء الأساليب والوصول إلى السمات كما يلي :

```
person1 = Person("John", 30)
person2 = Person("Alice", 25)
```

```
person1.greet() # سيطبع: Hello, my name is John and I am 30 years old.
```

```
person2.greet() # سيطبع: Hello, my name is Alice and I am 25 years old.
```

هذا هو ملخص مفصل للبرمجة الكائنية في بايثون. تسمح لك هذه المفاهيم بتنظيم وتجميع البيانات والوظائف ذات الصلة في هياكل منطقية وتسهيل الصيانة وإعادة الاستخدام في البرمجة .

ظ.ض -تمارينات الفصل :

ظ.ض.أ -التمرين الاول : تطبيق مدير المهام (To-Do List)

ظ.ض.أ -الحل :

يمكنك بناء تطبيقًا يسمح للمستخدم بإضافة وإدارة المهام. هذا مثال بسيط :

```
class Task:
    def __init__(self, name, description):
        self.name = name
        self.description = description
        self.completed = False

class ToDoList:
    def __init__(self):
        self.tasks = []

    def add_task(self, task):
        self.tasks.append(task)

    def view_tasks(self):
        for index, task in enumerate(self.tasks, start=1):
            print(f"{index}. {task.name} - {'منجزة' if task.completed else 'غير منجزة'}")

    def mark_complete(self, task_index):
        if 1 <= task_index <= len(self.tasks):
            self.tasks[task_index - 1].completed = True
            print(f"تم وضع علامة على المهمة '{self.tasks[task_index - 1].name}' كمنجزة.")
        else:
            print("الفهرس غير صحيح.")

todo_list = ToDoList()

while True:
    print("خيارات:")
    print("1. إضافة مهمة")
    print("2. عرض المهام")
    print("3. وضع علامة على مهمة كمنجزة")
    print("4. انتهاء")

    choice = input("اختر الخيار: ")

    if choice == '1':
        name = input("أدخل اسم المهمة: ")
        description = input("أدخل وصف المهمة: ")
        task = Task(name, description)
        todo_list.add_task(task)
        print("تمت إضافة المهمة بنجاح.")
    elif choice == '2':
        todo_list.view_tasks()
    elif choice == '3':
        task_index = int(input("أدخل رقم المهمة المنجزة: "))
        todo_list.mark_complete(task_index)
    elif choice == '4':
        break
    else:
        print("خيار غير صحيح. الرجاء المحاولة مرة أخرى")
```


ع -كود شامل :

بالطبع، يمكنني توفير كود شامل للبايثون الذي يغطي العديد من المفاهيم الأساسية .
هذا مثال على كود بايثون يحتوي على تعليقات لشرح الأمور بشكل مفصل :

Variables and Data Types

```
name = "John"  
age = 30  
height = 6.2  
is_student = True
```

Control Flow - if, elif, else statements

```
if age < 18:  
    print("You are a minor.")  
elif 18 <= age < 65:  
    print("You are an adult.")  
else:  
    print("You are a senior citizen.")
```

Loops - for loop and while loop

```
for i in range(5):  
    print("Iteration", i)
```

```
count = 0  
while count < 3:  
    print("Count is", count)  
    count += 1
```

Lists and List Comprehensions

```
numbers = [1, 2, 3, 4, 5]  
squared_numbers = [x ** 2 for x in numbers]
```

اكمال الكود

Functions

```
def greet_person(person_name):  
    return "Hello, " + person_name + "!"
```

```
greeting = greet_person(name)
```

Dictionaries

```
person_info = {  
    "name": name,  
    "age": age,  
    "is_student": is_student  
}
```

Classes and Object-Oriented Programming

```
class Animal:  
    def __init__(self, species):  
        self.species = species
```

```
    def make_sound(self):  
        pass
```

```
class Dog(Animal):  
    def make_sound(self):  
        return "Woof!"
```

Exception Handling

```
try:  
    result = 10 / 0  
except ZeroDivisionError:  
    result = "Division by zero is not allowed."
```

اكمال الكود

File Handling

```
with open("sample.txt", "w") as file:  
    file.write("This is a sample file.")
```

```
with open("sample.txt", "r") as file:  
    file_content = file.read()
```

Libraries

```
import math  
square_root = math.sqrt(16)
```

Print results

```
print("Name:", name)  
print("Greeting:", greeting)  
print("Squared Numbers:", squared_numbers)  
print("Person Info:", person_info)  
print("Dog Sound:", Dog("Labrador").make_sound())  
print("Division Result:", result)  
print("File Content:", file_content)  
print("Square Root:", square_root)
```

هذا الكود يغطي مجموعة متنوعة من المفاهيم الأساسية في بايثون، بدءاً من التعريفات الأساسية والمتغيرات وصولاً إلى الهياكل التحكم والقوائم والمكتبات القياسية والمدخلات من المستخدم. يمكنك تنفيذ هذا الكود وتعديله حسب احتياجاتك.

غ - النهاية :

' شكراً لك على اهتمامك وجهدك في قراءة كتاب تعلم البرمجة. نأمل أن وجدت هذا الكتاب مفيداً ومثرياً في رحلتك لاكتساب مهارات البرمجة. البرمجة هي مجال مثير ومهم، وأنا ممتن لاختيارك هذا الكتاب لتعزيز معرفتك .

إذا كانت لديك أي استفسارات أو اقتراحات، فلا تتردد في الاتصال بنا. نحن هنا لمساعدتك في كل مرحلة من رحلتك التعليمية. نتمنى لك النجاح والتقدم المستمر في عالم البرمجة .

مرة أخرى، شكراً لك على وقتك وجهدك، وأتمنى لك كل التوفيق في المستقبل.'

بأطيب التحيات

[عرايبيه عامر]

الفهرس

أ - التمهيد	04
ب - مقدمة	06
ج - البايثون؟ ماهي لغة البايثون؟	08
د - تثبيت البايثون	10
د.أ - كيفية تثبيت البايثون على الويندوز	10
د.ج - كيفية تثبيت البايثون على الاندرويد	12
ر - اساسيات البايثون	16
ر.أ - التعليقات	16
ر.ب - المتغيرات	19
ر.ج - الادخال والاخراج	38
و - التحكم في التدفق	45
و.أ - العبارات الشرطية	46

47.....	و.ب -الحلقات التكرارية
52.....	س -الوضائف
59.....	ش -الملفات
66.....	ص -المكتبات
72.....	ص.ص -بعض مكتبات البايثون
72.....	ص.ص.ب -مكتبة os
74.....	ص.ص.ب -مكتبة time
76.....	ص.ص.ج -مكتبة JSON
78.....	ص.ص.د -مكتبة random
80.....	ص.ص.و -مكتبة requests
82.....	ص.ص.ص -مكتبة BeutifulSoup
84.....	ص.ص.خ -مكتبة selenium
86.....	ص.ص.ط -مكتبة hashlib

88.....	ص.ص.ظ - مكتبة socket
91.....	ص.ص.ع - مكتبة NumPy
95.....	ض - التطبيقات العملية
98.....	ط - مشاريع بسيطة
98.....	ط.أ - بناء آلة حاسبة بسيطة
99.....	ط.ب - مولد كلمات السر
100.....	ط.ج - لعبة تخمين الرقم
102.....	ظ - البرمجة الكائنية
107.....	ع - كود شامل
111.....	غ - النهاية



حول

المؤلف : عرايبيّة عامر

الصفحات : 118 صفحة

حسابي GitHub : Amerlaceset

اسم الكتاب : هيا نتعلم البرمجة بلغة البايثون معا