□□□□□□ دروس حول تعليمية ملاحظات □

الرسمية، الوثائق على بناءً الآن، □□□□□□. لغة عن المفهم بعض لديها أصبح السابق، التعلم خلال من
مستمر في سنة كل في المكتسبة المعرفة بعض حول □□□□□□. الإضافية

الكود تدفق في التحكم

النوع

```
print(type(1))
```

```
<class 'int'>
```

```
print(type('a'))
```

```
<class 'str'>
```

الكائن. نوع لطباعة مفيدة type دالة تُعتبر

النطاق □range

الدالة تُستخدم range في □for التكرارية الحلقات في استخدامها يمكن الأرقام. من سلسلة لتوليد بايثون في
□□□□□□ التكرار لتنفيذ معينة عملية لعدد محدد من مرات.

الدالة: صيغة

```
range(start, stop, step)
```

- □ □□□□□: اختياري□ القيمة هي الافتراضية القيمة 0□.
- □ □□□□: إلزامي□ الأول القيمة هذه تتضمن. في النتائج□.
- □ □□□□: اختياري□ السلسلة في متتالية عددين كل بين الفرق هي الافتراضية القيمة 1□.

أمثلة:

1. توليد سلسلة من 0 إلى 4:

```
for i in range(5):
    print(i)
```

الناتج:

```
0
1
2
3
4
```

2. دي تولة سلسلة من 2 إلى 8 بخطوة 2:

```
for i in range(2, 9, 2):
    print(i)
```

الناتج:

```
2
4
6
8
```

3. دي تولة سلسلة تنازلية من 10 إلى 1:

```
for i in range(10, 0, -1):
    print(i)
```

الناتج:

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

بايثون. في التكرارية الحلقات مع التعامل في وفعالة قوية أداة range تُعتبر

جدًا. مفيدة دالة range تُعتبر

```
for i in range(5):
  print(i, end = ' ')
```

```
0 1 2 3 4
```

```
for i in range(2, 6, 2):
  print(i, end = ' ')
```

الترجمة:

```
for i in range(2, 6, 2):
  print(i, end = ' ')
```

ملاحظة: الطريقة بنفس تُكتب البرمجية الأكواد أن حيث الترجمة، عند يتغير لا □□□□□□ بلغة المكتوب الكود اللغة. عن النظر بغض

```
2 4
```

انظر إلى فيرعت دالة range.

```
class range(Sequence[int]):
    start: int
    stop: int
    step: int
```

من تسلسل لإنشاء تُستخدم هذه الفئة □□□□□□. البرمجة لغة في range لفئة تعريفًا أعلاه الكود تمثل خصائص: ثلاث على الفئة تحتوي الصحيحة. الأعداد

- start: تمثل القيمة الأولية في التسلسل.
- stop: تمثل القيمة النهائية في التسلسل □غير مشمولة □. في التسلسل
- step: تمثل الفرق بين كل عددين متتاليين في التسلسل.

هذا الكود يُظهر استخدام الفئة تعريف في Sequence[int] نوع أن إلى للإشارة تُرجع الفئة من تسلسلًا الصحيحة. الأعداد

هو Visible. كلاس

```
print(range(5))
```

```
range(0, 5)
```

من: بدلاً

```
[0,1,2,3,4]
```

اس تمر.

```
print(list(range(5)))
```

```
[0, 1, 2, 3, 4]
```

؟اذا لم ر ظن ا ى إل ف ي ر ع ت list.

```
class list(MutableSequence[_T], Generic[_T]):
```

ال ب ر م ر ج ة ل غ ة ف ي list ل ف ئ ة ت ع ر ي ف ًا أ ع ا ه ال ك ود ت م ث ل .□□□□□□□ ه ذ ه ة ئ ف ي ال ه ي ء ز ج ن م ة ب ت ك م typing و ت س ت خ د م ق ا ب ل ت س ل س ل ه ي list أ ن إ ل ى ي ش ي ر [_T]MutableSequence ه ن ا، ع ا م. ب ش ك ل □□□□□□□□ ال ق و ا ئ و م ن و ع ل ت ح ي د ل ل □□□□□□□□□ □□□□□□□□□□ م ن ر ص ا ن ع ال ن و ع م ن □□□□□□□□□□ □□□□□□□□□ ع ا م ن و ع ه و T_ ح ي ث ،T_ ال ن و ع م ن ر ص ا ن ع ال ن و ع أ ي ك و ن ال ب ي ا ن ا ت. م ن ن و ع أ ي ك و ن

ف ي ر ع ت list و ه و list(MutableSequence[_T], Generic[_T]):. ا م ن ي ب ف ي ر ع ت range و ه و class range(Sequence[int]). ال ـ list م ن ي ر ث م ن MutableSequence. ا م ن ي ب ـ ال range م ن ي ر ث م ن Sequence.

ر ا ر م ت س ا ل ا ب ف ي، ث ح ب ل ا د ج ت س ا م ا ي ل ي:

```
Sequence = _alias(collections.abc.Sequence, 1)
MutableSequence = _alias(collections.abc.MutableSequence, 1)
```

ا ن ه ا ل م ه ف ن ق ة ع ل ا ق ة ن ف ه م ا. ا ن ن ك م ي ة ا ب ة ت ا ك list(range(5)) ه ذ ه ب ة ق ي ر ط ا ل. ل ك ن ب ي ن ه م ا ا ذ ا ل م ا ن ك م ي ة ا ب ة ت ا ك list(range(5)) ه ذ ه ب ة ق ي ر ط ا ل.

ال دوال مع امل ات

ي ق ل ن ل ة ر ظ ن ر ى ل ع ة ف ر ف ر ع م ل ا ة ي ل ي ك ت م ل ا ح و ل ا ل د و ا ل.

```
def fn(a = 3):
  print(a)
```

□□□□

```shell
3
```

افتراضية. قيمة المعلمة يعطي هذا

```
def fn(end: int, start = 1):
  i = start
  s = 0
  while i < end:
    s += i
    i += 1
  return s
```

العربية: إلى الكود ترجمة

```
def fn(end: int, start = 1):
  i = start
  s = 0
  while i < end:
    s += i
    i += 1
  return s
```

الكود: شرح

- المُدخلين: تأخذ الدالة fn
  - end: عدد صحيح لتمثل النهاية.
  - start: عدد صحيح اختياري لتمثل البداية، وله قيمة افتراضية تساوي 1.
- يتم تهيئة المتغير i بقيمة start والمتغير s بقيمة 0.
- يتم تنفيذ حلقة while طالما أن i أقل من end. في كل تكرار:
  - يتم إضافة قيمة i إلى s.
  - يتم زيادة قيمة i بمقدار 1.
- في النهاية، يتم إرجاع قيمة s التي تمثل مجموع الأعداد من start إلى end - 1.

```
print(fn(10))
```

45

المقدمة. في الإلزامية المعلمات كتابة يجب أنه لاحظ إلزامية. معلمة هو end

```
def fn(start = 1, end: int):

    def fn(start = 1, end: int):
                                ^
SyntaxError:
```

لاحظ أن end هو non-default argument المباين start هو default argument. هذا يعني أن المعامل غير جميع قبل الافتراضية القيم وضع يجب آخر، بمعنى الافتراضي. المعامل بعد يأتي الافتراضي القيمة سيأخذ فإنه له، قيمة تمرير يتم لم إذا أنه أي افتراضي، معامل هو start الافتراضية. المعاملات افتراضية.

```
def fn(a, /, b):
  print(a + b)
```

كوسيطة يُمرر أن ويجب إلزامي مُعامل هو a المُعامل .b و a مُعاملين: تأخذ fn تُسمى دالة تعريف يتم أعلاه، الكود في موضعية [illegible] مسمّاة كوسيطة أو موضعية كوسيطة يُمرر أن يمكن b باين، [illegible] التي تلك عن فقط موضعية كوسائط تُمرر أن يجب التي المُعاملات باين لفصل يُستخدم / الرمز. [illegible] مسمّاة. كوسائط تُمرر أن يمكن

```
fn(1, 3)
```

والأخرى الموضع، على تعتمد إحداهما المعاملات: لتمرير طريقتان هناك المعاملات. أنواع لفصل / استخدام يتم المفتاحية. الكلمات تحديد على تعتمد

```
def fn(a, /, b):
  print(a + b)
```

يجب a المعلمة أن إلى للإشارة تُستخدم / الإشارة .b و a معلمتين: تأخذ fn دالة تعريف يتم أعلاه، الكود في اسمية [illegible] موضعي بشكل تُمرر أن يمكن ولا فقط، [illegible]. تطبع ثم b و a القيمتين بجمع تقوم الدالة اسمية. كمعلمة أو موضعي بشكل تمريرها يمكن b المعلمة باين النتيجة.

```
fn(a=1, 3)

    fn(a=1, 3)
            ^
SyntaxError:
```

تُعامل يجعلها هذا المفتاحية. الكلمة طريق عن تمريرها يتم المعلمة هذه أن تعني a=1 الطريقة. بهذه يعمل ل اهذ موضعية. كمعلمة تُعامل b بينما مفتاحية. كمعلمة

```
def f(pos1, pos2, /, pos_or_kwd, *, kwd1, kwd2):
      -----------    ----------     ----------
        |             |                  |
        |                       |
        |                                  -
         --
```

اتباع عليك يجب لذلك المعاملات. تمرير نوع ضمنيًا يعني * و / استخدام الدالة، تعريف عند أنه هنا لاحظ دعالقوا في تمريرها.

```
def fn(a, /, b):
  print(a + b)
```

إلزامي مُعامل هو a المُعامل .b و a مُعاملين: تأخذ والتي fn تُسمى البرمجة لغة في دالة أعلاه الكود تمثل كوسيطة تمريره يمكن b المُعامل بينما، موضعية كوسيطة تمريره يتم أن ويجب النتيجة. تطبع ثم b و a القيمتين بجمع تقوم الدالة. مسمّاة كوسيطة أو موضعية

```
fn(1, b=3)
```

أخطاء. أي عن الإبلاغ يتم لم الطريقة بهذه

```
def fn(a, /, b, *, c):
  print(a + b + c)
```

الدوال تعريف في * الموضعية المعلمات استخدام على مثالًا أعلاه fn الدالة تمثل / الكلمات والمفتاحية في لغة. هنا:

- a هي معلمة موضعية، فقط مما يعني أنه لا يمكن تمريرها إلا عن طريق وضع الموضع ليس عن طريق الكلمة المفتاحية.
- b هي معلمة يمكن تمريرها إما عن طريق وضع الموضع أو الكلمة المفتاحية.
- c هي معلمة كلمات مفتاحية، فقط مما يعني أنه يجب استخدام الكلمة المفتاحية ولا يمكن تمريرها عن طريق الموضع.

```
fn(1, 3, 4)
```

```
    fn(1, 3, 4)
TypeError: fn()                          3
```

fn نكمي أن لبقتست طقف نيتمعلم، نيتيعضوم لكن ولم تم رير3.

```
def fn(a, /, b, *, c):
  print(a + b + c)
```

فقط موضعية معلمة هي a المعلمة .c و b و a معاملات: ثلاثة تأخذ fn الدالة أعلاه، الكود في [illegible] أو موضعية كمعلمة إما تمريرها يمكن b المعلمة اسمها. باستخدام لا نكمي أنه يعني مما [illegible]، تمريرها يجب أنه يعني مما [illegible]، فقط مفتاحية معلمة هي c المعلمة اسمها. باستخدام اسمها.

```
fn(a = 1, b=3, c=4)
```

```shell
    fn(a = 1, b=3, c=4)
TypeError: fn()                                    :     'a'
```

المفتاحية. الكلمات باستخدام الآن تُمرر أصبحت الموضع طريق عن فقط تُمرر كانت التي المعاملات بعض

[illegible] الخرائط بشكل المعلمات

```
def fn(**kwds):
  print(kwds)
```

إلى: أعلاه الكود ترجمة تمت

```
def fn(**kwds):
  print(kwds)
```

[illegible] المسمية الوسائط من محدد غير عددًا تأخذ fn تُسمى دالة تعريف يتم الكود، هذا في kwds. ثم يتم طباعة هذا .القاموس القاموس في وتخزنها

```
fn(**{'a': 1})
```

```
{'a': 1}
```

```
def fn(**kwds):
  print(kwds['a'])

d = {'a': 1}
fn(**d)
```

تمرير d قاموسًا أعلاه الكود تمثل **d، باستخدام fn الدالة استدعاء عند 1. وقيمته 'a' مفتاح على يحتوي d قاموس

اسمية كلمات القاموس محتويات fn الدالة أن يعني هذا للدالة. [illegible] استدعاؤها سيتم

كالتالي: صريح بشكل مكتوبة المعاملات كانت لو كما

```
fn(a=1)
```

قاموس. محتويات على بناءً دالة إلى ديناميكية معاملات تمرير تريد عندما مفيدة الطريقة هذه

```
1
```

المعاملات. بتوسيع يقوم ** أن الواضح من

```
def fn(a, **kwds):
  print(kwds['a'])
```

العربية: إلى الكود ترجمة

```
def fn(a, **kwds):
  print(kwds['a'])
```

الكلمات على يحتوي قاموس [illegible] وهو **kwds اختياريًا ووسيطًا إلزاميًا a وسيطًا تأخذ fn الدالة الكود، هذا في

المفتاحية[illegible]. الإضافية الدالة تقوم بطباعة قيمة المفتاح 'a' من القاموس kwds.

ملاحظة: إذا تم استدعاء الدالة بدون تمرير قيمة للمفتاح 'a' في kwds، فسوف يتم إرجاع خطأ KeyError.

```
d = {'a': 1}
fn(1, **d)
```

مع fn الدالة استدعاء يتم ثم .1 وقيمته 'a' مفتاح على يحتوي d قاموس تعريف يتم أعلاه، الكود في

محتويات أن يعني مما ** باستخدام تمريره يتم الذي d القاموس هي والثانية، 1 الرقم هي الأولى وسيطتين:

للدالة. [illegible] اسمية كوسائط تمريرها سيتم القاموس

```
TypeError: fn()                    'a'
```

دنع ءاعدتسا ةالد لثم fn(1, **d)، متي اهءيسوت ىلإ fn(a=1, a=1). يلاتلابو ثدحيس أطخ.

```
def fn(**kwds):
  print(kwds['a'])

d = {'a': 1}
fn(d)

TypeError:     fn()    0                  1
```

إذا تم ءاعدتسا ةالدال لثم fn(d)، متيسف لماعتال اهعم تامالعمك ةيعضوم [illegible] [illegible] مسيول تامالعمك ةيحاتفم [illegible] [illegible] موسعة.

```
def fn(a, / , **kwds):
  print(kwds['a'])

d = {'a': 1}
fn(1, **d)
```

ي فالكود أعلاه، متي ءاشنإ موسقا d يوتحي على حاتفم 'a' هتميقو 1. ثم متي ءاعدتسا ةالدال fn عم ةطيسولا ال1، الأولى متي رييمرت موسقا d ةطيسولا كوسي ةيمسا مادختساب **d. هذا يعني أن حاتفملا 'a' متيس هريمرت ةالدلل fn عم ةميقلا 1.

هذه لمعي لكشب. يجيد حضوي أن لمحت سفن نم يكمي أن نيعيتت نيعضوملا تامالعملا والعملا على لكش ال.

```
def fn(a, / , a):
  print(a)
```

ي فالكود أعلاه، متي فيرعت ةالد مسا fn ذخأتس، نيلماعملا امهالك ىمسيُ a. ةملعملا ىلولأا a يه ةملعم ةيعضوم a في ط [illegible] [illegible] بسبب مادختسا /، امم ينعي هنأ لا نكمي اهريمرت مادختساب امهسا ةملعملا كلذ ةملك ةيحاتفم [illegible] [illegible]. ةملعملا ةيناثلا a يه ةملعم ةيدايع نكمي اهريمرت ةملعملا كلذ ةملك أو ةيعضوم ةملعملا كلذ ةملك ةيحاتفم.

تضاربًا. امم بسبي a، ال مسفن ال مهل ةملعملا نيتملعملا نأل ال تشغيل وقت في خطأ إلى سيؤدي الكود اذه ذلك، وعم ال، لغة في [illegible] الدالة. فيرعت في ال مسفن لل تامالعملا كون أن نكمي ال

```
d = {'a': 1}
fn(1, **d)
```

تمرير مع الدالة استدعاء يتم ثم .1 وقيمته 'a' مفتاح على يحتوي d قاموس تعريف يتم أعلاه، الكود في fn الدالة أن يعني هذا .**d باستخدام إضافية كوسائط d القاموس محتويات وتمرير أولى، كوسيطة 1 الرقم a=1. الوسيطة إلى بالإضافة 1 الوسيطة ستستقبل

```
SyntaxError:          'a'
```

الحالات. هذه بين الدقيقة العلاقة لاحظ خطأ. يحدث الشكل بهذا

```
def fn(a, / , **kwds):
  print(kwds['a'])
```

□□□1, □□□1,2□□

```shell
TypeError: __main__.fn()        **              (mapping)        (list)
```

** يجب أن يتبعه تعيين.

للتكرار القابلة الأنواع مع العمل

عند العمل مع الدوال في □□□□□□، قد تحتاج إلى تمرير مع العمل من أنواع قابلة للتكرار □□□□□□□□□□□ مثل □□□□□□□، أو القواميس □□□□□□□□□، المجموعات □□□□□□□، القوائم □□□□□□□. هذه الأنواع تسمح لك بتخزين مجموعة من العناصر والوصول إليها بشكل تسلسلي.

على القائمة تمرير مثال

```
def print_elements(items):
    for item in items:
        print(item)

my_list = [1, 2, 3, 4, 5]
print_elements(my_list)
```

استدعاء عند للتكرار. قابلة تكون أن يُفترض التي items مع القائمة تأخذ print_elements الدالة المثال، هذا في القائمة، my_list، تم طباعة كل عنصر في القائمة. قائمة مع الدالة

كمعلمة مجموعة تمرير على مثال

```python
def print_elements(items):
    for item in items:
        print(item)

my_tuple = (10, 20, 30)
print_elements(my_tuple)
```

المجموعة. في عنصر كل طباعة ويتم ،print_elements للدالة كمعلمة my_tuple مجموعة تمرير يتم ،هنا

كمعلمة قاموس تمرير على مثال

```python
def print_elements(items):
    for key, value in items.items():
        print(f"{key}: {value}")

my_dict = {'a': 1, 'b': 2, 'c': 3}
print_elements(my_dict)
```

items() الأسلوب استخدام يتم .print_elements للدالة كمعلمة my_dict قاموس تمرير يتم المثال، هذا في وطباعتها. القاموس في والقيم المفاتيح إلى للوصول

الخلاصة

سواء الاستخدام. لإعادة وقابلية مرونة أكثر دوال بكتابة لك يسمح للتكرار قابلة أنواع من معلمات تمرير بشكل معها والتعامل للدوال كمعلمات تمريرها بسهولة يمكنك قواميس، أو مجموعات، قوائم، مع تتعامل كنت متسلسل.

```python
def fn(*kwds):
  print(kwds)

fn(*[1,2])
```

□□□□ للتكرار قابل كائن أي أو قائمة من العناصر لفك تُستخدم * النجمية العلامة، □□□□□□ البرمجة لغة في منفصلة. كوسائط 2 و 1 العناصر ستستقبل fn الدالة المثال، هذا في دالة. وسائط إلى □□□□□□□

كالتالي: تُعرف fn الدالة كانت إذا

```
def fn(a, b):
    print(a, b)
```

الناتج: سيكون، fn(*[1, 2]) استدعاء عند

```
1 2
```

لأن القائمة [1, 2] تم فكها إلى وسيطين منفصلين و 1 و 2 للدالة fn.

```
(1, 2)
```

```
def fn(*kwds):
  print(kwds)
```

إلى: أعلاه الكود ترجمة تم

```
def fn(*kwds):
  print(kwds)
```

استدعاء عند .kwds* باسم تُعرف الوسائط من محدد غير عددًا تأخذ fn تُسمى دالة تعريف يتم الكود، هذا في إليها. تمريرها تم التي الوسائط جميع طباعة يتم الدالة،

```
TypeError: __main__.fn()        *
```

* يجب أن يتبع iterable.

```
def fn(a, *kwds):
  print(type(kwds))

fn(1, *[1])
```

قائمة هي الثانية والوسيطة، 1 الرقم هي الأولى الوسيطة وسيطتين. مع fn الدالة استدعاء يتم أعلاه، الكود في سيتم القائمة في الموجودة العناصر أن يعني هذا .* العامل باستخدام تفكيكها يتم ولكن، 1 الرقم على تحتوي fn. للدالة ثانية كوسيطة 1 الرقم تمرير سيتم الحالة، هذه في للدالة. منفصلة كوسائط تمريرها

fn الدالة كانت إذا المثال، سبيل على صحيح. بشكل استدعاؤها فسيتم وسيطتين، تتوقع fn الدالة كانت إذا كالتالي: تعرف

```
def fn(a, b):
    return a + b
```

إن استدعاء fn(1, *[1]) سيعادل fn(1, 1) وسيُرجع الناتج 2.

```
<class 'tuple'>
```

لنطبع النوع. هذا هو السبب في أن أعلاه الإخراج كان (1,2) وليس [1,2].

```
def fn(*kwds):
  print(kwds)
```

تم ترجمة الكود أعلاه إلى:

```
def fn(*kwds):
  print(kwds)
```

في هذا، يتم تعريف دالة تُسمى fn تأخذ عددًا غير محدد من وسائط الكلمات المفتاحية باستخدام *kwds. عند استدعاء الدالة، يتم طباعة جميع الوسائط التي تم تمريرها إليها.

```
fn(1, *[1])
```

الترجمة:

```
fn(1, *[1])
```

في الدالة البرمجية، يتم استدعاء fn مع وسيطة واحدة 1 وقائمة تحتوي على عنصر واحد هو 1 وعامل الاستخدام *. حيث يتم فك القائمة،

```
(1, 1)
```

لاحظ أنه عند استدعاء fn(1, *[1])، يتم تحليل fn(1, 1). ثم عند استدعاء fn(*kwds)، يقوم kwds بتحويل مجموعة (1, 1) إلى 1, 1.

```
def concat(*args, sep='/'):
  return sep.join(args)
```

الترجمة إلى الكود العربية:

```
def concat(*args, sep='/'):
  return sep.join(args)
```

الكود: شرح

- الدالة concat تأخذ عددًا غير محدد من الوسائط *args اختياريًا ووسيطًا sep قيمة افتراضية /.
- الدالة تجمع الوسائط باستخدام الفاصل المحدد sep وتعيد النتيجة كسلسلة نصية واحدة.

مثال:

```
print(concat('path', 'to', 'file'))  # :     path/to/file
print(concat('path', 'to', 'file', sep='-'))  # :     path-to-file
```

ملاحظة: الكود يبقى كما هو لأن الإنجليزية لغة برمجة. في تُترجم الدوال أسماء والمتغيرات لكي تعمل الشيفرة.

```
print(concat('a','b','c', sep=','))
```

```
a,b,c
```

تعبيرات

الحاسوب؟ علوم أغاز الحل مقالة في ذكرناه ما تتذكر هل كمتغيرات. الدوال لحفظ طريقة هي lambda

```
def incrementor(n):
  return lambda x: x + n
```

هذه الكود عزّف يسمى دالة incrementor تأخذ عددًا واحدًا n. الدالة تُرجع دالة أخرى تُعرف مسبقًا دالة المدمجة تأخذ عددًا واحدًا x وتُرجع مجموع x و n. بمعنى آخر، الدالة المُرجَعة تزيد قيمة x بمقدار n.

```
f = incrementor(2)
print(f(3))
```

```
5
```

مثال آخر. إلى النظر

```
pairs = [(1, 4), (2, 1), (0, 3)]

pairs.sort(key = lambda pair: pair[1])
```

العربية: إلى تُرجم

```
pairs.sort(key = lambda pair: pair[1])
```

شرح:
 pairs: قائمة من أزواج pair مثل أو . sort: دالة ترتب العناصر في القائمة. key = lambda pair: pair[1]: تحدد أن الترتيب سيتم بناءً على العنصر الثاني pair[1] لكل pair.
 lambda: دالة مجهولة تُستخدم لإنشاء دالة صغيرة في سطر واحد.

ملاحظة:
 باللغة الإنجليزية. تُكتب بايثون في المتغيرات الدوال أسماء لأن هو كما يبقى الكود

```
print(pairs)
```

```
[(2, 1), (0, 3), (1, 4)]
```

```
pairs = [(1, 4), (2,1), (0, 3)]
```

```
pairs.sort(key = lambda pair: pair[0])
```

```
print(pairs)
```

```
[(0, 3), (1, 4), (2, 1)]
```

الرقم على بناءً الفرز يتم ،pair[1] استخدام وعند الأول. الرقم على بناءً الفرز يتم ،pair[0] استخدام عند الثاني.

التوثيق تعليقات

```
def add():
    """
  """
  pass

print(add.__doc__)
```

الدالة توقيع

```
def add(a:int, b:int) -> int:
  print(add.__annotations__)
  return a+b
```

الكود: ترجمة

```
def add(a:int, b:int) -> int:
  print(add.__annotations__)
  return a+b
```

الكود: شرح

- الدالة add تأخذ مُدخلين من نوع int وتُرجع صحيحًا من نوع int.
- add.__annotations__ تطبع القيود التوضيحية للدالة، والتي تُظهر أنواع المُدخلات والمُخرجات.
- الدالة تُرجع مجموع a و b.

ملاحظة: في بايثون تُكتب التوضيحات والتعليقات والمتغيرات الدوال أسماء لأن هو كما يبقى الكود الرسمية اللغة.

```
add(1, 2)
```

```
{'a': <class 'int'>, 'b': <class 'int'>, 'return': <class 'int'>}
```

البيانات هياكل

القوائم

```
a = [1,2,3,4]

a.append(5)
print(a)   # [1, 2, 3, 4, 5]

a[len(a):] = [6]
print(a)   # [1, 2, 3, 4, 5, 6]
```

```
a[3:] = [6]
print(a)   # [1, 2, 3, 6]

a.insert(0, -1)
print(a)   # [-1, 1, 2, 3, 6]

a.remove(1)
print(a)   # [-1, 2, 3, 6]

a.pop()
print(a)  # [-1, 2, 3]

a.clear()
print(a)  # []

a[:] = [1, 2]
print(a.count(1)) # 1

a.reverse()
print(a)   # [2, 1]

b = a.copy()
a[0] = 10
print(b)   # [2, 1]
print(a)   # [10, 1]
```

عند تنفيذ الكود أعلاه، تم إنشاء نسخة من القائمة a وتخزينها في المتغير b. ثم تم تعديل العنصر الأول في القائمة الأصلية a ليصبح 10. عند طباعة القائمة b، نلاحظ أنها لا تزال تحتوي على القيمة الأصلية [2, 1]، مما يدل على أن التعديلات على القائمة الأصلية لم تؤثر على القائمة a التي تم إنشاؤها باستخدام copy().

```
b = a
a[0] = 3
print(b)  # [3, 1]
print(a)  # [3, 1]
```

الق وائم بناء

```
print(3 ** 2)   # 9
print(3 ** 3)   # 27
```

أدبنل م علتب ية ملع، ية ابي حس وهي **. هذه ية ملعال ي نعت .

```
sq = []
for x in range(10):
  sq.append(x ** 2)

print(sq)
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

برجنل آن ال دام خت اس map.

```
a = map(lambda x:x, range(10))
print(a)
# <map object at 0x103bb0550>
print(list(a))
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

sq = map(lambda x: x ** 2, range(10))
print(list(sq))
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

sq = [x ** 2 for x in range(10)]
print(sq)
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

نكمي ظة احل م أن for ة رن م. جدًا

```
a = [i for i in range(5)]
print(a)
# [0, 1, 2, 3, 4]

a = [i+j for i in range(3) for j in range(3)]
print(a)
# [0, 1, 2, 1, 2, 3, 2, 3, 4]
```

```
a = [i for i in range(5) if i % 2 == 0]
print(a)
# [0, 2, 4]

a = [(i,i) for i in range(3)]
print(a)
# [(0, 0), (1, 1), (2, 2)]
```

ءانب مئاوقلا ةلخادتملا

```
matrix = [[(i+j*4) for i in range(4)] for j in range(3)]
print(matrix)
# [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]

t = []
for j in range(3):
  t.append([(i+j*4) for i in range(4)])
print(t)
# [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
```

ظحال ةقيرطلا يتلا بتكت اهب ناذه نيعطقملا من الكود. أي:

```
[[(i+j*4) for i in range(4)] for j in range(3)]
```

ريسفتلا:

اذه دوكلا ئشنُي ةمئاق ةلخادتم ةمئاق □ يوتحت على مئاوق □ رخأ مادختساب موهفم □□□□□ □□□□□□□□□□□□□□□□ □□□□□□ في ةغل □□□□□□. لصفنل دوكلا ةوطخ بةوطخ:

1. ةقلحلا ةيجراخلا □`for j in range(3)`□:
   - □ هذه ةقلحلا لمعت ثلاث ،تارم ثيح j ذخأتو ميقلا 0، 1، 2.
2. ةقلحلا ةيلخادلا □`for i in range(4)`□:
   - □ لكل ةميق من j، لمعت هذه ةقلحلا عبرأ ،تارم ثيح i ذخأتو ميقلا 0، 1، 2، 3.
3. ريبعتلا `(i + j * 4)`:
   - □ لكل جوز من i و j، متي باسحو ةميقلا `(i + j * 4)`.
   - □ ىلع ليبس لاثملا:

- عندما j = 0 و i = 0، تكون القيمة 0 + 0*4 = 0.
- عندما j = 1 و i = 2، تكون القيمة 2 + 1*4 = 6.

4. **النهائية: النتيجة**

- أقوم بإنشاء قائمة تحتوي على ثلاث قوائم، كل قائمة تحتوي على أربعة أرقام.
- ستكون النتيجة:

```
[[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
```

ملاحظة: الكود الموجود في الإجابة هو جزء من لغة البرمجة التي تم اختياره بناءً على السياق. يمكن استخدام أي لغة برمجة أخرى لتحقيق نفس النتيجة.

هذا هو الكود:

```
for j in range(3):
   [(i+j*4) for i in range(4)]
```

ملاحظة: الكود أعلاه مكتوب بلغة البرمجة التي تم اختيارها. إذا كنت بحاجة إلى شرح الكود أو ترجمته إلى لغة برمجة أخرى، فيرجى التعليق أدناه.

هذا هو الكود:

```
for j in range(3):
  for i in range(4):
      (i+j*4)
```

ملاحظة: الكود أعلاه مكتوب بلغة البرمجة التي تم اختيارها. إذا كنت بحاجة إلى شرح الكود أو ترجمته إلى لغة برمجة أخرى، فيرجى التعليق أدناه. سأكون سعيدًا بمساعدتك في ذلك.

لا يوجد أي إجراء مفيد يمكنني القيام به بناءً على المعلومات المتوفرة.

```
matrix = [[(i+j*4) for i in range(4)] for j in range(3)]
print(matrix)
#  [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
```

الترجمة:

```
matrix = [[(i+j*4) for i in range(4)] for j in range(3)]
print(matrix)
#  [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]]
```

النواتج المصفوفة. [illegible] باستخدام الأبعاد ثنائية مصفوفة إنشاء يتم الكود، هذا في
الناتجة المصفوفة تحتوي على 3 صفوف و4 أعمدة، حيث يتم استخدام عناصر كل حساب الصيغة (i + j * 4).
هي [[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]].

```
mt = [[row[j] for row in matrix] for j in range(4)]
print(mt)
# [[0, 4, 8], [1, 5, 9], [2, 6, 10], [3, 7, 11]]

print(list(zip(*matrix)))
[(0, 4, 8), (1, 5, 9), (2, 6, 10), (3, 7, 11)]
```

[illegible]

أو القواميس، القوائم، مثل الكائنات من مجددة عناصر لحذف تُستخدم بايثون في del المفتاحية الكلمة
استخدامها: كيفية على الأمثلة بعض إليك الذاكرة. من المتغيرات

قائمة: من عنصر حذف

```
my_list = [1, 2, 3, 4, 5]
del my_list[2]  #              2
print(my_list)  # :    1], 2, 4, 5]
```

قاموس: من مفتاح حذف

```
my_dict = {'a': 1, 'b': 2, 'c': 3}
del my_dict['b']  #         'b'
print(my_dict)  # :    }'a': 1, 'c': 3}
```

متغير: حذف

```
x = 10
del x  #         x
print(x)  #            x
```

الحاجة عند أو كبيرة كائنات مع العمل عند خاصة فعال، بشكل الذاكرة لإدارة مفيدًا يكون أن يمكن del استخدام
الذاكرة. مساحة تحرير إلى

```
a = [1, 2, 3, 4]

del a[1]
print(a)  # [1, 3, 4]

del a[0:2]
print(a) # [4]

del a
print(a) # NameError: name 'a' is not defined
```

الق اموس

```
ages = {'li': 19, 'wang': 28, 'he' : 7}
for name, age in ages.items():
    print(name, age)
```

الع ربي ة: إل ى ال ك ود ت رج م ة

```
     = {' ': 19, '   ': 28, ' ': 7}
for    ,     in :()items.
    print(   , (
```

ح ل ق ة اس ت خ دام ي ت م ث م ال أش خ اص. ب ع ض وأع م ار أس م اء ع ل ى ي ح ت وي ب اس م ق ام وس ت ع ري ف ي ت م ال ك ود، ه ذا ف ي ل ه. ال م ق ابل ال ع م ر م ع اس م ك ل ل ط ب اع ة for

19 لي

28 وانغ

7 هو

```
for name in ages:
    print(name)
```

```
#
#
#

for name, age in ages:
    print(name)
```

□□□□□□□□□□□: عدد ريبك ًدا جن من ميقال اهكفل عقوتمال□ وه 2□

```
for i, name in enumerate(['li', 'wang', 'he']):
    print(i, name)
```

لي 0

وانغ 1

هو 2

```
print(reversed([1, 2, 3]))
# <list_reverseiterator object at 0x10701ffd0>
```

هذا نفسها. العكوسة القائمة من بدلاً list_reverseiterator نوع من كائن طباعة يتم أعلاه، الكود تنفيذ عند لرؤية مباشرة. عكوسة قائمة ولي س الع ناصر، يعكس □□□□□□□□□□□ مكررًا كائنًا تُرجع reversed() الدالة لأن list() الدالة باستخدام قائمة إلى المكرر الكائن تحويل يمكنك العكوسة، القائمة:

```
print(list(reversed([1, 2, 3])))
# [3, 2, 1]

print(list(reversed([1, 2, 3])))
# [3, 2, 1]

###      (Modules)

###

```python
import sys
```
```

```python
def f(n):
    if n < 2:
        return n
    else:
        return f(n-1) + f(n-2)
```

هذه الدالة f(n) تحسب عدد فيبوناتشي للعدد n. إذا كان n أقل من 2، فإن الدالة ترجع n مباشرة. وإلا، فإنها ترجع مجموع الدالة f(n-1) و f(n-2). هذه هي الطريقة الكلاسيكية لحساب أعداد فيبوناتشي باستخدام العودية □□□□□□□□□□□.

```python
if __name__ == "__main__":
    r = f(int(sys.argv[1]))
    print(r)
```

```
% python fib.py 3
2
```

```
% python -m fib 5
5
```

دليل □□□□□

```python
import fib

print(dir(fib))
```

```
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec_
```

```python
import builtins
print(dir(builtins))
```

سيقوم هذا الكود باستيراد الوحدة المدمجة builtins ثم طباعة قائمة بجميع الأسماء المحددة في هذه الوحدة، والتي تشمل الدوال والمتغيرات المدمجة في لغة □□□□□□ مثل print، len، range، وغيرها.

□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□, □□□□□□□□□□□□□□□, □□□□□□□ □□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□ □□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□□□□□□□□□□□□□, □□□□□□□□□□, □□□□

[illegible]

##

`packages`.

```shell
pk.py
fibp
   cal
      cal.py
   pt
       pt.py
```

إلى: أعلاه الهيكل تم ترجمة

```
pk.py
fibp
   cal
      cal.py
   pt
       pt.py
```

الملفات أسماء أن حيث تجمرت إلى حاجتي ولا وملفات، مجلدات هيكل عن عبارة هو المذكور الهيكل ملاحظة:
هي. كما تبقى والمجلدات

□□□.□□:

```
def f(n):
    if n < 2:
        return n
    else:
        return f(n-1) + f(n-2)

def fl(n):
    return list(map(f, range(5)))
```

ملاحظة: الوحدة الدالة باستخدام فيبوناتشي متسلسلة بحساب يقوم □□□□□□ بلغة مكتوبة أعلاه الكود
f. الدالة fl تقوم بإرجاع قائمة تحتوي على أول 5 أرقام في متسلسلة فيبوناتشي.

pt.py:

```
def p(l):
    print(l, end=' ')

def pln(l):
    print(l)

# pk.py

import fibp.cal.cal
import fibp.pt.pt
```

□□□□.□□.□□.□□□□□□□.□□□.□□□.□□□10□□

`pk.py` :

```python
from fibp.cal import cal
from fibp.pt import pt
```

الغة في هي كما تبقى والدوال النمطية الوحدات أسماء أن حيث ترجمة إلى يحتاج لا المقدم الكود ملاحظة:
 الإنجليزية.

□□.□□□□□.□□□10□□ □□