Information Technology Institute

بايثون

اليوم 04

البرمجة للجميع

بيثون كائني المنحى

مقدمة

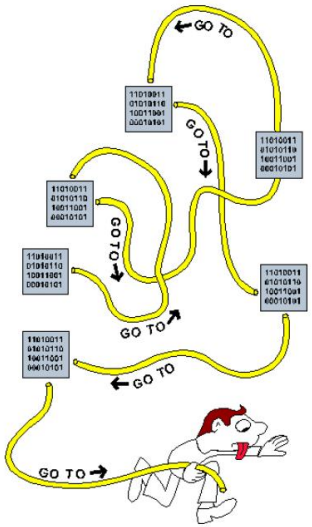
وجوه المنحى
مستوى

الوحدات

المستوى المعياري

المهام

إجرائية
مستوى



مستوى Speghatti

ملكيات

السرعة 200 =م / ث
الفرامل 2 =

الطول 175 =سم
الوزن 76 =كجم

وقف ()تحرك
()

المشي ()الكلام
()

طُرق

ركوب (BikeObj)

كائن الدراجة

كائن الرجل

کلمات OOP

فئة هي تعريف قالب لخصائص الكائن وطرقه.

الطبقة البشرية

فئة الإنسان:

يمر

هدف

الكائن هو مثيل في فئة.

الطبقة البشرية

كائن الرجل

فئة الإنسان:

يمر

رجل =إنسان ()

البناء

المُنشئ هو طريقة تسمى في اللحظة التي يتم فيها إنشاء كائن.

الطبقة البشرية

حرارة () __

كائن الرجل

فئة الإنسان:

الحرف الأول (ذاتي):

طباعة ("مرحبًا بك")

رجل =إنسان ()

انتاج:

أهلاً

المتغيرات الخاصة

متغير المثيل هو خاصية كائن ، مثل الاسم.

الطبقة البشرية

اسم

حرارة ()

الاسم احمد

كائن الرجل

فئة الإنسان:

الحرف الأول :(self، name)

self.name = الاسم

man = Human(“Ahmed”)

فئة متغير

ClassVariableهو المتغير المشترك بين جميع المثيلات.

فئة الإنسان:

makeFault = صحيح

```
    def init (self، name): self.name = name؛
```

man = Human(“Ahmed”) man2 = Human(“Mohamed”)

الاسم محمد

يرتكب أخطاء

الاسم احمد

يرتكب أخطاء

فئة متغير

فئة الإنسان:

أخطاء 0 =

def init (self، name): self.name = name؛

Human(“Ahmed”) man2 = Human(“Mohamed”)

man =

أخطاء الرجل 1 =

طباعة (man.faults، "Man :")

طباعة (man2.faults، "Man 2:")

طباعة (Human.faults، "Human:")

أخطاء الإنسان 2 =

طباعة (man2.faults، "Man 2:")

طباعة (Human.faults، "Human:")

طباعة (man.faults، "Man :")

انتاج:

الرجل: 1

Man2 : 0

الإنسان: 0

Man2 : 2

الإنسان: 2

الرجل: 1

طريقة المثيل

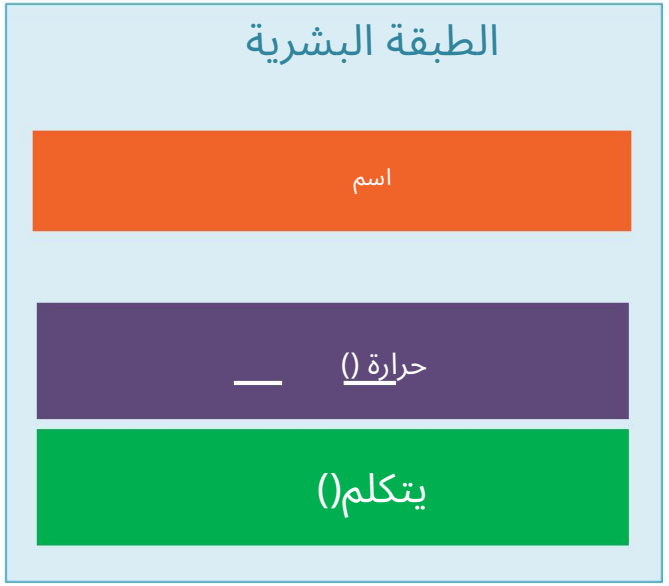
طريقة المثيل هي قدرة كائن ، مثل المشي.

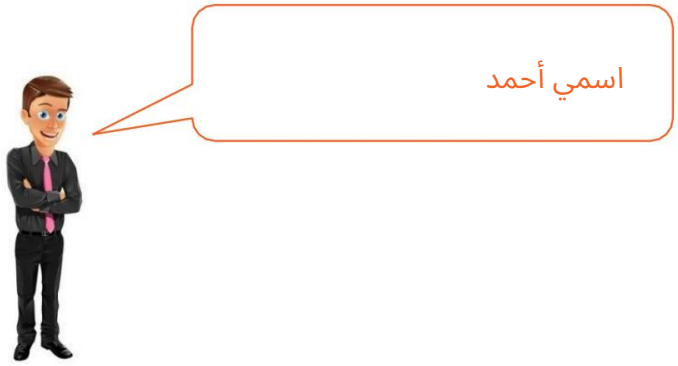
فئة الإنسان:

```
def init (self، name): self.name = name

def talk (self): print (“My Name is ” + self.name)

man = الإنسان ("أحمد") man.speak ()
```





طريقة الفصل

ClassMethodهي طريقة مشتركة بين جميع مثيلات الفئة

فئة الإنسان:

أخطاء 0 =

الحرف الأول :(self، name)

الاسمself.name =

تضمين التغريدة

جعل الافتراضي أخطاء :(cls)

cls.faults + = 1

طباعة (cls.faults)

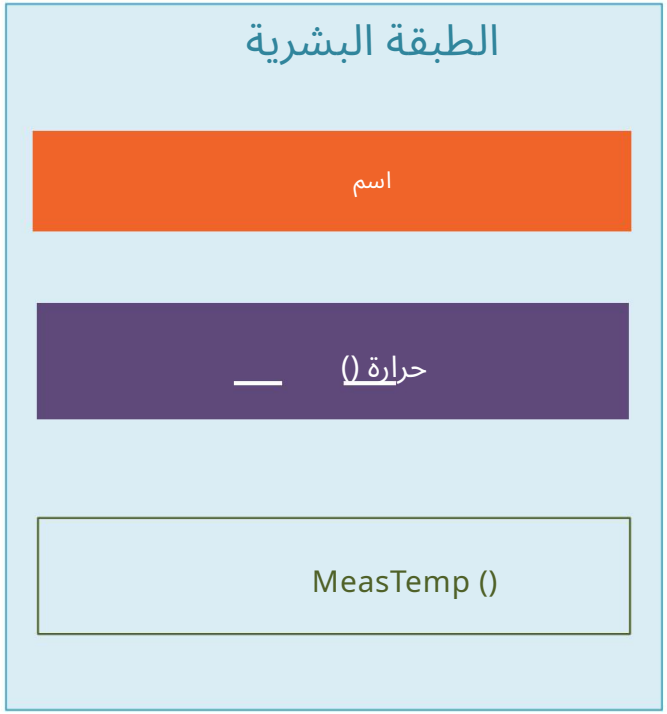
Human.makeFaults () # 1

man = Human(“Ahmed”)

man.makeFaults () # 2

طريقة ثابتة

الطريقة الثابتة هي وظيفة عادية لها منطق متعلق بالفئة



فئة الإنسان:

الحرف الأول :(self، name) _ _

self.name = الاسم

تضمين التغريدة

مقياس defTemp(درجة الحرارة):

إذا (درجة الحرارة :(37 ==

إرجاع "عادي"

إرجاع "غير عادي"

Human.measureTemp (38) # ليس عادي

tatic vs Class Methods

طريقة ثابتة

تحتوي على

المنطق الذي يتعلق بالفئة.

#نسميها الطريقة المساعدة

فئة الإنسان:

staticmethod def sleep ():

@

طباعة ("قف")

الإنسان.النوم ()

طريقة الفصل

#طريقة الفصل مرتبطة بالفصل نفسه.

فئة الإنسان:

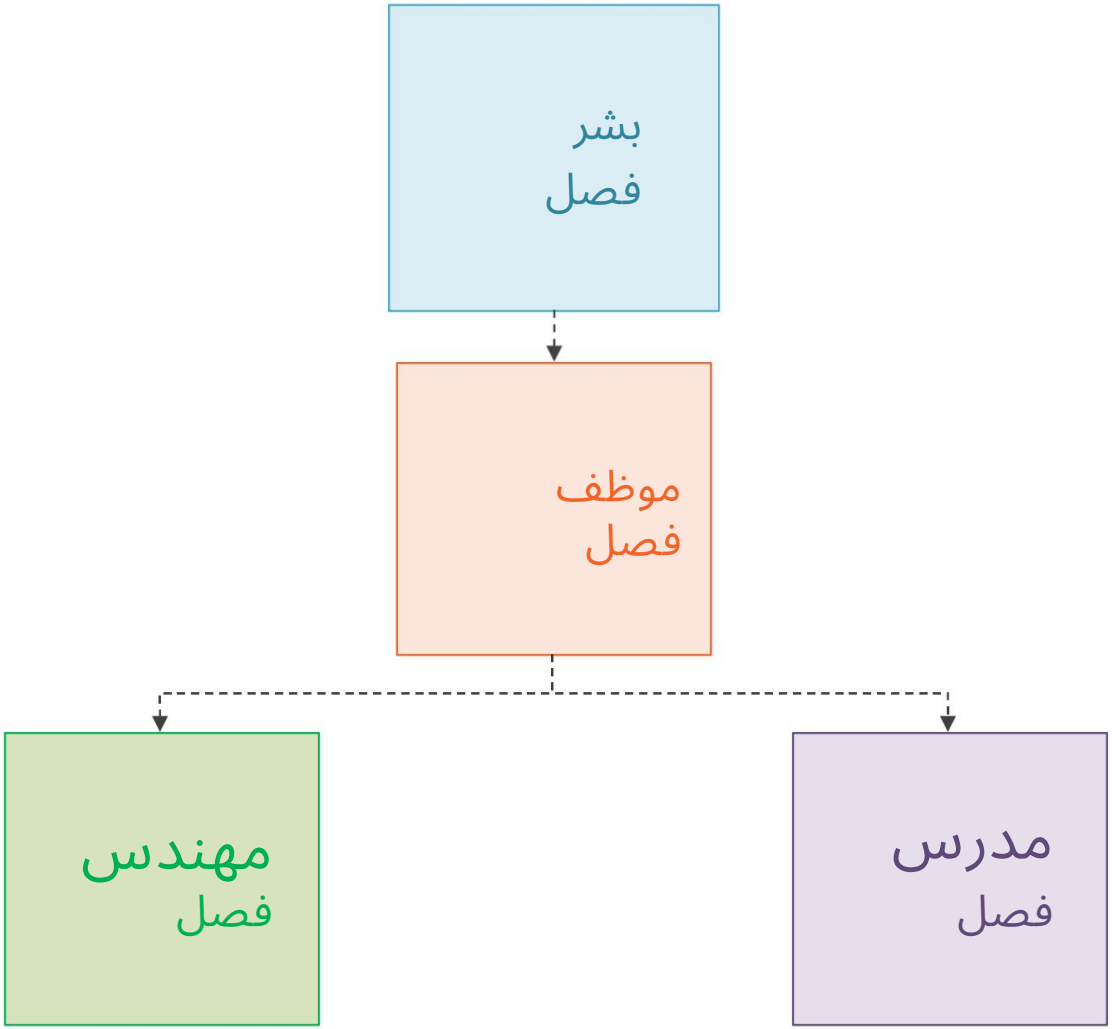
def walk (cls): print ("Walk...")

classmethod

المشي البشري ()

مفاهيم OOP

میراث



مثال

فئة الإنسان:

الحرف الأول :(self، name) __ __

الاسمself.name =

مواطنه يتحدث (النفس):

print ("My Name " + self.name) ؛

موظف فئة (بشري):

init def(الذات ، الاسم ، الراتب): __ __

سوبر (موظف ، ذاتي). الحرف الأول (الاسم)

الراتب الذاتي =الراتب

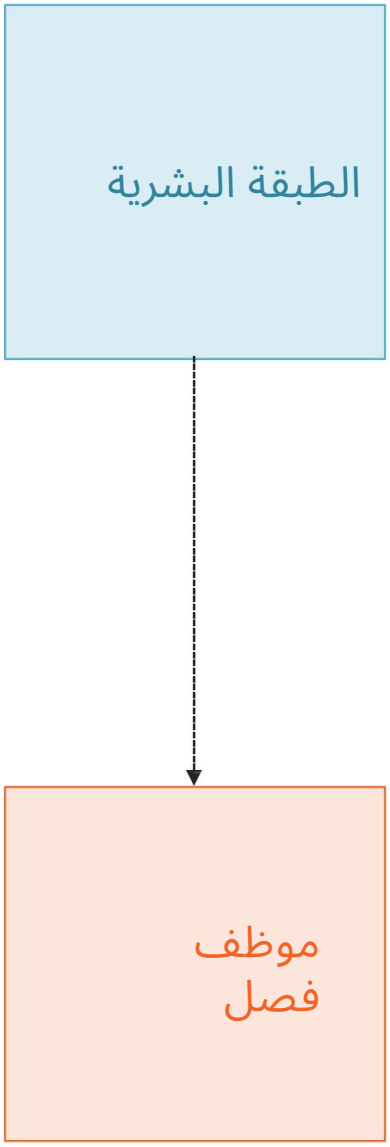
العمل def(الذات):

طباعة ("أنا أعمل الآن") ؛

emp =موظف ("أحمد" ، 500)

emp.speak ()

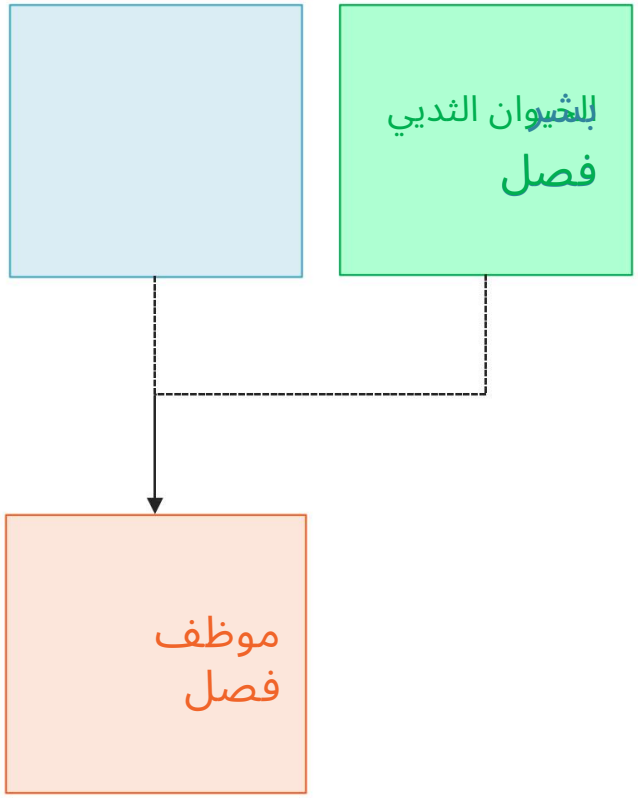
emp.work ()



تعدد الميراث

تدعم Pythonالوراثة المتعددة

تقرير**:



1-كيف تتعامل الوظيفة الفائقة مع الوراثة المتعددة .

2-إذا كان الإنسان والثديي لهما نفس الطريقة

مثل أكل ولكن مع تنفيذ مختلف. متى

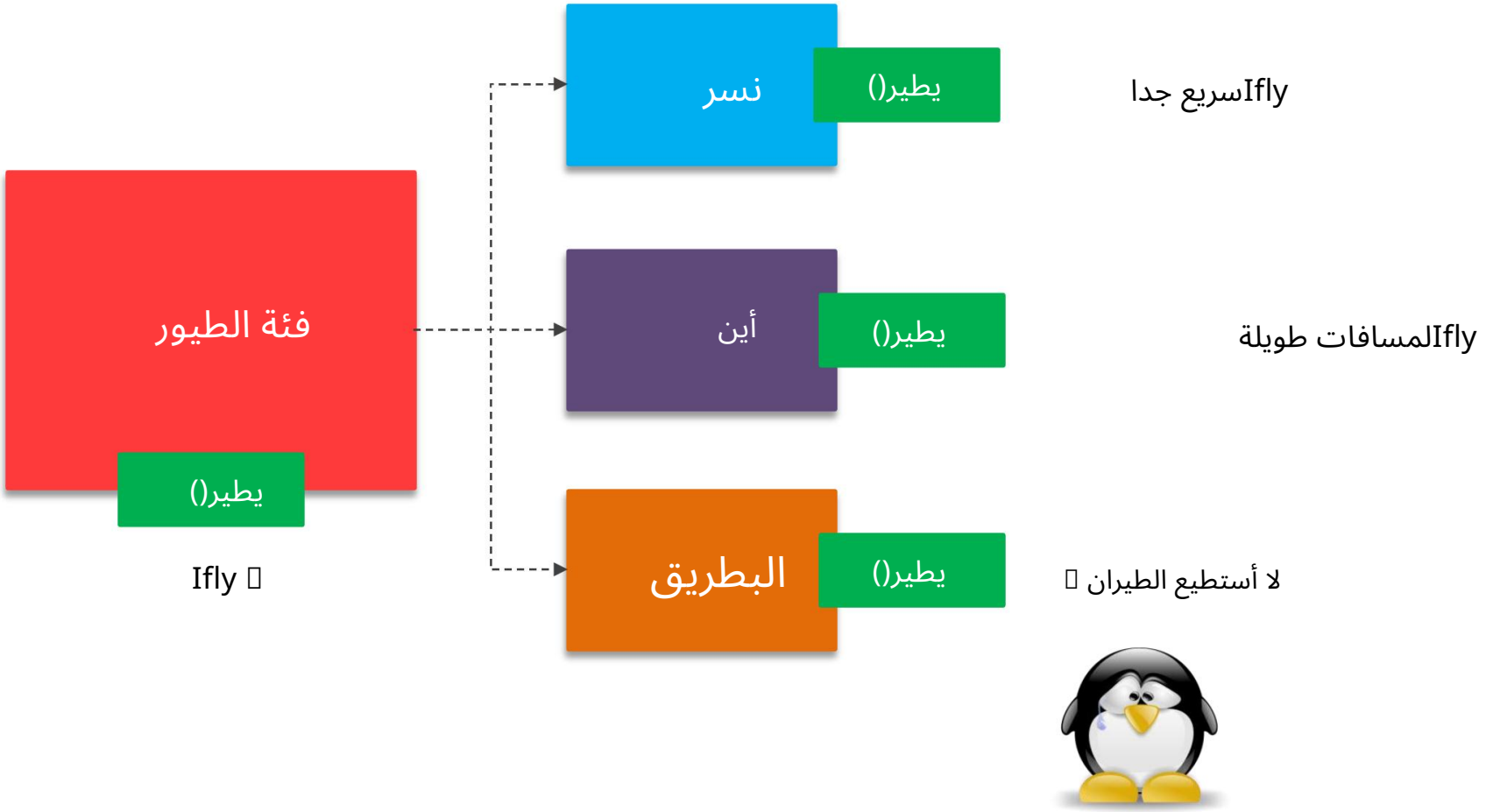
الطفل [الموظف] طريقة استدعاء المقعد كيف بيثون

التعامل مع هذه الحالة.

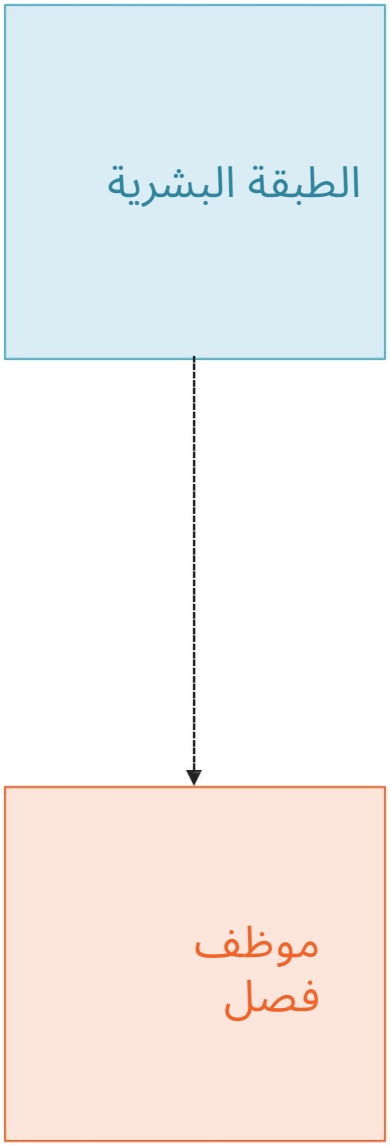
**أثبت رأيك بالأمثلة.

تعدد الأشكال

Polyتعني "كثير" و Morphismيعني "أشكال". قد تحدد الفئات المختلفة نفس الطريقة أو الخاصية.



أسلوب التجاوز



فئة الإنسان:

الحرف الأول :(self، name) __ __

self.name = الاسم

مواطنه يتحدث (النفس):

print ("My Name is " + self.name) ؛

موظف فئة (بشري):

def init(الذات ، الاسم ، الراتب): __ __

سوبر (موظف ، ذاتي). الحرف الأول (الاسم)

الراتب الذاتي =الراتب

مواطنه يتحدث (النفس):

طباعة ")راتبي " + "الراتب الذاتي) ؛

emp = موظف ("أحمد" ، (500

emp.speak () راتبي هو 500

طريقة التحميل الزائد

تقرير**:

هل يمكننا القيام بالحمل الزائد في بايثون؟

إذا كانت الإجابة بنعم ، قل لي كيف ؟؟

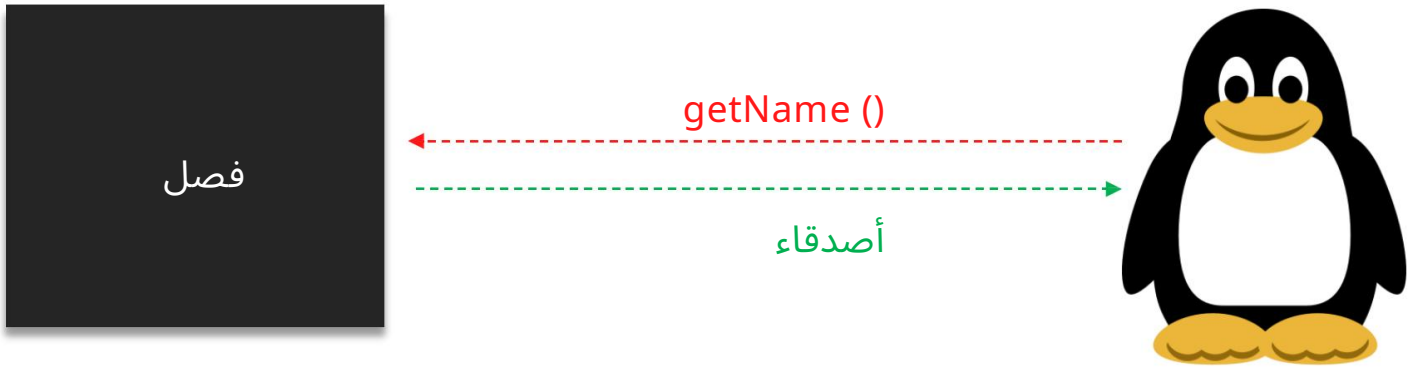
إذا كانت الإجابة لا ، أخبرني لماذا ؟؟

**دعم YourAnswer من خلال الأمثلة.

التغليف

مقدمة

التغليف هو تجميع البيانات والوظائف في مكون واحد (على سبيل المثال ، فئة) ثم التحكم في الوصول إلى هذا المكون.



مثال

```
فئة الإنسان:

    الحرف الأول :(self، name)

        الذات. الاسم =الاسم

    def getName(ذاتي):

        العودة الذاتية. اسم

man = Human(“Mahmoud”)

طباعة (اسم الرجل )

AttributeError: الكائن "البشري" ليس له سمة "اسم"

طباعة (man.getName ())

#الإخراج: محمود
```

@ملكية

```
فئة الإنسان:

    def init(الذات ، العمر): self.age = age

    property defالعمر (الذات): العودة الذاتية. عمر

    ___

    age.setter def Age @(الذات ، العمر): إذا كان العمر > 0:
        إذا كان العمر <= 0: العمر =العمر النفس. العمر

    ___

    ___ = 0 الذات. العمر

man = الإنسان() # 0 print (man.age) -25 =

man.age 23 # print (man.age) (23)
```

طرق خاصة

أسلوب خاص يتحكم في كيفية معاملة الكائن على أنه قابل للطباعة

فئة الإنسان:

الحرف الأول :(self، name)

الاسمself.name =

def str(ذاتي):

إرجاع "مرحبًا ، أنا إنسان واسمي self.name + "

man = Human("Ahmed")

طباعة (رجل)

#الإخراج: <كائن بشري رئيسي على 0x000000FD81804400>

طباعة (رجل)

#الإخراج: مرحبًا ، أنا إنسان واسمي أحمد

طريقة خاصة تتحكم في كيفية عرض الكائن على أنه قابل للاستدعاء

```
فئة الإنسان:

    الحرف الأول (self، name):

        self.name = الاسم

    نداء def (ذاتي):

        طباعة ("لقد اتصلت بي!")

man = Human("Ahmed")

رجل()

#output: TypeError: كائن "الموظف" غير قابل للاستدعاء

رجل()

#الإخراج: اتصلت بي!
```

طريقة خاصة تتحكم عند قياس طول الكائن

```
فئة الحيوان:

    def init(النفس ، الساقين):
        النفس.الساقين = الساقين

ديف لين (النفس):
    عودة النفس.الساقين

كلب = حيوان (4)
لين (كلب)
#output: TypeError: الكائن "الموظف" لا يحتوي على len ()
لين (كلب)
#الإخراج: 4
```

النصائح والحيل

تعابير لامدا

تستخدم تعبيرات Lambdaلعمل دوال مجهولة

إدخال لامدا : الإخراج

مثال

```
lmdaFn = lambda x: x + 4

lmdaFn (3) # 7

def sumFn (n):
    عودة لامدا x: x + n

sumFn (5) # <function....>

sumFn (5) (4) # 9
```

التكرارات (التكرار والتالي)

iterلتوليد مكرر من التكرارات.

يتم استخدام nextلإرجاع التكرار التالي من التكرارات.

مثال

```
l = ["JavaScript" ، "Python" ، "Java"]
#مترجمة
it = iter(l)
#تحويل قابل للتكرار إلى مكرر

التالي (هو)

#output:"جافا سكريبت"

التالي (هو)
#output:"بايثون"

التالي (هو)
#output:"جافا"
```

يتم استخدامه لتوليد التكرارات

مثال

```
def genFn ():
    لأني في النطاق :(5)
        العائد أنا
z () = genFn
```

#الإخراج: 0

#الإخراج: 1

التالي (ز)

التالي (ز)

```
def nonGenFn ():
    لأني في النطاق :(5)
        العودة أنا
من () = nonGenFn
```

التالي

TypeError: الكائن "int" هو

ليس مكررًا

وظيفة الخريطة

الخريطة (الوظيفة ، التسلسل)

تُستخدم وظيفة الخريطة لجعل المتغيرات من تطبيق الوظيفة المحددة على كل عنصر في التسلسل المحدد

---------- مثال ----------

```
it = الخريطة ([1،3،5]، lambda x: x + 4)
```

لأني فيه : _

طباعة (ط)

```
# 4
# 7
# 9
```

وظيفة التصفية

مرشح (عطل ، تسلسل)

تُستخدم وظيفة التصفية لإنشاء عناصر متكررة من تصفية كل عنصر في التسلسل المحدد بواسطة الوظيفة المحددة

مثال

it = مرشح ((lambda x: x٪ 5 == 0، [-15، -8، -5، 3، 5، 9، 25])

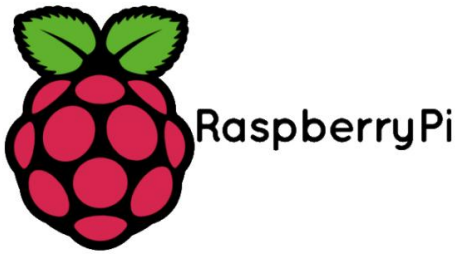
لأني فيه : _

طباعة (i ، end = "، ")

-15 ، -5 ، 5 ، 25

ماذا بعد ؟

الأطر والمكتبات



شكرًا لك