معهد العراب العالي للتدريب

اساسيات برمجة الحاسب بلغة البايثون

102سيبر

م. سمية أحمد

نبذة عن المقرر

تتعرف المتدربة من خلال هذا المقرر على مجموعة من المهارات الأساسية في برمجة أجهزوة الكمبيوتر بإستخدام البايثون. حيث تغطي اساسيات كيفية إنشاء برنامج من سلسلة من التعليمات البسيطة في بايثون

الأهداف العامة والتفصيلية من المقرر

الهدف العام:

يهدف هذا المقرر الى اكساب المتدرب المهارات الأساسية في البرمجة غرضية التوجيه

الأهداف التفصيلية:

- يحدد معنى العمليات والتعبيرات المنطقية المختلفة
- يتكلم عن أساليب استخدام الطرق Methods و أنواعها
- يحدد مفهوم المصفوفات والغرض منها وانواعها
- يحدد مفاهيم مبادئ البرمجة وغرضية التوجيه OOP
- يبني الدوال Methods
- يستدعي الدوال Methods
- يمرر المتغيرات على الدوال Methods
- يبرمج حل مشكلة برنامج باستخدام البرمجة غرضية التوجيه OOP

معهد العراب العالي للتدريب

01

المحاضرة الأولى

مقدمة في لغة البايثون

2024

محتويات المحاضرة

1. ماهي البرمجة
2. ماهي Python
3. فوائد python
4. المتغيرات
5. المدخلات/ المخرجات
6. Logical Operators / Boolean Operator
7. الشرطية IF

ماهي البرمجة

البرمجة : هي عملية بناء برامج حاسوبية يستطيع اي كمبيوتر فهمها ، وذلك من اجل تنفيذ مهام محددة مقل العمليات الحسابية أو الرسم أو حفظ تخزين المعلومات•

البرمجة تعد من أهم مجالات علم الحاسوب لأنها تدخل تقريبًا ي كل مهام الحاسوب مهما اختلفت الصناعة او التطبيق ، فصناعة الصواريخ تعتمد على البرمجة

معهد العراب العالي للتدريب

Python

لغة برمجة عالية المستوي، تتميز بسهولة قراءتها، وكتابتها كذلك تعد لغة سهلة التعلم. وتعتبر لغة برمجه كائنية التوجة ذات مصدر مفتوح وتعبر قابلة للتطوير. كذلك تعد لغة البرمجة بايثون لغة تفسيرية وذات أغراض متعددة ولها العديد من الاستخدامات في عدة مجالات، مثل انشاء البرامج المستقلة بواسطة الواجهات الرسومية، كذلك تستخدم في انشاء برامج الويب. وتعتبر من أسهل اللغات البرمجية تعلما. مكان الصور ان وجدتابتكرها وطورها جايدو ڤان أواخر ثمانينات القرن الماضي في مركز العلوم والحاسب الألي بأمستردام، وقد تم الإعلان عنها لأول مره عام 1991.
كما أنه تم كتابة نواتها بلغة البرمجة.

ما فوائد python

لغة بايثون تقدم العديد من المميزات لمستخدميها، وأهم هذه المميزات هي-:

- تعتبر من اسهل اللغات التي يتم استخدامها في البرمجة، وهذا ما يجعلها لغة مناسبة جداً للمبتدئين
- تحتوي على مجموعة بسيطة وغير معقدة من الجمل
- مصادرها متاحة مجاناً ولست في حاجة لدفع الأموال للحصول عليها، كما أن التحديثات الخاصة بها متاحة أيضاً ويمكنك الحصول عليها بسهولة.ومعرفة آخر التطورات التي تلحق بها.
- يعمل المبرمجين على تطوير هذه اللغة يوماً عن يوم

المتغيرات

برنامج لحساب مساحة المستطيل

- المدخلات (Input)
- خرجات (Output)

```
l= int(input("enter length: "))
w= int(input("enter width: "))
area = l*w
print("the area is: ",area)
```

Logical Operating

Logical Operation	Operator	Example
المساواة	==	if x == 5
أقل من	<	if x < 5
أقل من أو يساوي	<=	if x <= 5
أكبر من	>	if x > 5
أكبر من أو يساوي	>=	if x >= 5

Boolean Operating

Boolean Operation	Operator	Example
كل القيمتين يجب ان تكون صحيحة لنتيجة صحيحة	AND	if x >= 5 **AND** x <= 100 Returns TRUE if x is a number between 5 and 100
يكفي الحصول على عبارة واحدة صحيحة لنتيجة صحيحة	OR	if x == 1 **OR** x == 10 Returns TRUE if X is 1 or 10
للحصول على نتيجة صحيحة يجب أن تكون العبارة خاطئة	NOT	if **NOT** y Returns TRUE if the y value is False

تمارين

```
a = 1
if a == 1 or a > 10:
    print("a is either 1 or above 10")
```

تمارين

```
name = "bob" hungry = True
if name == "bob" and hungry == True:
     print("bob is hungry")
elif name == "bob" and not hungry:
     print("Bob is not hungry")
else: # If all other if conditions are not met
     print("Not sure who this is or if they are hungry")
```

IF الشرطية

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: -16
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
Negative changed to zero
```

IF الشرطية

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

تمارين

```
x = float (input("Enter the 1st number: "))
y = float (input("Enter the 2nd number: "))
op = input("Enter your operathion {+,-,*,/}: ")
z = 0

if (op == "+"):
    z = x + y
    print ("x + y =", z)

elif (op == "-"):
    z = x - y
    print ("x - y =", z)

elif (op == "*"):
    z = x * y
    print ("x * y =", z)

elif (op == "/"):
    z = x / y
    print ("x / y =", z)

else:
    print ("Error,,, Enter your operathion {+,-,*,/}:")
```

سؤال واجابة

- لغة أنشأت منها لغة البايثون وتعتبر نواة (السي ، السي بلس، الجافا ، الفيجوال بيسك)
- عملية بناء وتصميم برامج حاسوبية يستطيع أي كمبيوتر فهمها (الترجمة ، البرمجة ، التشفير ، الحماية)

معهد العراب العالي للتدريب

مراجعة عامة

2024

معهد العراب العالي للتدريب

102سيبر
اساسيات برمجة الحاسب بلغة البايثون

م: سمية أحمد

معهد العراب العالي للتدريب

02

المحاضرة الثانية

أنواع البيانات

2024

محتويات المحاضرة

1. المتغرات
2. المعاملات
3. الادخال و الإخراج

IF الشرطية

```
name = "bob" hungry = True
if name == "bob" and hungry == True:
    print("bob is hungry")
elif name == "bob" and not hungry:
    print("Bob is not hungry")
else: # If all other if conditions are not met
    print("Not sure who this is or if they are hungry")
```

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: -16
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
Negative changed to zero
```

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

```
x = float (input("Enter the 1st number: "))
y = float (input("Enter the 2nd number: "))
op = input("Enter your operathion {+,-,*,/}: ")
z = 0

if (op == "+"):
    z = x + y
    print ("x + y =", z)

elif (op == "-"):
    z = x - y
    print ("x - y =", z)

elif (op == "*"):
    z = x * y
    print ("x * y =", z)

elif (op == "/"):
    z = x / y
    print ("x / y =", z)

else:
    print ("Error,,, Enter your operathion {+,-,*,/}:")
```

المتغيرات في البايثون

المبرمج غير مسؤول عن تحديد أنواع المتغيرات التي يعرفها في برنامجه. فعليا, عندما تقوم بتعريف متغير و تضع فيه أي قيمة, سيقوم مفسر لغة بايثون بتحديد نوع هذا المتغير بناءا على القيمة التي أسندتها إليه بشكل تلقائي وقت التشغيل.

في بايثون يجب إسناد قيمة إلى المتغير أثناء تعريفه

```
var = 5          # هنا قمنا بتعريف متغير إسمه var و قيمته 5
print(var)       # هنا قمنا بطباعة قيمة المتغير var
```

```
x = y = z = 10       # هنا قمنا بتعريف ثلاث متغيرات قيمتها 10

print('x = ', x)     # هنا قمنا بطباعة قيمة المتغير x
print('y = ', y)     # هنا قمنا بطباعة قيمة المتغير y
print('z = ', z)     # هنا قمنا بطباعة قيمة المتغير z
```

معرفة نوع المتغير

لمعرفة نوع أي متغير يمكنك إستخدام الدالة () Type

تذكر: نوع المتغير في بايثون غير ثابت لأنه يتغير بشكل تلقائي على حسب نوع القيمة التي يتم تخزينها فيه

```
var = 10            # var هنا وضعنا رقم في المتغير
print(type(var))    # هنا طبعنا نوع قيمة المتغير var. لاحظ أن نوعها سيكون int لأنها عبارة عن رقم

var = 'harmash'     # var هنا وضعنا نص في المتغير
print(type(var))    # هنا طبعنا نوع قيمة المتغير var. لاحظ أن نوعها سيكون str لأنها عبارة عن نص
```

أنواع المتغيرات

تنقسم أنواع المتغيرات في بايثون إلى 6 أنواع أساسية و هي:

- أرقام – Numbers
- نصوص – Strings
- منطقية – Booleans
- مصفوفات ليس لها حجم ثابت يقال لها Lists
- مصفوفات حجمها و قيمها ثابتة, و غير قابلة للتغيير يقال لها Tuples
- مصفوفات ليس لها حجم ثابت, و لا يمكن حذف قيمها, و يمكن إضافة قيم جديدة فيها يقال لها Sets

تخزين البيانات في List

الـ Listعبارة عن مصفوفة حجمها غير ثابت و يمكنها تخزين قيم من مختلف الأنواع في وقت واحد.

في بايثون نستخدم الرمز [ ]

لتعريف مصفوفة أحادية (أي ذات بعد واحد) ليس لها حجم ثابت.

```
A = []                                  # هنا قمنا بتعريف مصفوفة فارغة
B = [10, 20, 30, 40, 50]                # هنا قمنا بتعريف مصفوفة تحتوي على أعداد صحيحة فقط
C = ['Mhamad', 'Samer', 'Abdullah']     # هنا قمنا بتعريف مصفوفة تحتوي على نصوص فقط
D = [1, 'two', 'three', 4]              # هنا قمنا بتعريف مصفوفة تحتوي على أعداد صحيحة و نصوص
```

تخزين البيانات في Tuple

الـ Tupleعبارة عن مصفوفة حجمها ثابت و قيمها غير قابلة للتغيير و يمكنها تخزين قيم من مختلف الأنواع في وقت واحد.

في بايثون نستخدم الرمز ( )

لتعريف مصفوفة أحادية (أي ذات بعد واحد) حجمها و قيمها ثابتة

تخزين البيانات في Tuple

A= ()

B= (10, 20, 30,40,50,60)

C= (‘Muhamad’, ‘Alen’ , ‘Abdullah’)

D= (1, ‘Ali’ , ‘sara’ , 4)

```
# هنا قمنا بتعريف مصفوفة ليس لها نوع محدد و تتألف من 4 عناصر
languages = ('Arabic', 'French', 'English', 'Spanish')


# هنا قمنا بعرض قيم المصفوفة و عدد عناصرها
print('Stored languages:', languages)
print('Number of stored languages is:', len(languages))
```

مفهوم العوامل

وامل - operators عبارة عن رموز لها معنى محدد و يمكننا تقسيمها إلى 5 مجموعات أساسية هي:

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Arithmetic Operators

==	(a == b)	هل قيمة a تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True
!=	(a != b)	هل قيمة a لا تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True
>	(a > b)	هل قيمة a أكبر من قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True
<	(a < b)	هل قيمة a أصغر من قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True
>=	(a >= b)	هل قيمة a أكبر أو تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True
<=	(a <= b)	هل قيمة a أصغر أو تساوي قيمة b ؟ إذا كان الجواب نعم فإنها ترجع True

```
a = 10
b = 10
c = 20

# الشرط التالي يعني أنه إذا كانت قيمة المتغير a تساوي قيمة المتغير b سيتم تنفيذ أمر الطباعة
if a == b:
    print('a = b')

# الشرط التالي يعني أنه إذا كانت قيمة المتغير a تساوي قيمة المتغير c سيتم تنفيذ أمر الطباعة
if a == c:
    print('a = c')
```

إدخال البيانات

الدالة input ()

لجعل المستخدم قادر على إدخال بيانات في البرنامج

في كل مرة تقوم فيها باستدعاء هذه الدالة يقوم مفّسر لغة بايثون بانتظارك لإدخال ما تريد من لوحة المفاتيح

بعد الإنتهاء من الإدخال و النقر على الزر Enterسيتم إرجاع الشيء الذي قمت بإدخاله كنص في المكان الذي تم منه إستدعاء الدالة.

ملاحظة: عند استدعاء الدالة () input فإنك حتى لو قمت بإدخال رقم فإنها سترجعه كنص.

لذلك في حال كنت تريد من المستخدم أن يدخل رقم, سيكون عليك تحويل ما ترجعه الدالة لرقم

```
# name الإسم الذي سيدخله سيتم تخزينه في المتغير ا بإظهار رسالة تطلب من المستخدم أن يدخل إسمه.
name = input("What's your name? ")

# name هنا قمنا بعرض جملة ترحيب مبنية على إسم المستخدم الذي قمنا بتخزينه قبل قليل في المتغير
print("Nice to meet you", name)
```

```
# a الذي سيدخله سيتم تحويله من النوع str إلى النوع int. بعدها سيتم تخزين العدد في المتغير
a = int(input("Enter a: "))

# b الذي سيدخله سيتم تحويله من النوع str إلى النوع int. بعدها سيتم تخزين العدد في المتغير
b = int(input("Enter b: "))

# هنا قمنا بعرض ناتج جمع العددين اللذين تم إدخالهما
print('a + b =', a + b)
```

إخراج البيانات

دالة الطباعة : عند استخدام دالة الادخال input يتم اعطاء خاصية بالكتابة بين علامات الاقواس ويتم طباعتها

ودالة الطباعة في لغة البايثون هى print

وسوف نستعملها فى طباعة بعض الجمل التعبيرية

```
text = input("Enter your name :")
print("welcom :",text)
```

```
Enter your name :"abdallah ahmed"
welcom :abdallah ahmed
```

معهد العراب العالي للتدريب

النـهايـة

مراجعة سريعة

2024

سؤال و إجابة

عبارة عن مصفوفة حجمها ثابت وقيمها غير قابلة للتغيير

List , Tuple , set , Directory

لمعرفة نوع أي متغير يمكنك إستخدام الدالة

Print() , Input() , Int() , Type()

معهد العراب العالي للتدريب

شكرا

معهد العراب العالي للتدريب

اساسيات برمجة الحاسب بلغة البايثون

102 سيبر

م. سمية أحمد

معهد العراب العالي للتدريب

المحاضرة الثالثة

03

جمل التحكم في بايثون

2024

محتويات المحاضرة

1. For loop
2. While loop

ماهو loop

- هو عباره عن block of code يتم وضع فيه الكود يشتغل اكثر من مره الى ان يتوقف على شرط معين.
- مثلا كود يعد من ١ الى ١٠٠ ولكن بشرط يكون الرقم اصغر من ١٠٠ فقط ليستمر و اذا كان اكبر سيتوقف.

```
i = 0
for i in range (1, 101):
    print(i)
```

For Loop

تتضمن لغة البايثون تعبيرًا خاصًا بالحلقة التكرارية for يتم استخدامه لتكرار تنفيذ شيفرة برمجية ما داخل حلقة التكرار. تختلف طريقة كتابة الحلقة التكرارية في بايثون عن اللغات الأخرى بشكل مُلفت، حيث تعتمد جملة التكرار for في البايثون على كائن يكون قابل للمرور على عناصره، ويُسمى iterable object

For loop

لو أردنا مثلًا تكرار طباعة رقم ١ خمسة مرات، يجب علينا الحصول على كائن iterable يحتوي على خمسة عناصر، وفي هذه الحالة يُمكننا الاستفادة من الدالة المُضمنة تلقائيًا في البايثون range التي تُعتبر iterable كما في المثال التالي:

```
1  for r in range(5):
2      print(1)
3
```

```
4  # iteratable
5  for number in range(1, 6):
6      print(number)
7
8
9  # start: Y
10 # end: a (EOL)
11 for character in "Yahya":
12     print(character)
```

For loop

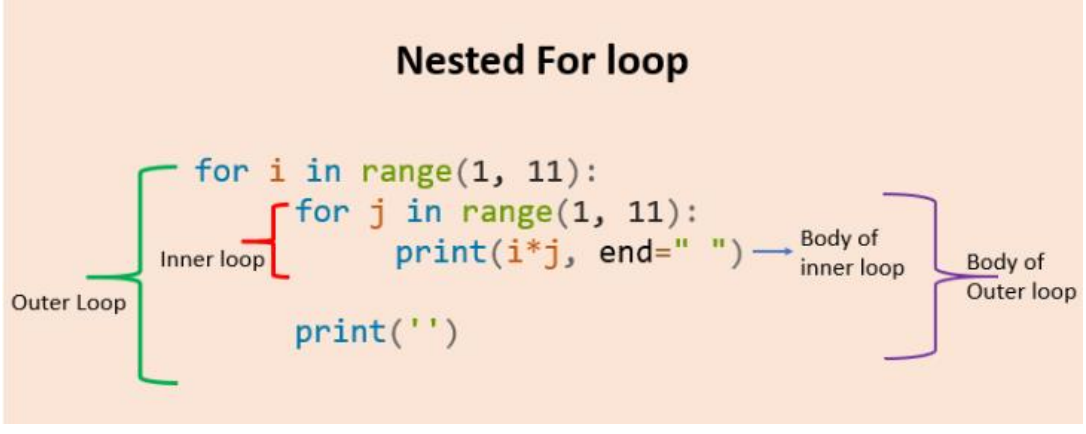
المثال التالي لحلقة تكرار هدفها المرور على عناصر قائمة ما وطباعة عناصرها، وبالطبع القائمة هنا أيضًا iterable:

```
1  for animal in ["dog", "cat", "mouse"]:
2
3      # You can use format() to interpolate formatted strings
4
5      print("{} is a mammal".format(animal))
```

Nested For loop

الحلقة المتداخلة هي حلقة داخل جسم الحلقة الخارجية. يمكن أن تكون الحلقة الداخلية أو الخارجية من أي نوع.

سيكون عدد التكرارات مساوياً لعدد التكرارات في الحلقة الخارجية مضروباً في التكرارات في الحلقة الداخلية.



For loop

في بعض الحالات قد نحتاج الى ايقاف حلقة التكرار عند تحقق شرط ما، في هذه الحالة، نسخدم تعبير break الذي يعمل على ايقاف حلقة التكرار كما في المثال التالي:

```
for animal in ["dog", "cat", "mouse"]:

    # You can use format() to interpolate formatted strings
    if animal == 'cat':
        break
    print("{} is a mammal".format(animal))
```

For loop

في جانب اخر قد نحتاج ان نتجاهل التكرار الحالي في حلقة التكرار، في المثال التالي نتجاهل قيمة cat ونعدي عنها بإستخدام تعبير **continue** :

```
for animal in ["dog", "cat", "mouse"]:

    # You can use format() to interpolate formatted strings
    if animal == 'cat':
        continue
    print("{} is a mammal".format(animal))
```

البيانات القابلة للمرور عليها بإستخدام جملة التكرار

كما يُمكننا المرور على عناصر قائمة من خلال حلقة التكرارfor ، يُمكننا المرور على المتغيرات ذات الأنواع التالية:

القاموس - dictionary

المجموعة - set

tuple

متغير نصي - string

الملفات - files

البيانات القابلة للمرور عليها بإستخدام جملة التكرار

```python
d = {1:'a',2:'b',3:'c',4:'d'}

#المرور على مفاتيح القاموس
for k in d:
    print(k)

#طريقة أخرى للمرور على مفاتيح القاموس
for k in d.keys():
    print(k)

#المرور على المفاتيح والقيم
for k,v in d.items():
    print(k,v)

#المرور على قيم القاموس فقط
for v in d.values():
    print(v)
```

القائمة السابقة ليست حصرية، ويُمكننا بناء أنواع بيانات خاصة بنا تكون iterable وقابلة للاستخدام مع for loop. المثال التالي يُوضح كيفية المرور على عناصر قاموس بعدة طُرق:

جملة التكرار While loop

نستخدم حلقة التكرار while لتكرار تنفيذ شيفرة برمجية، بنفس مفهوم جملة for ولكن الاختلاف هنا هو ان تكرار التنفيذ يستمر في حالة تحقق شرط جملة التكرار، وليس كما في for التي تُنفذ الشيفرة البرمجية بمقدار مُحدد أو مُعين.

```
1  x = 0
2
3  while x < 4:
4
5      print(x)
6      x += 1
```

جملة التكرار While loop

```
1   i = 0
2
3   while i < 10:
4       print(i)
5       if i == 5:
6           break
7       i += 1
8
9
10  i = 0
11
12  while i < 10:
13      if i == 3:
14          i += 1
15          continue
16
17      print(i)
18
19      if i == 5:
20          break
21      i += 1
```

يُمكننا استخدام كل من break و continue مع جملة التكرار while كما هو الحال في for. الأمثلة التالية توضح ذلك:

التمرين

جدول الضرب للعدد 2

```
for i in range(13):
    print(2,"*",i,"=",2*i)
```

For loop

```
for x in range(11):
    for y in range(13):
        print(x,"*",y,"=",x*y)
```

Nested for loop

التمرين

جمع ارقام موجبة

```
sum = 0
x = float(input("Enter the number"))
while x!=-1:
    sum = sum + x
    x = float(input("Enter the number"))
print("Sum = ", sum)
```

While loop

سؤال و إجابة

١	عبارة عن مصفوفة حجمها ثابت و قيمها غير قابلة للتغيير						
أ	List	ب	Tuple	ج	Set	د	Directory
٢	لمعرفة نوع أي متغير يمكنك إستخدام الدالة						
أ	Type()	ب	Int()	ج	Input()	د	Print()

معهد العراب العالي للتدريب

شكرا

معهد العراب العالي للتدريب

04

المحاضرة الرابعة

الدوال

2023

محتويات المحاضرة

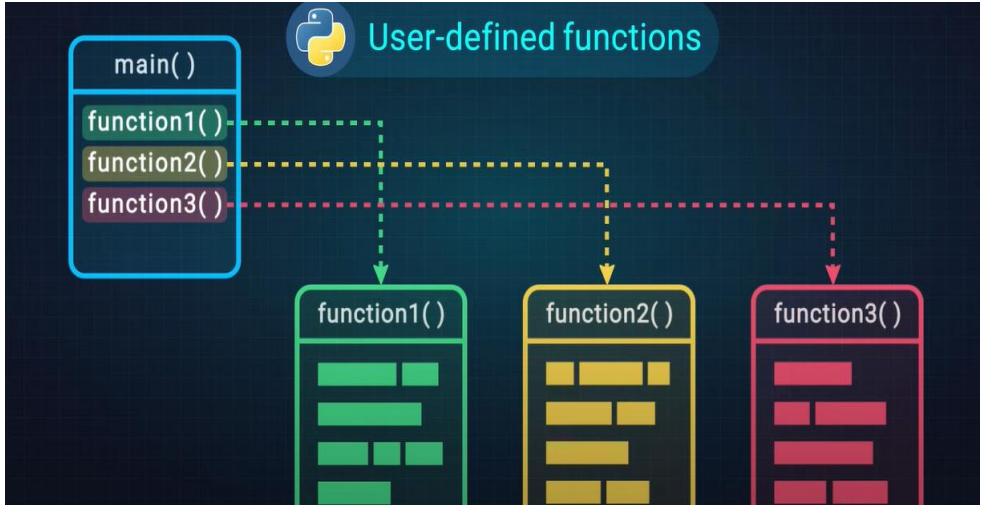
١. Function
٢. أنواع Function
٣. Scope
٤. Lambda function
٥. Array

Function

هي مجموعة من الأوامر البرمجية التي تؤدى وظيفة معينة
بايثون تحتوي على مجموعة كبيرة جداً من الدوال الجاهزة و التي سبق أن إستخدامنا بعضها مثل
الدوال **print()**
output()
Type()
وغيرهم من الدوال التي تطرقنا إليها في درس سابق.

أنواع Function

١. الدوال الجاهزة في بايثون، يقال لها **Built-in Functions**.
٢. الدوال التي يقوم المبرمج بتعريفها، يقال لها **User-defined Functions**.
   - ❖ دالة لا ترجع القيم
   - ❖ دالة ترجع القيم

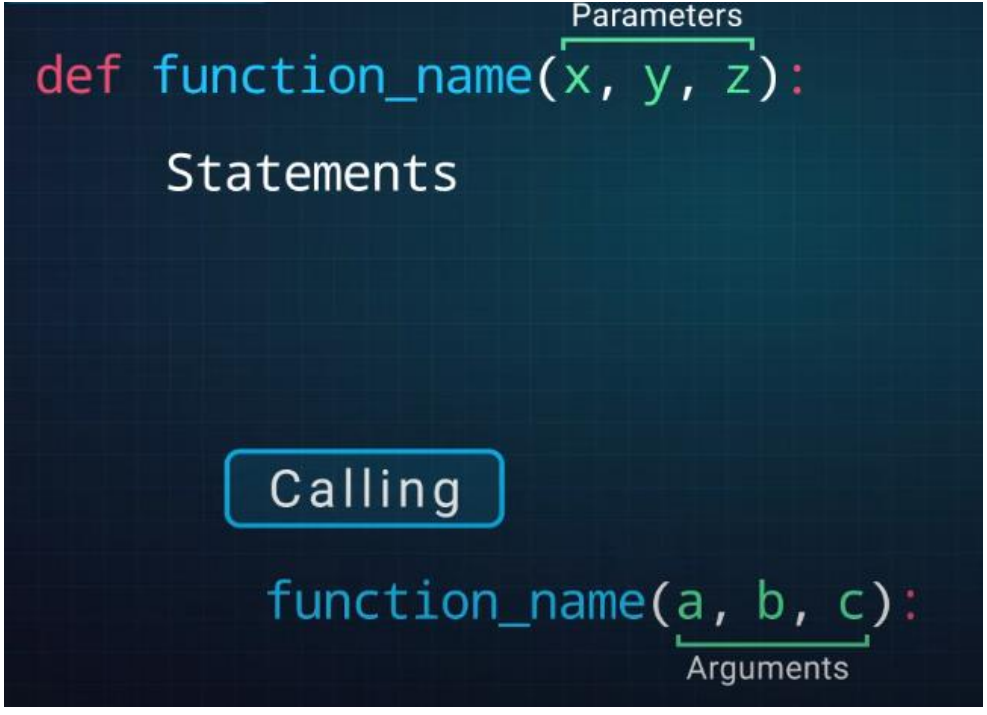


تكوين Function

```
def functionname():
    function_suite
```

- `def` : تعني أنك تعرف دالة جديدة.
- `functionname` : نضع مكانها الإسم الذي نعطيه للدالة, و الذي من خلاله يمكننا استدعاءها.
- `()` : بداخل القوسين يمكنك وضع باراميترات و يجب أن تضع `:` مباشرةً بعد القوسين و من ثم تنزل على سطر جديد لتبدأ بكتابة الأوامر التي ستتنفذ عند إستدعاء الدالة.
- `function_suite` : تعني الأوامر التي سنضعها في الدالة و التي ستتنفذ عند إستدعائها.

Function تكوين



دالة لا ترجع قيم

```
def printBill(item, price, qty):
    print(f"You bought {qty} {item}")
    print(f"Total {qty * price} $")


printBill("IPhone 14", 899, 3)
```

دالة ترجع قيم

```
def sum(n1,n2):
    sum= n1+n2
    return sum


amira = sum( n1: 3, n2: 5)
print(amira)
```

```
def add(*x):
    sum = 0
    for i in range(len(x)):
        sum = sum + x[i]
    return sum

print(add(5, 6, 5))
```

Lambda function

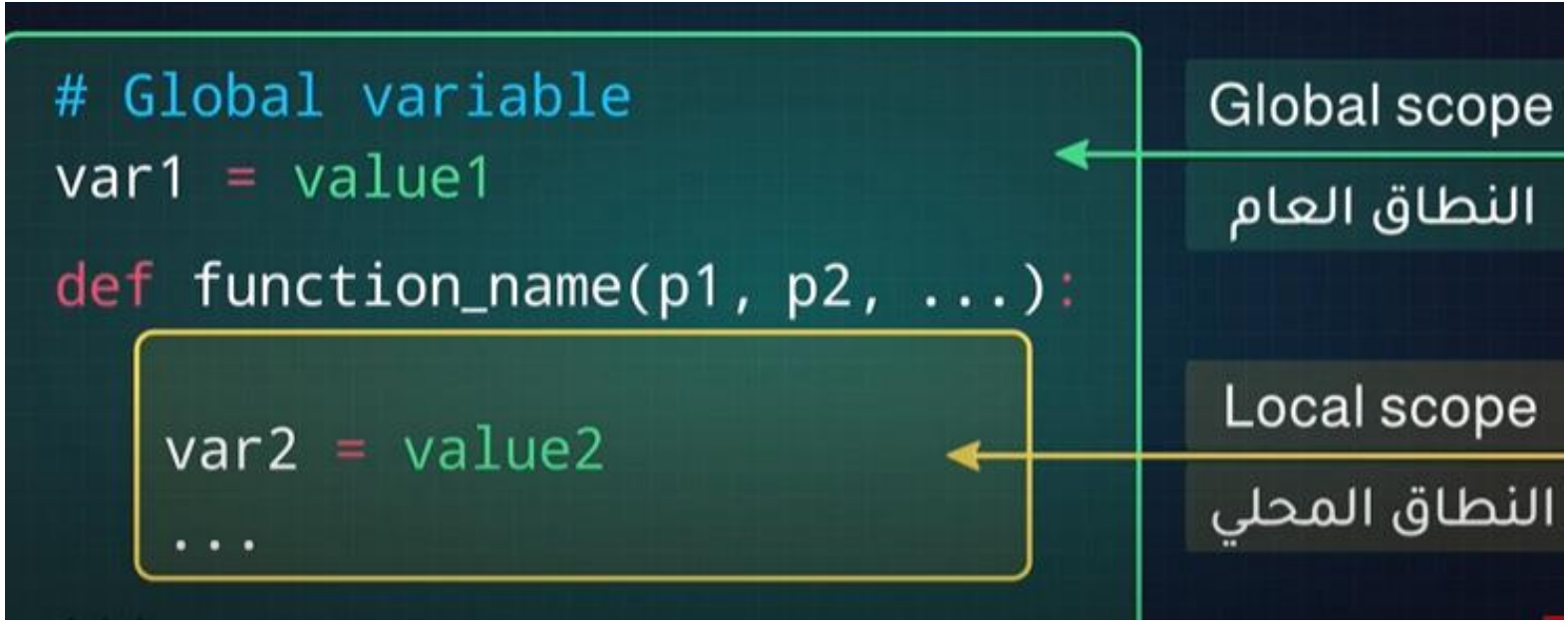
وظيفة لامدا هي وظيفة مجهولة صغيرة. يمكن لدالة لامدا أن تأخذ أي عدد من الوسائط، ولكن يمكن أن تحتوي على تعبير واحد فقط.

```
total = lambda price, tax : price + tax


print(total(150, 9))
```

الاختلافات الرئيسية بين دالة لامدا والدوال العادية هي بناء الجملة: تحتوي دالة لامدا على بناء جملة أكثر إيجازًا من الدوال العادية، مما يجعلها مناسبة للدوال البسيطة التي يمكن تعريفها في سطر واحد من التعليمات البرمجية.

Scope نطاق التأثير



تدريب: إضافة على القائمة

```
l = [3, 5, 6, 9, 8]

print(l)
```

```
l = [3, 5, 6, 9, 8]

print(l[1:4])
```

```
for i in range(1,11):
    l.append(float(input("Enter the number")))

print(l)
```

تدريب: لعبة التخمين

```
l = [2, 5, 7, 1, 9]

x = int(input("Enter the number"))


if x in l:
    print("Yes")
else:
    print("Not found")
```

سؤال و إجابة

١	عبارة عن مصفوفة حجمها ثابت و قيمها غير قابلة للتغيير						
أ	List	ب	Tuple	ج	Set	د	Directory
٢	لمعرفة نوع أي متغير يمكنك إستخدام الدالة						
أ	Type()	ب	Int()	ج	Input()	د	Print()

معهد العراب العالي للتدريب

شكرا

معهد العراب العالي للتدريب

05

المحاضرة الخامسة

البرمجة الكائنية

2023

محتويات المحاضرة

١. مفهوم OOP
٢. مفهوم Object
٣. مفهوم Class

OOP

البرمجة كائنية التوجّه
تتكون من class & object
فعند رؤيتك سيارات في الشارع من نفس النوع لكن بالوان مختلفة يمكن اعتبار السيارة (car) ب Class او الفئة ... وهذه الفئة تضم عدد من الكائنات لوكالات تجارية للسيارات ومنها فورد، هوندا، جمس، تويوتا. حيث يحتوي كل كائن او نوع من هذه السيارات على صفات تميز كل كائن او كل نوع عن غيره ، على سبيل المثال اللون color و السعر price وغيرها من الصفات.

اي ان لكل كلاس انواع مختلفة من الكائنات ولكل كائن ميزات مختلفة تختلف عن الكائنات الاخرى .
فعندما نريد ان نستدعي الكائن فورد من فئة car سيتم استدعاء جميع الميزات Attribute الخاصة بهذا الكائن .

Car class

Model, Price, Color, Build year

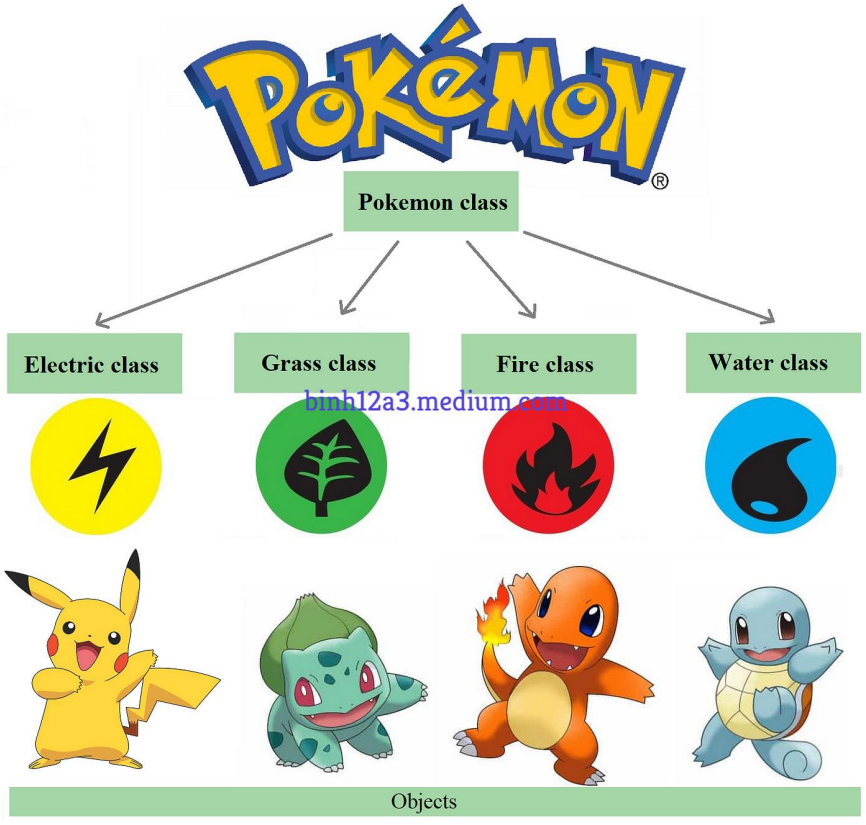
Objects

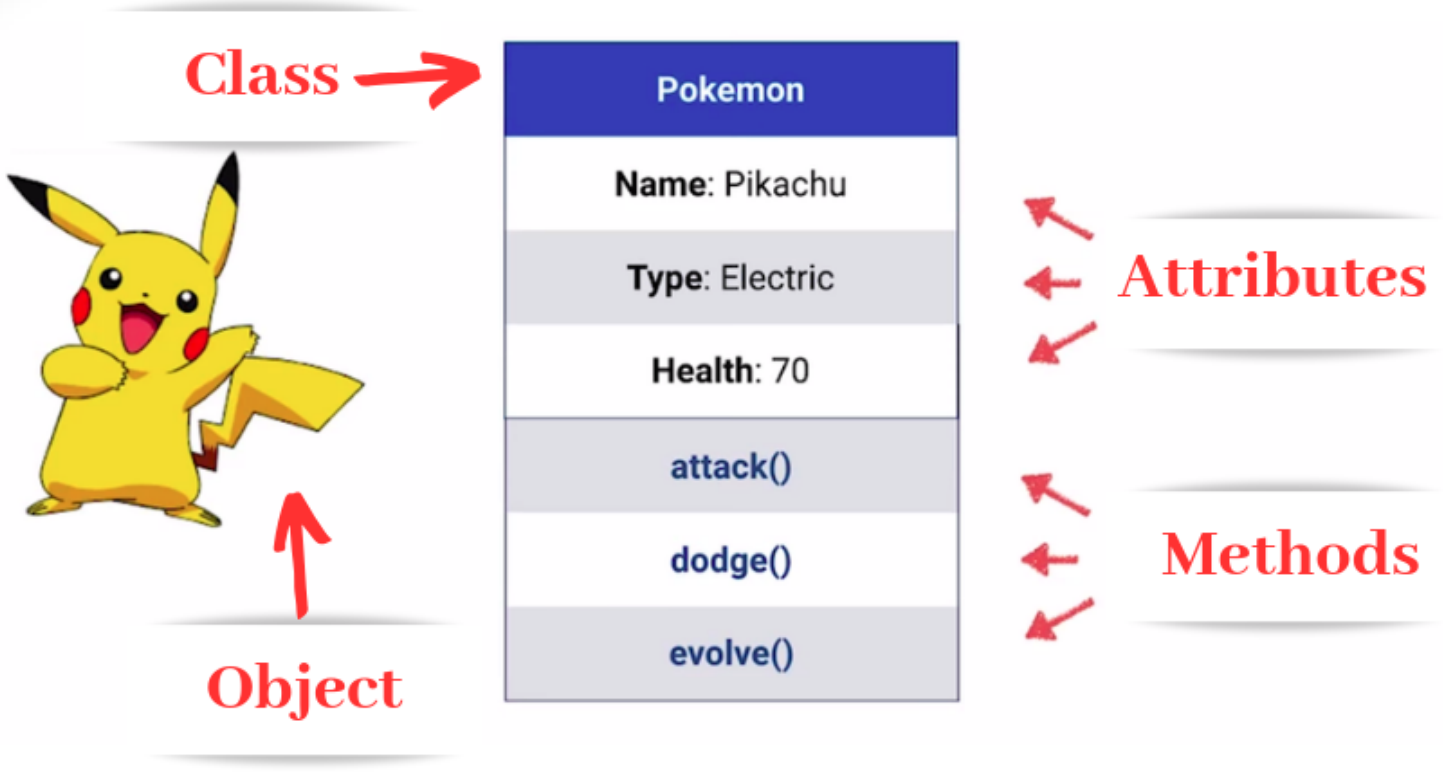
Model: AAA
Price: 10K
Color: Orange
Build year: 2015

Model: BBB
Price: 15K
Color: Blue
Build year: 2018

Model: CCC
Price: 45K
Color: Green
Build year: 2015

Class


Pokemon class
Electric class
Grass class
Fire class
Water class
binh12a3.medium.com
Objects


Class
Pokemon
Name: Pikachu
Type: Electric
Health: 70
attack()
dodge()
evolve()
Attributes
Methods
Object

Class

```
class Student:
    #Attributes (Instance Variables)

    #Constructors

    #Behaviour (Methods)
```


class Student:
lname = ""
fname = ""
def __init__(self, fname, lname):
self.__fname = fname
self.__lname = lname
#Behaviour (Methods)
def getName(self):
return self.fName + ' ' + self.lName

Student Class تمارين

Student.py

```python
class Student:

    lname = ""
    fname = ""

    def __init__(self, fname, lname):
        self.__fname = fname
        self.__lname = lname


    def getname(self):
        return self.__fname + " " + self.__lname

```

main.py

```python
from Student import Student

s1 = Student("Jouri", "Zahrani")
print(s1.getname())
```

Employee Class

Student.py empolyee.py

```
    fname = ""
    salary = 0

    def __init__(self, fname, lname, salary):
        self.__fname = fname
        self.__lname = lname
        self.__salary = salary


    def getname(self):
        return self.__fname + " " + self.__lname

    def salary(self):
        print("your salary ", self.__salary)

```

main.py

Visual layout of bidirectional text can depend on the base direction (View | Bidi Te› Ch

```
from Student import Student
from empolyee import empolyee

s1 = Student("Jouri", "Zahrani")
print(s1.getname())

e1 = empolyee("Jouri", "Zahrani", 10000)
print(e1.getname())
e1.salary()

```

Main

```python
from Student import Student
from Employee import Employee


s1 = Student('Ahmed', 'Ali', 111, 'ahmed@aa', Date(1,1,2000),3)
print(s1.getName())

e1 = Employee('Abdelghny', 'Orogat', 1223, 'aorogat@gmail.com', Date(1,1,1995), 1000)
print(e1.getName())
```

Inheritance التوريث

الكلاس المشترك Person

```python
from builtins import property
from Date import Date
class Person:
    #Constructor (Like Method)
    def __init__(self, fName, lName, id, email, date):
        #Instance variables
        self.__fName = fName
        self.__lName = lName
        self.__id = id
        self.__email = email
        self.__dateOfBirth = date


    #Behaviour (Methods)
    def getName(self):
        return self.__fName + ' ' + self.__lName

    def changeId(self, oldId, newId):
        if oldId == self.id:
            self.id = newId
```

Student Class بعد التوريث

```
from Date import Date
from Person import Person
class Student(Person):
    #Constructor (Like Method)
    def __init__(self, fName, lName, id, email, date, gpa):
        #Instance variables
        Person.__init__(fName, lName, id, email, date)
        self.__gpa = gpa
```

Employee Class بعد التوريث

```
from Date import Date
from Person import Person
class Employee:
    #Constructor (Like Method)
    def __init__(self, fName, lName, id, email, date, salary):
        #Instance variables
        Person.__init__(fName, lName, id, email, date)
        self.__salary = salary
```

Exercises

https://www.w3resource.com/python-exercises/

https://www.programiz.com/python-programming/online-compiler/

سؤال و إجابة

١	عبارة عن مصفوفة حجمها ثابت و قيمها غير قابلة للتغيير						
أ	List	ب	Tuple	ج	Set	د	Directory
٢	لمعرفة نوع أي متغير يمكنك إستخدام الدالة						
أ	Type()	ب	Int()	ج	Input()	د	Print()

معهد العراب العالي للتدريب

شكرا