

PYTHON COURSE

Python course , from scratch
to professionalism ...

Made by : Taha Khaldoun Amro

دورة تعليم لغة بايثون

النصوص

في بداية تعرفنا على عالم النصوص , يتم تعريف النصوص في لغة بايثون على أنها كل ما يوضع ما بين علامات التنصيص الثنائية ("") أو الأحادية (').

```
x = 'my string with single quote'
```

```
print(x)
```

```
x = "my string with double quote"
```

```
print(x)
```

كما يمكننا أن نلاحظ , فان استعمال علامات التنصيص الثنائية او الاحادية لن يختلف في النتيجة .

في الحقيقة يوجد هذا النظام فقط من أجل تمكين المستخدم من ابراز كلمة معينة في النص من خلال وضعها في علامات التنصيص الاخرى (غير التي استخدمها في تحديد النص).

دورة تعليم لغة بايثون

النصوص

```
x = 'mystring "oh yeaaah" '
```

```
print(x)
```

```
x = "mystring 'oh yeaaah' "
```

```
print(x)
```

من الأشياء الجديدة بالذكر أن المسافة يتم التعامل معها على أنها نص بنفسه (character) .

النصوص

من الأشياء الجميلة التي يمكننا فعلها بالنصوص هو كتابة النصوص في سطور متعددة , و ذلك من خلال وضع ثلاثة علامات تنصيص , و هكذا سيفهم الكود أن كل ما يكتب بين هذه الثلاث علامات سيكون نصاً واحداً :

```
x = """this is  
my multible  
line string"""  
# try replacing the double Quotes (""") with single Quotes ("  
print(x)
```

دورة تعليم لغة بايثون

تشرح النصوص و ترقيمها

يعتبر التشرح و الترقيم في لغات البرمجة من أهم الامور التي تساعد في التعامل مع البيانات , حيث أنه يمكننا من الوصول الى عنصر معين داخل النص من خلال موقعه , أو مجموعة من العناصر من خلال انشاء سلسلة تبدأ من أول عنصر و تنتهي بالعنصر الذي يلي اخر عنصر نريده (سيتم شرح الأمر لاحقاً).

1. الترقيم

لنقل ان لديك نصاً ("we love python") و تريد مثلاً الوصول الى العنصر الاول ("w") .
بكل بساطة يمكنك استخدام موقعه العددي داخل النص (index) , من أجل القيام بذلك نفذ الكود
الاتي :

```
x = 'we love python'
```

```
print(x[1]) # use the square brackets to access to any number in the string [int]
```

دورة تعليم لغة بايثون

تشرح النصوص و ترقيمها

لا بد أنك لاحظت أن الكود قام بطباعة الحرف ("e") , الخطأ ليس منك , بل هو في الواقع شيء مدمج في اللغة الا و هو مفهوم (index zero) أي القيمة صفر , حيث أن أي كائن في اللغة يحتوي على عناصر , يحمل العنصر الأول دائما الرقم "0" , لذا قم بتغيير القيمة [1] الى [0] .

بهذه الطريقة يمكننا الوصول الى أي عنصر داخل الكائن .

طبعا يوجد حيلة توفرها لنا لغة بايثون ألا و هي قدرتنا على الوصول الى اخر عنصر من خلال استخدام الرقم السالب , و لكن يجب أن ننتبه الى أن اخر عنصر يحمل قيمة [-1] و ليس [-0] و الذي يليه من النهاية يحمل رقم [-2] و هكذا ...

```
x = 'we love python'
```

```
print(x[-1]) # accessing the last letter 'n'
```

دورة تعليم لغة بايثون

تشرح النصوص و ترقيمها

2. التشرح (slicing)

تكمّن أهمية التشرح (slicing) في إمكانية وصولنا إلى مقطع كامل من النص (من حرف إلى حرف آخر) يتم ذلك من خلال تعيين موقع بدء و نهاية من خلال مواقع الأحرف . سنحاول طباعة "we lov"

```
x = 'we love python'
```

```
print(x[0:5]) # x[start index : end text]
```

لا بد أنك لاحظت مشكلة جديدة , لكن لا بأس , لغة بايثون تحتوي على مفهوم :

not including last index

و الذي يعني عدم تضمين آخر قيمة يعني أنك عندما تريد طباعة "we lov" يجب عليك أن تأخذ بعين الاعتبار أن العنصر (5) و الذي هو (v) لن يظهر إلا إذا استخدمت القيمة 6 كنهاية .

دورة تعليم لغة بايثون

تشرح النصوص و ترقيمها

بعض الأمور التي يجب الانتباه لها :

1. عند عدم تعيين قيمة البداية يتم البدء من القيمة 0
2. عند عدم تعيين قيمة النهاية يتم الانتهاء عند اخر قيمة

```
x = 'we love python'
```

```
print(x[:5]) # start from the index 0 untill the end
```

```
print(x[2:]) # start from index 2 untill the end
```


دورة تعليم لغة بايثون

تشرح النصوص و ترقيمها

أحد الأمور المهم تعلمها في عملية التشرح هي تحديد الخطوات (steps) و الذي يعني عدد العناصر التي يجب ان يقفز عنها .

القيمة الافتراضية للخطوات هي 1 , مما يعني انه سينتقل للعنصر الذي يليه فوراً .

```
x = 'we love python'
```

```
print(x[::1]) # x[the first index : the last index : steps]
```

```
print(x[::2]) # x[the first index : the last index : steps]
```