Cours 2b Introduction a la synthaxe python_part2

November 3, 2020

Table of Contents

1 Tour d'horizon de la syntaxe python : organiser son code (fonction, classes et bonnes pratiques, etc.)

Les fondamentaux de la syntaxe python sont bien résumés dans un poly qui n'est plus maintenu mais qui reste globalement à jour (Xavier Dupré - ENSAE). Il vous est fortement recommandé, par vous-mêmes, d'être curieux et d'aller fouiller sur le site de Xavier Dupré (une mine d'or !!!)

Dans ce second temps, nous allons nous intéresser à la bonne manière d'organiser son code : fonctions, classes et bonnes pratiques

http://www.xavierdupre.fr/app/teachpyx/helpsphinx/c_resume/python_sheet.html#le-langage

http://www.xavierdupre.fr/enseignement/initiation/resume_utile.pdf

1.1 Pourquoi organiser son code ?

1.1.1 Eviter les copiers-collers malheureux

[1]:
```
# Il vaut mieux éviter :

for i in [0,2,4,6] :
    print(i)
for i in [1,3,5,7]:
    print(i)
for i in ['a','a','b','b']:
    print(i)
for i in ['a','b','c','d']:
    print(i)
```

```
0
2
4
6
1
3
5
7
a
a
b
b
a
b
c
d
```

[2]:
```
#et préférer :

def print_i(liste):
    for i in liste :
        print(i)

print_i([0,2,4,6])
print_i([1,3,5,7])
print_i(['a','a','b','b'])
print_i(['a','b','c','d'])
```

```
0
2
4
6
```

```
1
3
5
7
a
a
b
b
a
b
c
d
```

1.1.2 Faciliter la lisibilité

"Un code est plus souvent lu qu'écrit !!!"

[3]:
```python
def print_each_element_of_the_list(liste_initiale):
    """Cette fonction a pour objectif de lister tous les elements passes en␣
 ↪parametre

    :param liste_initiale: liste dont les elements sont à afficher
    :type liste initiale: list
    """
    for element in liste_initiale:
        print(element)
    pass
```

[ ]:
```python
#Petit test : le maj + tab (pour obtenir rapidement de l'aide sur une fonction)
print_each_element_of_the_list(
```

2 Les fonctions : les règles de base

1) une fonction en python s'écrit comme suit :

`def <nom de la fonction>(<parametre>): <instructions> return <donnees a renvoyer>`

2) Une variable créée à l'intérieure d'une fonction n'existe pas à l'extérieur : c'est une variable locale

3) Au premier return rencontré, la fonction "s'arrête" (et renvoie la valeur prévue dans le return)

4) Sans le mot clef "return", la valeur retournée est None

Attention, le `return ""` ou le `return None` n'est pas un `return "Nothing"`

4) Une fonction peut être récursive et s'appliquer à elle-même

```
def recursive(z):     if X/2 < 1 :          return 1     else :          return
recursive(z/2)+1
```

5) Par convention, on note les fonctions en minuscules avec des underscores pour séparer les noms

[13]:
```
def add(x,y):
    if x > 0 :
        z=x+y
        return z
```

[15]:
```
def polynome_degre_deux(x, a, b, c):
    z = a*x^2 +b*x +c
    return z
```

[16]:
```
polynome_degre_deux(3,1,1,1)
```

[16]: 5

[14]:
```
def polynome(x, coefficient ):
    return sum([ x**i * c for i,c in enumerate(coefficient) ])
```

2.0.1 Exercice : prise en main des fonctions

[1]:
```
#Fonctions "triviales" : def afficher un parametre, etc.
```

[ ]:
```
#Cf. correction en cours
```

3 Les classes : vers la programmation orientée objet

3.0.1 Ex: Dessine moi une voiture

[1]:
```
#Creez un classe qui permette d'instancer des voitures. Les voitures se␣
↪caractérisent
#par une marque et par une vitesse

#NB : par convention, les classes s'écrivent en "CamelCase"
#Les "fonctions" présentent dans les classes sont appelées "méthodes"
```

[2]:
```
class Voiture() :
    def __init__(self, desc, vit):
        self.description = desc
        self.vitesse = vit
```

```
    def __del__(self):
        print("La voiture, " + self.description + " a été détruite")

    def accelerer(self, valeur):
        self.vitesse += valeur
        self.etat_des_lieux()

    def etat_des_lieux(self):
        print(self.description, "roule a", self.vitesse, "km/h")
```

[3]:
```
voiture1 = Voiture("Mercedes classe A", 80)
```

[4]:
```
voiture1
```

[4]:
```
<__main__.Voiture at 0x7f4b1c1f08d0>
```

[5]:
```
voiture1.etat_des_lieux()
```

```
Mercedes classe A roule a 80 km/h
```

[6]:
```
voiture1.accelerer(20)
```

```
Mercedes classe A roule a 100 km/h
```

[7]:
```
voiture1.accelerer(50)
```

```
Mercedes classe A roule a 150 km/h
```

[8]:
```
del voiture1
```

4 Autres trucs et astuces

Voir la fin du poly cheat sheet de Xavier Dupré. Des questions ?

5 Pour aller plus loin

https://www.codingame.com/