td1a_cenonce_session1

July 1, 2022

1 1A.0 - Premiers pas en Python

Questions très simples pour ceux qui savent coder, quelques éléments principaux du langage Python pour les autres.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.0.1 Partie 1

Un langage de programmation permet de décrire avec précision des opérations très simples sur des données. Comme tout langage, il a une grammaire et des **mot-clés**. La complexité d'un programme vient de ce qu'il faut beaucoup d'opérations simples pour arriver à ses fins. Voyons cela quelques usages simples. Il vous suffit d'exécuter chaque petit extrait en appuyant sur le triangle pointant vers la droite ci-dessus. N'hésitez pas à modifier les extraits pour mieux comprendre ce que le programme fait.

La calculatrice

[2]:
```
x = 5
y = 10
z = x + y
print(z)    # affiche z
```

```
15
```

On programme sert souvent à automatiser un calcul comme le calcul mensuel du taux de chômage, le taux d'inflation, le temps qu'il fera demain… Pour pouvoir répéter ce même calcul sur des valeurs différentes, il faut pouvoir décrire ce calcul sans savoir ce que sont ces valeurs. Un moyen simple est de les nommer : on utilise des variables. Une variable désigne des données. `x=5` signifie que la variable `x`contient 5. `x+3` signifie qu'on ajoute 3 à `x` sans avoir besoin de savoir ce que `x` désigne.

L'addition, l'incrémentation

[3]:
```
x = 2
y = x + 1
print(y)
x += 5
print(x)
```

```
3
7
```

Lorsqu'on programme, on passe son temps à écrire des calculs à partir de variables pour les stocker dans d'autres variables voire dans les mêmes variables. Lorsqu'on écrit `y=x+5`, cela veut dire qu'on doit ajouter 5 à `x` et qu'on stocke le résultat dans `y`. Lorsqu'on écrit `x += 5`, cela veut dire qu'on doit ajouter 5 à `x` et qu'on n'a plus besoin de la valeur que `x` contenait avant l'opération.

La répétition ou les boucles

[4]:
```
a = 0
for i in range (0, 10) :
    a = a + i             # répète dix fois cette ligne
print (a)
```

```
45
```

Le mot-clé `print` n'a pas d'incidence sur le programme. En revanche, il permet d'afficher l'état d'une variable au moment où on exécute l'instruction `print`.

L'aiguillage ou les tests

[5]:
```
a = 10
if a > 0 :
    print(a)     # un seul des deux blocs est pris en considération
else:
    a -=  1
    print(a)
```

```
10
```

Les chaînes de caractères

[6]:
```
a = 10
print(a)     # quelle est la différence
print("a")   # entre les deux lignes
s = "texte"
s += "c"
print(s)
```

```
10
a
textec
```

Toute valeur a un type et cela détermine les opérations qu'on peut faire dessus. `2 + 2` fait `4` pour tout le monde. `2 + "2"` fait quatre pour un humain, mais est incompréhensible pour l'ordinateur car on ajoute deux choses différentes (`torchon + serviette`).

[7]:
```
print("2" + "3")
print(2+3)
```

```
23
5
```

1.0.2 Partie 2

Dans cette seconde série, partie, il s'agit d'interpréter pourquoi un programme ne fait pas ce qu'il est censé faire ou pourquoi il provoque une erreur, et si possible, de corriger cette erreur.

Un oubli

[8]:
```
a = 5
a +=  4
print(a)   # on voudrait voir 9 mais c'est 5 qui apparaît
```

```
9
```

Une erreur de syntaxe

[9]:
```
a = 0
for i in range (0, 10):   # il manque quelque chose sur cette ligne
    a = a + i
print(a)
```

```
45
```

Une autre erreur de syntaxe

[10]:
```
a = 0
for i in range (0, 10):
    a = a + i                # regardez bien
print(a)
```

```
45
```

Une opération interdite

[11]:
```
a = 0
s = "e"
print(a + s)    # petit problème de type
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-6600ae759b48> in <module>()
      1 a = 0
      2 s = "e"
----> 3 print (a + s)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Un nombre impair de...

[12]:
```
a = 0
for i in range (0, 10) :
      a = (a + (i+2)*3  )       # comptez bien
print(a)
```

```
195
```

1.0.3 Partie 3

Il faut maintenant écrire trois programmes qui :

- Ecrire un programme qui calcule la somme des 10 premiers entiers au carré.
- Ecrire un programme qui calcule la somme des 5 premiers entiers impairs au carré.
- Ecrire un programme qui calcule la somme des qui 10 premières factorielles : $\sum_{i=1}^{10} i!$.

A propos de la parité :

[13]:
```
14%2, 233%2
```

[13]:
```
(0, 1)
```

1.0.4 Tutor Magic

Cet outil permet de visualiser le déroulement des programmes (pas trop grand, site original pythontutor.com).

[14]:
```
%load_ext tutormagic
```

[15]:
```
%%tutor --lang python3

a = 0
for i in range (0, 10):
    a = a + i
```

```
<IPython.lib.display.IFrame at 0x958af60>
```

Arriverez-vous à résoudre le première exercice du site CodinGame ?

[16]:
```

```