TD1. Premiers programmes

Bases de la Programmation Impérative – Chapitre 1 : bases
version du 7 octobre 2022

L'objectif de ce premier TD est d'être capable d'écrire **et de comprendre** les premiers programmes python que nous réaliserons en TP.

Attention : comme nous l'avons vu dans le premier cours, python est un langage de haut niveau. Par conséquent, bien que les programmes que nous allons voir dans ce premier TD soient petits et simples, ils cachent beaucoup de choses. L'objectif ici n'est pas de rentrer dans toutes ces choses cachées, mais de comprendre le nécessaire pour aborder les premiers TP.

Exercice 1 : portée des variables

On considère le programme suivant :

```
1  #!/usr/bin/env python3
2
3  une_var_glob = 7
4
5  def f(un_param):
6      une_var_loc = 4
7      une_autre_var_loc = un_param + une_var_loc
8      print(une_autre_var_loc)
9
10 f(une_var_glob)
11 print(une_var_glob)
```

Question 1

Comme nous l'avons fait en cours (diapositive 21 ici), exécutez le programme une instruction à la fois en reportant, sur papier, dans un tableau, le contenu des variables accessibles lors de l'exécution de ladite ligne. Ce tableau indiquera pour chaque variable : son nom, son type, sa valeur et sa portée. Notez également ce qui s'affiche sur la sortie standard.

Exercice 2 : portée des variables avec même nom

On considère le programme suivant :

```
1  #!/usr/bin/env python3
2
3  a = 7
4
5  def f(a):
6      b = 8
7      a = a + b
8      print(a)
9
10 a = 11
11 print(a)
12 f(a)
13 print(a)
```

Question 1

> Même question que dans l'exercice précédent.

Exercice 3 : typage dynamique

On considère le programme suivant :

```
#!/usr/bin/env python3

a = "1"
b = "3"
print(a + b)

a = 1
b = 3
print(a + b)

a = "1"
b = 3
print(a + b)

t1 = (1, 3)
print(t1[0] + t1[1])

t2 = (1, "3")
print(t2[0] + t2[1])
```

Question 1

> Même question que dans les exercices précédents.

Exercice 4 : mes propres types

On considère le programme suivant :

```
#!/usr/bin/env python3
"""Illustration des namedtuple"""

import collections

bob = ("Robert", 23)
bill = ("William", 47)

# affiche le prénom de bob et l'age de bill (beurk)
print(bob[0])
print(bill[1])

bill[1] = 43

Personne = collections.namedtuple("Personne", "prenom, age")

bob_nt = Personne("Robert", 23)
bill_nt = Personne("William", 47)

# affiche le prénom de bob et l'age de bill (plus lisible !)
print(bob_nt.prenom)
print(bill_nt.age)

bill_nt.age = 43
```

Question 1

> Même question que dans les exercices précédents.

Exercice 5 : et dans un autre langage, ça donne quoi ?

On considère le programme suivant :

```ruby
#!/usr/bin/env ruby

a = 2010

def f(p)
  puts "bienvenue "
  return p + 11
end

puts "à l'Ensimag "

a = f(a)
puts "en " + String(a)
puts "(signé Yoda)"
```

Question 1

> Même question que dans les exercices précédents.

Exercice 6 : et encore dans un autre langage, ça donne quoi ? (pour aller plus loin)

Question 1

> Réécrire le programme de l'exercice précédent dans le langage impératif de votre choix (autre que Python et Ruby, en C par exemple).

Question 2

> Identifier les différences fondamentales avec Python.