Introduction au python

Master IMST "Système d'information et d'adchivage numérique" (IMST-SIAN)

Fabien Rico

Univ. Claude Bernard Lyon 1

4 octobre 2022

Plan

Introduction

Le python est un langage interprété très populaire.

- Date de 1991.
- Licence libre.
- Multiplatforme, il est présent sur la plupart des systèmes.
- Propose beaucoup de bibliothèques (`Pandas`, `numpy`, etc).
- Interprété, il est lent mais facile d'utilisation et ses bibliothèques sont efficaces.
- 2e langage le plus utilisé sur github.
- Réputé simple à apprendre.

Les types

Python est un langage fortement typé

- Les variables ont un type qui est déduit par l'interpreteur
- Les opérations qui n'ont pas de sens provoquent une erreur

```
>>> "hello"+"word"
'helloword'
>>> "hello"+"1"
'hello1'
>>> "hello"+1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

- On peut définir des types grâce aux classes (langage objet)

Type de base

Python est un langage fortement typé. Les type principaux sont :

- `bool` les booléens (vrai ou faux)
- `int` les entiers (de taille arbitraire)
- `float` les nombres a virgule flottante
- `str` les chaines de caractères
- `list` les listes ou tableau
- `tuple` comme les listes mais leur taille est fixe
- `dict` les dictionnaires ou tableaux associatifs

Il est possible de définir ses propres types. Par exemple la bibliothèque pandas propose les `DataFrame` équivalent de ceux du langage `R`

Les listes

```
# définition
li = []
# ajout àla fin
li.append(1)
li.append("mot")
print(li)
# -> [1, 'mot']
# concaténation
li2 = li+li
print(li2)
# -> [1, 'mot', 1, 'mot']
```

```
# indice des éléments
# premier
print(li2[0])
# -> 1
# dernier
print(li2[-1])
#-> mot
# plusieurs à la fois
print(li2[1:3])
# -> ['mot', 1]
```

Les dictionnaires

C'est une structure de données qui associe des clefs avec des valeurs

```
# définition
d1 = {}
# ajout d'un élément
d1["nom"] = "Lagaffe"
d1["prenom"] = "Gaston"
# déclaration d'un tableau rempli
d2 = {
  "profession": "Garçon de bureau",
  "age": 65
}
# affichage
print(d1)
```

```
# Fusion de tableau
d = {**d1, **d2}
# Liste des clefs
d.keys()
#List des valeurs
d.values()
```

```
# récupération d'une valeur
d1["nom"]
d1["clefquinexistepas"] #-> génère une erreur
# avec une valeur par défaut
d1.get("clefquinexistepas", 24)
```

Branchement conditionnel `if`

```
if condition:
  bloc d'instruction
else:
  autre bloc
```

Choix du bloc à exécuter selon une condition

Exemple

```
val = int(input("Donnez un nombre svp"))
if val>=0:
      print(f"La racine carée de {val} est ")
      print(math.sqrt(val))
else:
      print(f"Impossible de calculer la racine carée de {val}")
```

Les boucles `for`

```
for indice in sequence:
  bloc d'instruction
```

Répétition d'un bloc d'instruction dans lequel l'`indice` prend chacune des valeurs de la `séquence`

Exemple

```
sum = 0
for i in range(0,10):
  sum = sum+i
  print(f"i={i}")
print(f"total={sum}")
```

Exemple

```
d={
  'nom':'Lagaffe','prenom':'Gaston',
  'profession': 'Garçon de bureau',
  'age': 65}

for k,v in d.items():
  print(f"valeur de {k} : {v}")
```

Les boucle while

```
while condition:
  Bloc d'instructions
```

Le bloc d'instructions et répété jusqu'à ce que la condition soit fausse.

Exemple

```
val = int(input("Donnez un nombre positif svp "))
while val < 0:
  val = int(input("Donnez un nombre positif svp "))
print(f"La racine carée de {val} est ")
print(math.sqrt(val))
```

Affichage

Le python permet un affichage simple avec la fonction `print`

- Sans format `print("Message à afficher")`
- Avec format `print(f"Votre valeur est {val}")`
- Ou `print("Votre valeur est {}".format(val))`

Dans ces exemple, la variable `val` est automatiquement convertie en chaine de carractères pour être insérée dans le message affiché. Attention, la conversion peut dépendre du type de la variable.
On peut lire une chaine de carractère depuis le clavier avec `input`

```
text = input("Message d'invite àafficher : ")
```

Si ce qui est désiré est différent d'une chaine de carractère, il faut le convertir

```
valentiere = int(input("Message d'invite àafficher : "))
```

Entrée/sortie dans les fichiers

On peut lire ou sauvegarder des chose dans les fichiers :

```
f = open("demofile.txt", "r")
print(f.read())
```

Le second paramètre est le mode d'ouverture, ici en lecture (*read*). Là en écriture (*write*)

```
f = open("demofile.txt", "w")
f.write("Quelquechose")
# ne pas oublier de signaler la fermeture du fichier
f.close()
```

Pour sauvegarder des données complexes, il peut etre utile de les sérialiser.

Sérialisation

La sérialisation est la transformation de données en une description qu'on peut sauvegarder dans un fichier ou transmettre sur le réseau.
En python, les plus simples à utiliser sont :

- Le *json* : condensé et très portable, mais uniquement des types simples.
- *pickle* : spécifique à python, peut sérialiser un grand nombre de choses.