Python

Introduction: Installation-Les bases du langage

Mohamed Essaied Hamrita

IHEC, Université de Sousse-Tunisie

2025/2026

Table des matières

1. Introduction
2. Installation
   - Sous Windows
   - Mac OS X et Linux
   - Choix d'un éditeur
3. Bases de Python
   - Premières opérations
   - Les types de données
4. Structures de données
   - Listes
   - Tuples

Introduction

- Python est un langage de programmation de haut niveau : Il gère tout le côté matériel sans que le programmeur ait à s'en préoccuper. La seule tâche du programmeur est l'aspect purement algorithmique ;
- Python est langage semi-compilé : Il ne nécessite pas une compilation spécifique avant lancement ;
- Python est un langage orienté-objet : Il supporte l'héritage multiple et la surcharge des opérateurs ;
- Python est un langage libre et gratuit.

Utilisation

Python peut être utilisé de deux manières différentes :

- En mode interactif avec la console (shell) : dialoguer directement avec Python à partir du clavier en tapant une instruction et obtenant directement la réponse ;
- En mode fichier ou script : écrire l'ensemble des instruction (programme) dans un fichier (script) et le sauvegarder avec l'extension .py et l'exécuter plus tard.

Installation

Avant de pouvoir commencer la programmation, vous avez besoin d'installer un nouveau logiciel. Ce qui suit est une brève description de la façon de télécharger et d'installer Python.

Sous Windows : Python est téléchargeable depuis l'adresse suivante : `http://www.python.org`. Vous pouvez vous aider de ce tutoriel vidéo.
[`https://www.youtube.com/watch?v=km7KOVS-hBY`].
Une fois l'installation est terminée, vous pouvez démarrer Python en tapant sur `Démarrer (Start)` → `Python3.7` → `IDLE(Python 3.7)`.

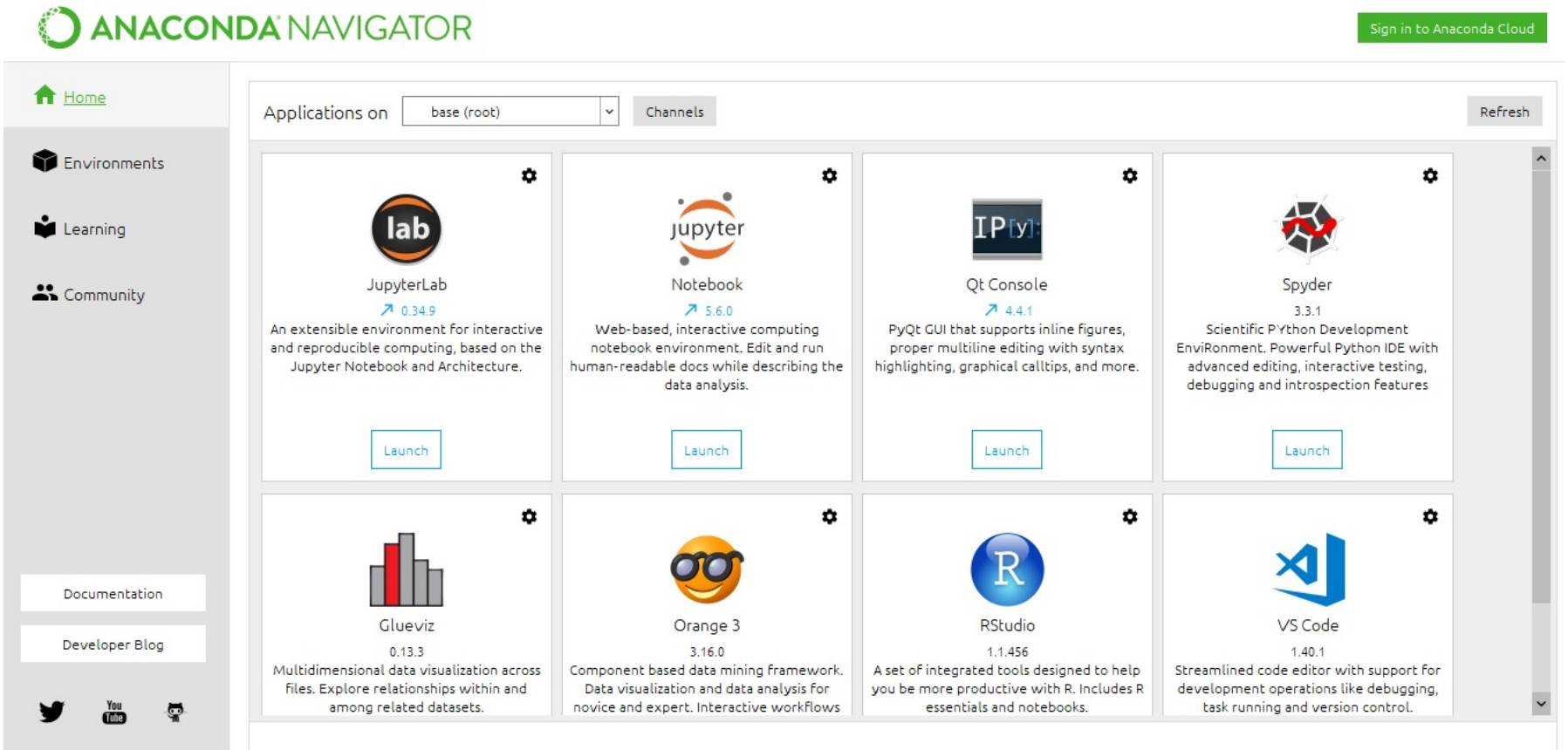
Mac OS X et Linux : Un interpréteur Python est déjà présent dans les distributions Linux et Mac OS X. L'installation de Python est parfois rendue nécessaire si on désire utiliser la dernière version du langage.

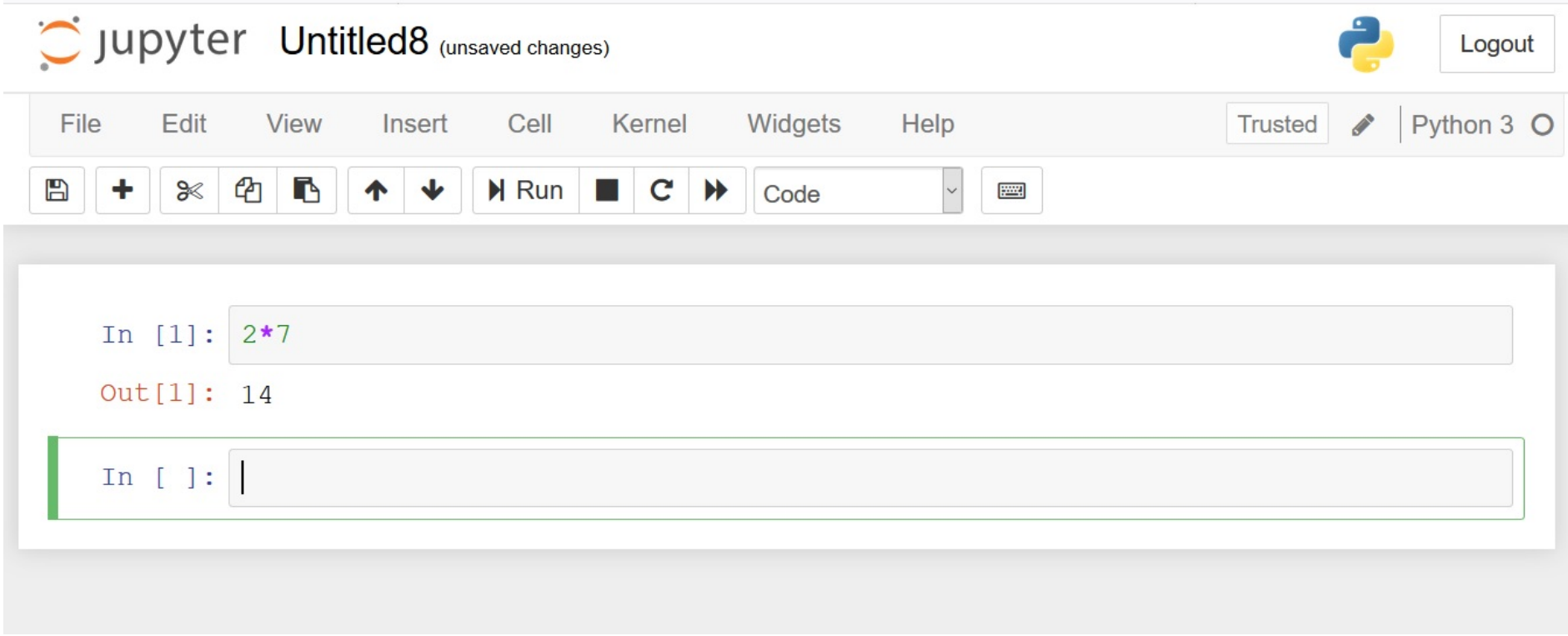
Choix d'un éditeur

Il existe plusieurs éditeurs de texte, payant ou gratuits. Dans ce cours on propose l'installation de la plateforme "anaconda". Cette plateforme est une distribution libre et open source des langages de programmation **Python** et **R** appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique. Il est téléchargeable depuis l'adresse suivante : https://www.anaconda.com.

Let's go

Une fois la distribution anaconda est installée, on peut utiliser comme éditeur, soit "jupyter" soit "spyder".




jupyter Untitled8 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]: 2*7
Out[1]: 14
In [ ]:

Remarques

- Un commentaire s'écrit en Python en commençant par le symbole ♯.

```
1 # Ceci est un commentaire
```

- Le langage Python est conçu pour le monde anglophone et l'utilisation des accents ne va pas de soi. Dans ce cas, il faut ajouter une ligne placée en première position qui précise que des accents pourront être utilisés.

```
1 # coding: utf-8
2 print("accentué") # accentué
```

- Pour nommer des variables, utiliser des suites de caractères non accentués, écrits avec des lettres minuscules et/ou majuscules, des chiffres et le caractère souligné "_". Ne pas commencer par un chiffre.

Remarques

- Python est sensible à la case ce qui signifie que les caractères majuscules et minuscules sont distingués. Par exemple, `Var`, `VAR`, `var` désignent 3 variables différentes.
- Éviter soigneusement les mots clés réservés qui ont un sens pour Python (`and`, `as`, `break`...).

```
1 and=1
2 SyntaxError: invalid syntax
3
4 4var=4
5 SyntaxError: invalid syntax
6
7 _var=4
8 _var+2
9  6
```

Premières opérations

```
1 a=2.0
```

a est une **variable**, en interne elle a été automatiquement typée en flottant "**float**" parce qu'il y a un point décimal. a est l'**identifiant** de la variable , = est l'**opérateur d'affectation**.
Lors de la déclaration d'une variable, bien que le programmeur ne spécifie pas le type de la variable, l'exécution de l'affectation attribue automatiquement et immédiatement le type qui lui semble le plus pertinent compte tenu de la valeur affectée à la variable. On dit que le typage des variables en Python est un **typage dynamique**.

```
1 b=1
2 type(b) # int
3 b=float(1) #Forcer le typage d'une variable
4 del b # supprimer la variable b
```

Types de données

Les principaux types de données utilisés par Python sont :

```
int # entiers, de longueur non bornée
float # flottants (réels)
complex # nombres complex
complex(1) # 1+0j
complex(1,-1) # 1-1j
bool # booléen (True/False)
str # chaînes des caractères
```

Une constante chaîne de caractère doit être délimitée par des guillemets (ou des quotes)

```
print('Mohamed') # Mohamed
print("Mohamed")  # idem
print('''Mohamed''')  # idem
```

Conversion

Conversion en numérique

```
1 a="12"  # a est une chaîne de caractère
2 b=float(a) # b est de type float
```

Conversion en logique

```
1 a=bool("TRUE")  # a est booléen et contient la valeur
    True
2 b=bool(0) # b est booléen et contient False
```

Conversion en chaînes des caractères

```
1 a=str(120) # a est de type chaîne et contient 120
```

Opérations usuels

```
1+1; 1*2   # 2 (addition); 2 (multiplication)
1-2; 1/2 # -1 (soustraction); 0.5 (division)
1//2; 1%2 # 0 (division euclidienne); 1 (reste de la
   division)
abs(x) # valeur absolue
```

Affichage des nombres dans un texte :

```
print("J'ai %d stylos" % 2)  # J'ai 2 stylos
print("J'ai %3d stylos" % 2) # J'ai   2 stylos
print("J'ai %03d stylos" % 2) # J'ai 002 stylos
print("J'ai %f stylos" % 2) # J'ai 2.000000 stylos
print("J'ai %.2f stylos" % 2) # J'ai 2.00 stylos
```

Affectation avec opérations

```
a+=2  # a=a+2
a-=5  # a=a-5
a*=3  # a=a*3
a**=2 # a=a**2 (puissance)
a/=2  # a=a/2
a//=3 # a=a//3
```

Opérateurs de comparaison :

```
<, <=, >, >=, !=  , ==
a = (12 == 13) # a est de type bool, il a la valeur
    False
```

Saisie et affichage

`input()` permet d'effectuer une saisie au clavier. Il est possible d'afficher un message d'invite. La fonction renvoie toujours une chaîne, il faut convertir.

```
1 a=input("Saisir votre valeur")
2 a=float(a)
```

L'affichage se fait à l'aide de la fonction `print()`.

```
1 print(a)   # affichage explicite
2 b=2.0
3 print((a, b)) # affichage mutiple
```

Opérations sur les chaînes

Pour les chaînes de caractères, deux opérations sont possibles, l'addition et la multiplication. L'opérateur d'addition "+" permet de concaténer (assembler) deux chaînes de caractères et l'opérateur de multiplication "*" permet de dupliquer plusieurs fois une chaîne.

```
1 ch="Bonjour"
2 ch + " tout le monde" # Bonjour tout le monde
3 ch*2  # BonjourBonjour
4 2*(ch + " tout ") # Bonjour tout Bonjour tout
```

Afin d'accéder au $i^{\text{ième}}$ caractère de la chaîne `ch`, on peut saisir `ch[i]` avec `0 <= i <= longueur(ch)-1` car les indices commencent à 0.

```
1 nom="Mohamed"
2 nom[0] # M
3 nom[1:3] # oh
4 nom[7] # IndexError: string index out of range
```

Opérations sur les chaînes

On peut utiliser aussi les indices négatifs compris entre -longueur(ch) et -1 pour accéder aux caractères de la chaîne ch. L'accès se fait de droite à gauche.

```
nom="Mohamed"
nom[-1] # d
nom[-3:-1] # me
nom[-7] # M
```

La fonction `len(ch)` permet de renvoyer le nombre de caractères dans la chaîne `ch` sous la forme d'un entier.

```
nom="Mohamed"
len(nom)   # 7
```

Minuscules et majuscules

```
print(nom.lower())      # mohamed
print(nom.upper())      # MOHAMED
print(nom.swapcase())   # mOHAMED
```

Une structure de données est un moyen de stocker des données et d'avoir des méthodes particulières pour les récupérer ou les manipuler. Python a quelques structures de données intégrées.

Listes :

```
1  a=[1, 2, 0.75]     # création d'une liste
2   a.append(1.35)      # ajouter un élément à la fin de la
       liste
3  a.insert(0,0.75)  # ajouter un élément au début de la
     liste
4  a.count(0.75)  # compter combien de fois on a 0.75
5  a.remove(1) # supprimer la valeur 1 de la liste
6  b = [45, 56, 90]
7  a.append(b) # [1, 2, 0.75, [45, 56, 90]]
8  a.remove(b)
9  a.extend(b) # [1, 2, 0.75, 45, 56, 90]
10 a.sort() # ordre croissant
11 a.reverse()  # renverser les éléments de la liste
```

Structures des données

Tuples :

Les tuples sont des séquences qui contiennent des éléments de types hétérogènes. Les tuples composés d'un seul élément ont une écriture un peu particulière.

```
1 a=(123)    # a est un entier
2 type(a)    # <class 'int'>
3 b=(123,)   # b est un tuple
4 type(b)    # <class 'tuple'>
```

La commande range()

La commande `range()` permet de créer des objets de type 'range' listant une suite arithmétique d'entiers.

```
1 a=range(1,5,2)     # renvoie les entiers de 1 à 4 avec
    un pas 2
2 list(a)  # afficher les éléments de a
3 range(1,5) # avec un pas=1
4 range(5)   # séquence de 0 à 4 avec un pas=1
```