TD 7 — Programmation Python : outils de travail

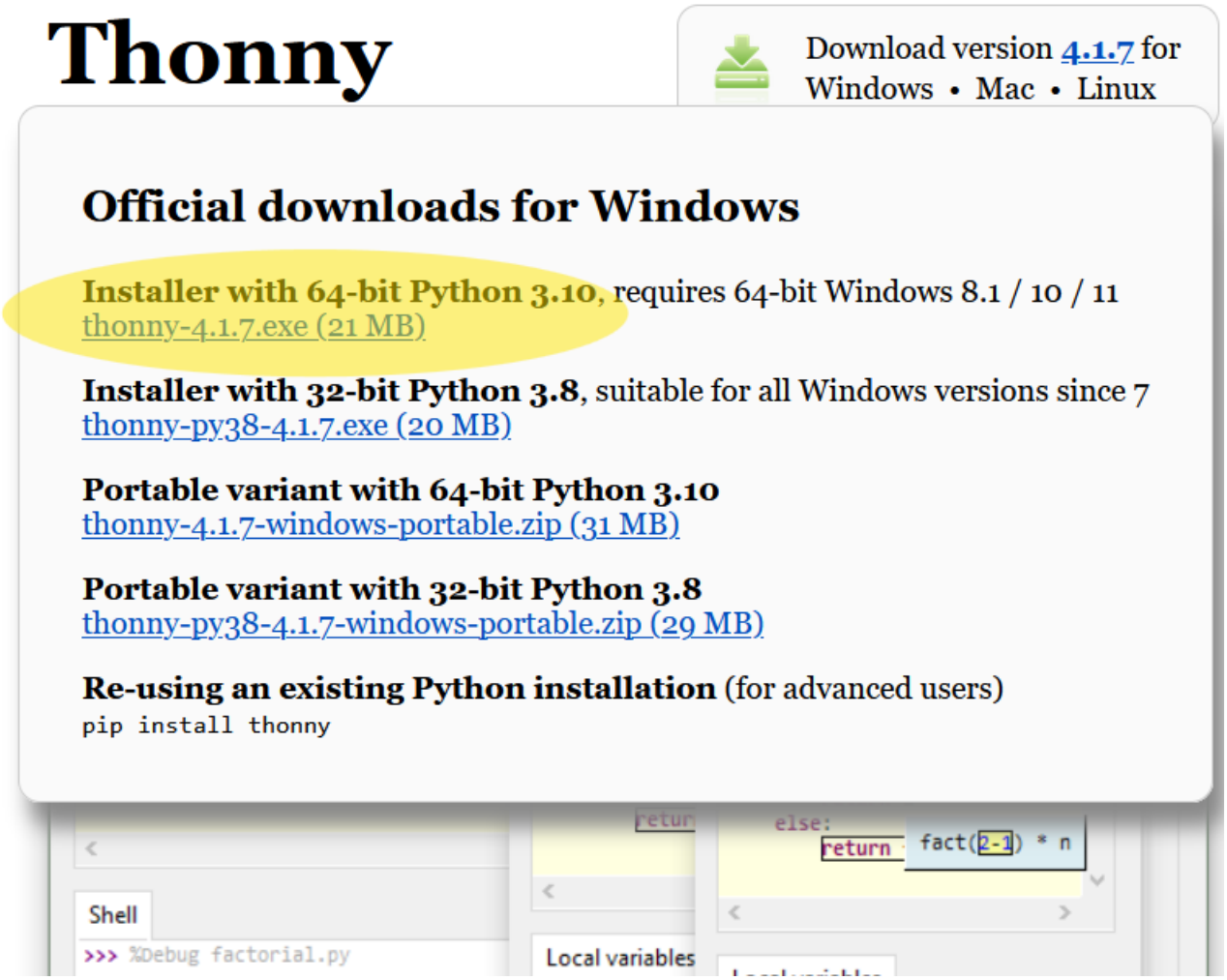
Dans ce TD, nous allons commencer la programmation en Python. C'est un langage de plus en plus utilisé. Contrairement au langage C, c'est un langage de haut haut niveau : il n'offre pas d'accès direct à la mémoire. En contrepartie, les codes Python sont souvent beaucoup plus faciles à écrire et compacts.

Nous allons commencer par installer les outils de travail. Nous allons utiliser l'environnement de développement Thonny. C'est environnement très adapté pour débuter en programmation. *L'utilisation de tout autre outil de programmation pour cette ressource est formellement proscrite.* En particulier, les examens se feront sous Thonny.

Dans la deuxième partie de ce TD, nous écrivons et analysons les premiers codes en Python.

1 Installation de Thonny

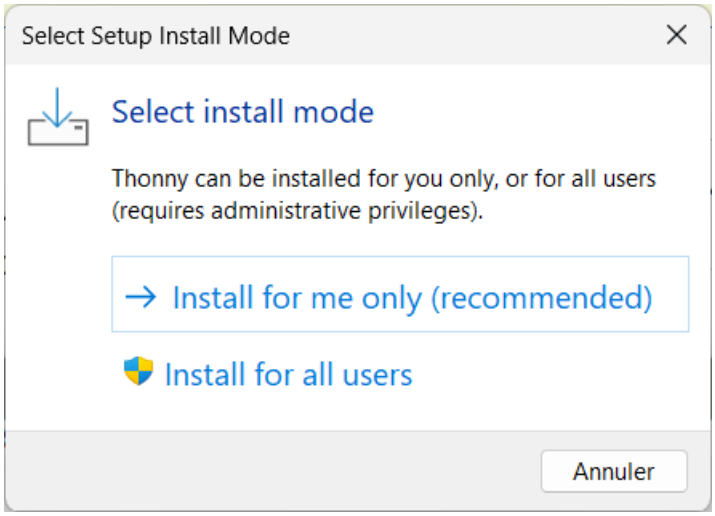
Rendez-vous sur la page https://thonny.org/ puis téléchargez la version `Installer with 64-bit Python 3.10`. Cet installeur contient l'environnement de développement et les outils nécessaires.



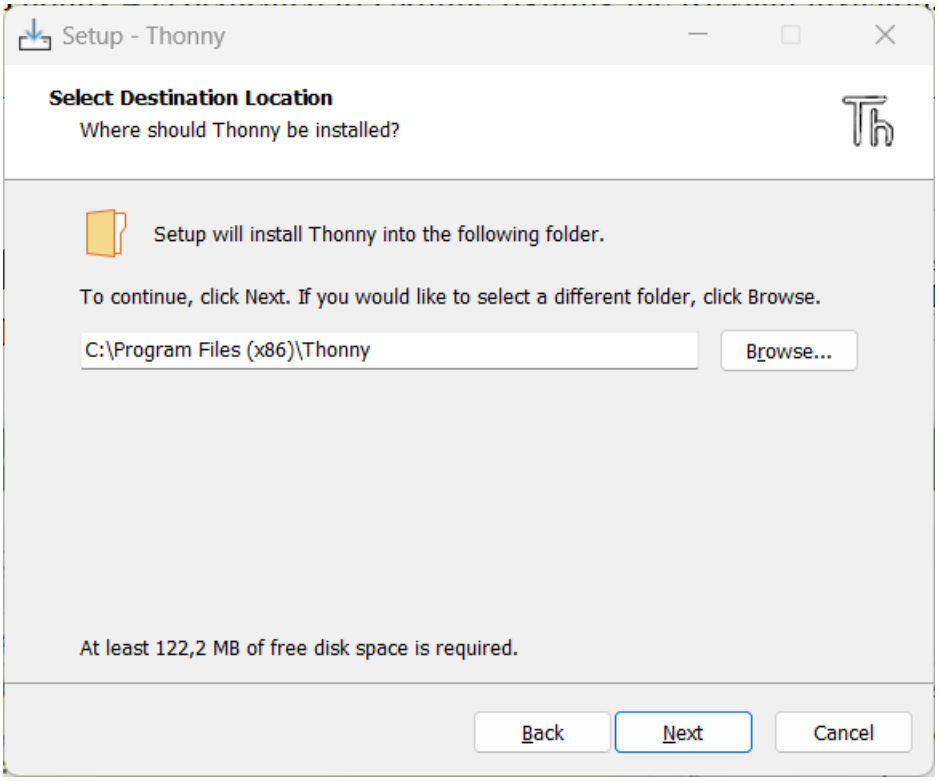
Consignes d'installation

Veillez à suivre scrupuleusement les instructions d'installation. En cas de mauvaise configuration, vous ne pourriez pas passer les éventuelles évaluations sur machine.

Lorsque le téléchargement est terminé, exécutez l'installeur, et choisissez `Install for all users`.



Suivez les instructions d'installation : Next, puis acceptez la licence. Dans la boite de dialogue `Select Destination Location`, vérifiez que le chemin d'accès est `C:‣Program Files (x86)‣Thonny`. Les autres étapes ne nécessitent pas de configuration : cliquez sur Next autant de fois que nécessaire pour terminer l'installation.



2 Variables et affichage

2.1 La fonction `print`

En Python, le code principal du programme, contrairement au langage C, n'a pas besoin d'être placé dans une fonction `main`. On écrit les instructions directement au début du code source. Les codes Python ont l'extension `.py`.

On vous fournit ci-dessous quelques codes à tester et à comprendre.

Python
```python
print("Hello world !")
```

Remarquez que, contraitement au C, les instructions ne se terminent pas par un point-virgule, mais par un saut de ligne.

L'affichage d'une variable se fait comme suit :

Python
```python
a = 7 # Création et initialisation d'une variable.
print(a)
print("a =", a)
```

La fonction `print` offre une syntaxe bien plus agréable, nommée *f-string*, pour afficher des variables :

Python
```python
a = 3
b = 5.4
print(f"La somme de {a} et {b} vaut {a+b}.")
```

Les types de données en Python

En langage C, les variables ont un type, déterminé au moment de la compilation et déclaré explicitement dans le code source. En langage Python, le système de types est très différent : ce sont les *valeurs* qui ont un type, pas les variables. Par exemple, l'instruction `int a = 7` est incorrecte, alors que l'instuction `b = 7` est correcte.

La valeur `7` est de type `int`, mais la variable `b` n'a pas de type définitif. Par conséquent, le code suivant est correct :

Pyhon
```python
b = 7
print(b)        # Affiche 4
print(type(b)) # Affiche <class 'int'>

b = -7.5
print(b)        # Affiche -7.5
print(type(b)) # Affiche <class 'float'>
```

2.2 Formatage des entiers et des flottants

Analysez les codes suivants :

```python
a = 14
print(f"a = {a}")      # Pas d'instruction de formatage.
print(f"a = {a:4d}")   # Au moins 4 caractères, si besoin des blancs.
print(f"a = {a:04d}") # Au moins 4 caractères si besoin des 0.
```

```python
b = 3.14116
print(f"b = {b}")        # Pas d'instruction de formatage.
print(f"b = {b:.3f}")    # 3 chiffres après la virgule
print(f"b = {b:6.2f}")   # Au moins 6 caractères, dont deux chiffres après la virgule,
                         # si besoin des blancs.
print(f"b = {b:06.2f}") # Au moins 6 caractères, dont deux chiffres après la virgule,
                         # si besoin des zéros.
```

2.3 La fonction `input`

Pour récupérer une saisie clavier, on utilise la fonction `input` :

```python
saisie = input("Saisir une valeur : ")
print(saisie) # Affiche la variable saisie
```

La fonction print renvoie une valeur de type `str` (*string*, chaine de caractères). Si l'on souhait modifier le type de la valeur retournée (par exemple, si c'est un nombre entier), on procède à un *cast* (changement de type) :

```python
a = input("Saisir un entier : ")
print(type(a)) # Affiche <class 'str'>
int(a)         # Nécessite que la valeur saisie soit un nombre !
print(type(a)) # Affiche <class 'int'>
```

Exercice 1. Demander la saisie d'un caractère au clavier, puis afficher à l'écran la lettre située 10 lettres plus loin dans l'alphabet. S'il reste moins de 10 lettres avant la fin, recommencer partir de la lettre `'a'`. On ne gèrera que les lettres minuscules.