GASSENDI

vendredi 4 décembre 2020

Le Club Informatique Gassendi

TP monde connecté : cours du 26/11/2020 : configuration Pi, VNC, WinSCP

Élaboration	
4 décembre 2020	
Jean D	GASSENDI
Animateur	
Administration informatique	
Nom du fichier	00_TP_monde_connecte_cours_26_11_2020_Python_IDE-Thonny_V0.1.odt

Python

Généralités

Utilisation du langage Python pour réaliser des programmes de traitement des données.

Python est un langage **de haut-niveau**, **interprété, interactif** et **orienté objet**

intérêt du langage **haut niveau**:

code facile à lire, syntaxe peu complexe donc: beaucoup plus facile à écrire qu'en assembleur (gain de temps)

plus facile de modifier le programme et de le corriger

il est **portable** c'est-à-dire qu'il pourra fonctionner sur des machines différentes sans trop de modifications.

intérêt d'un langage **interprété**:

les instructions sont exécutées directement grâce à un interpréteur. Pas besoin de compilation avant l'exécution du programme.

intérêt d'un langage **interactif**:

le développeur peut écrire sont programme directement sous le prompt Python « **>>>** ».

intérêt du langage **orienté objet** :

Permet d'encapsuler le code des programmes sous forme d'objet réutilisable dans d'autres programmes.

Python

Structure générale d'un programme Python

Utilisation du format universel utf-8

Zone d'insertion des bibliothèques utilisées

Zone d'insertion des constantes et variables utilisées avec typage implicite

Zone d'insertion des fonctions utilisées

Zone programme principal

tabulation de 4

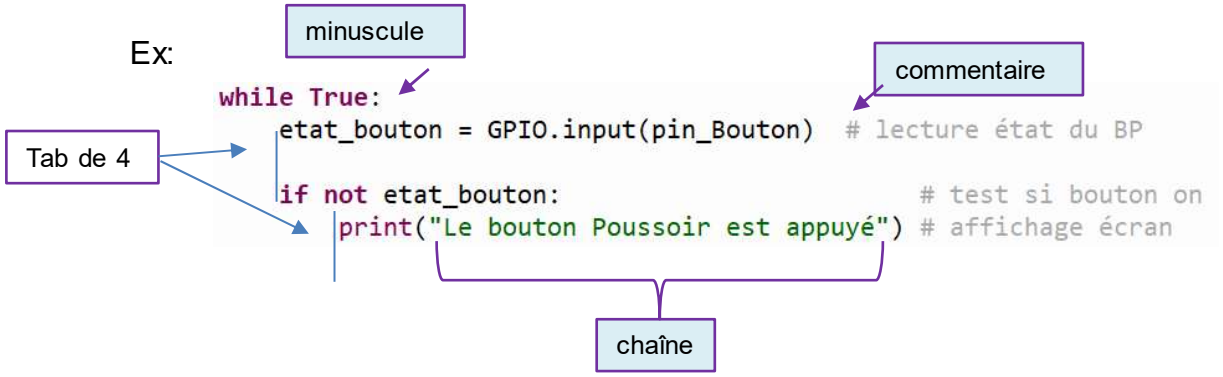
```
# -*- coding: utf-8 -*-
import pi
from time import time.sleep

passe, dehors = 0, 3                                          # type int
tempo = 0.2                                                   # type float
msg = ‘’Chiffre en dehors de la plage. Recommencez!”          # string
touche = None                                                 # Boolean
code = [ ]                                                    # liste

def essai(a):
….clavier = [[1, 2, 3], [4, 5, 6], [7, 8, 9], ["*", 0, "#"]]
.... ligne, colonne = 0+a, 2-a
.... return clavier[ligne][colonne]

def val_code(x):
….code_ref = [5, 3, 3, 5]
….suite de la fonction

If __name__ == ‘__main__’:                                    # prg principal
....while  passe < 4 :
.....  chi = int(input(”entrez n° ligne”)
.....while  chi >dehors :
…….....print(  msg)
…….. chi = int(input(”entrez n° ligne”)
.....  while touche == None:
…….. code.append(essai(chi))
…….. time.sleep(tempo)
…….. touche  = not None
…….. print("chiffre",  code)
……..passe = passe + 1
……..touche = None
….val_code(code)
```

Python

Contraintes et règles de rédaction Python

. Python est sensible à l'indentation (utilisation de la touche: ''**TAB**'' ⇄)

Ex:



. Les instructions doivent respecter la syntaxe ➔ sinon message d'erreur lors de l'exécution

pas de caractères accentués,
utilisation de minuscule,
les commentaires sont précédés de #
une chaîne de caractère est délimitée par:
'chaîne'
''chaîne'' ou
''' La phrase est écrite sur plusieurs lignes'''

. Mots réservés à ne pas utiliser pour les variables ou fonctions

and, assert, break, class, continue, def, del, elif, else, except, exec, finally, from, for, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

Python

Différents types d'erreurs

Erreurs de syntaxe: *(facile à corriger)*
un message d'erreur survient lors du « run » programme.

Erreurs de logique: *(plus difficile voir organigramme)*
pas de message d'erreur mais le programme n'a pas le résultat escompté

Erreurs à l'exécution: *(ardu introduire dans le programme le traitement d'erreur)*
plantage du programme (prévoir des exceptions sinon point de salut)

la fenêtre « run » indique le type de plantage et l'exception à instancier.

Exemples courant de plantage :
div/ 0,
test avec condition inconnue,
lire/écrire dans fichier mal répertorié.

Python

Grâce à sa portabilité, Python fonctionne également sous Windows.
mais uniquement en mode graphique avec l'IDE Thonny.

Pas de mode console comme sur la Raspberry (cf. console)

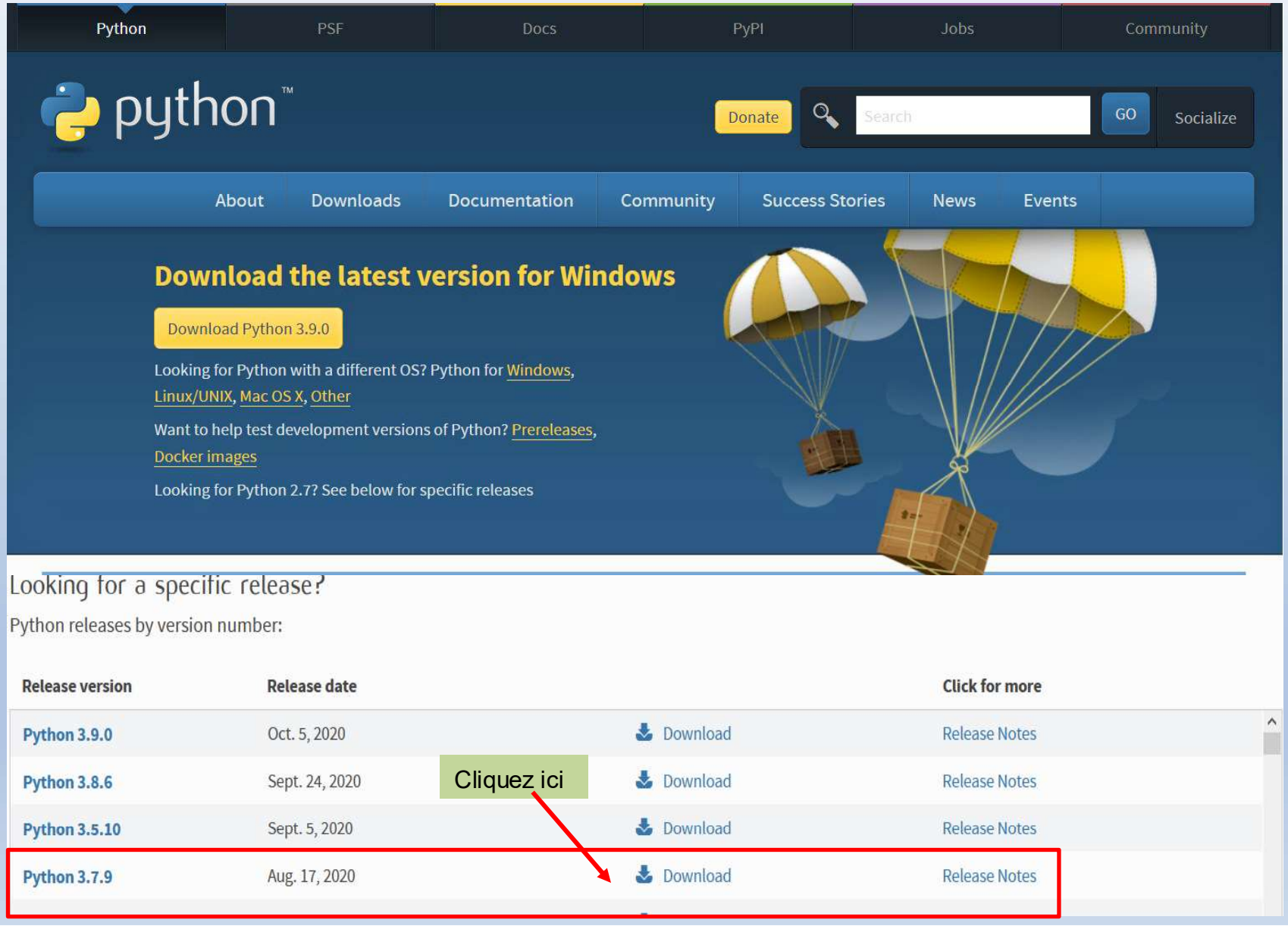
Télécharger Python sur le PC Windows

A: aller sur le site:

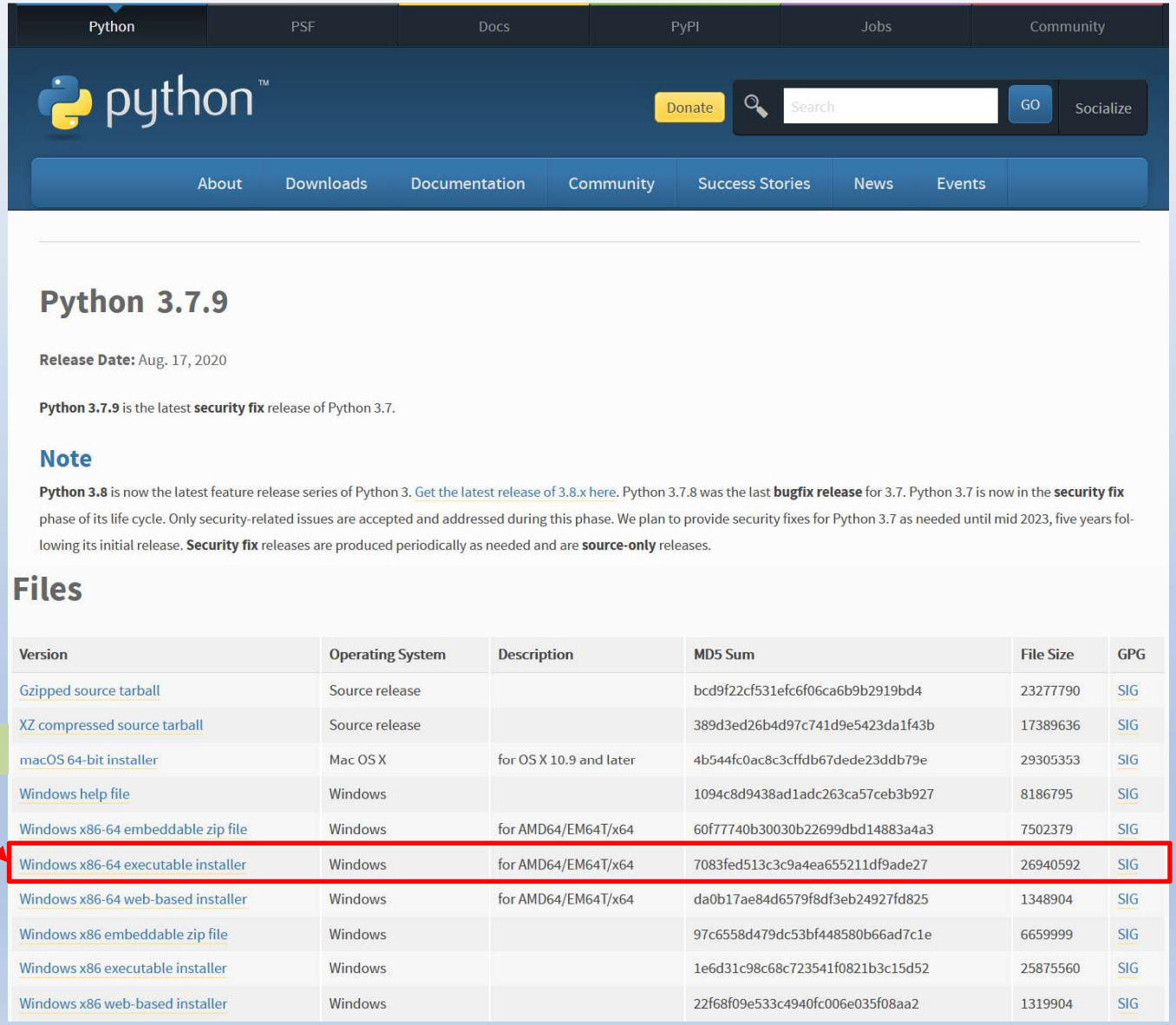
https://www.python.org/downloads/release/python-379/

Python

B: Choisir la version désirée:



Python



Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		bcd9f22cf531efc6f06ca6b9b2919bd4	23277790	SIG
XZ compressed source tarball	Source release		389d3ed26b4d97c741d9e5423da1f43b	17389636	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	4b544fc0ac8c3cffdb67dede23ddb79e	29305353	SIG
Windows help file	Windows		1094c8d9438ad1adc263ca57ceb3b927	8186795	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	60f77740b30030b22699dbd14883a4a3	7502379	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	7083fed513c3c9a4ea655211df9ade27	26940592	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	da0b17ae84d6579f8df3eb24927fd825	1348904	SIG
Windows x86 embeddable zip file	Windows		97c6558d479dc53bf448580b66ad7c1e	6659999	SIG
Windows x86 executable installer	Windows		1e6d31c98c68c723541f0821b3c15d52	25875560	SIG
Windows x86 web-based installer	Windows		22f68f09e533c4940fc006e035f08aa2	1319904	SIG

Cliquez ici

Python

Ci-après des exemples qui illustrent l'utilisation de Python avec la Raspberry

en mode inter-actif.

mode console

Python

```
pi@raspberrypi~ $: python3        (lancement de l'interpréteur)
                infos sur la version Python 3.1.1+
>>>  5+2                                          # instruction 1
7     # résultat 1

>>> print (' Bonjour ')                          # instruction 2
Bonjour                                         # résultat 2

>>>  phrase=''passage de minuscule en majuscule''   # instruction 3
>>> print (phrase.upper)                         # utilisation de la méthode upper 4
PASSAGE DE MINUSCULE EN MAJUSCULE               # résultat 3,4

>>>from math import pi                          # appel de la fct pi dans la biblio math 5
>>>rayon=2                                      # instruction 6
>>> print (''La surface du cercle = '', pi * rayon**2)   # instruction 7
La surface du cercle = 12.566                   # résultat 5,6,7
```

Exemple code 'Objet'

Python est un logiciel adapté aux débutants et qui comporte de nombreuses applications

Nota: Toute la documentation sur Python est accessible en ligne.

https://www.python.org/doc/

Le **mode interactif** est limité à une suite d'instructions mais n'est pas adapté pour des programmes plus complexes et de taille importante.

Passage en **mode script** avec l'IDE

Python

Le **mode script** consiste à écrire un programme en utilisant un IDE. (**Thonny**)
(environnement de développement qui contient Editeur, Debugger, Exécutable)

Pour cela: Nous allons créer un fichier qui s'appellera: **tp1_bp.py**

py pour indiquer Python'

sous **Windows**

créer un répertoire '*TP_connect*'.

ouvrir l'IDE **Thonny** et enregistrer le fichier sous: **tp1_bp.py**

sous **Raspbian**

Se mettre en mode console

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ mkdir TP_connect
```

Création répertoire TP_connect'

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ cd TP_connect
```

Aller au répertoire TP_connect'

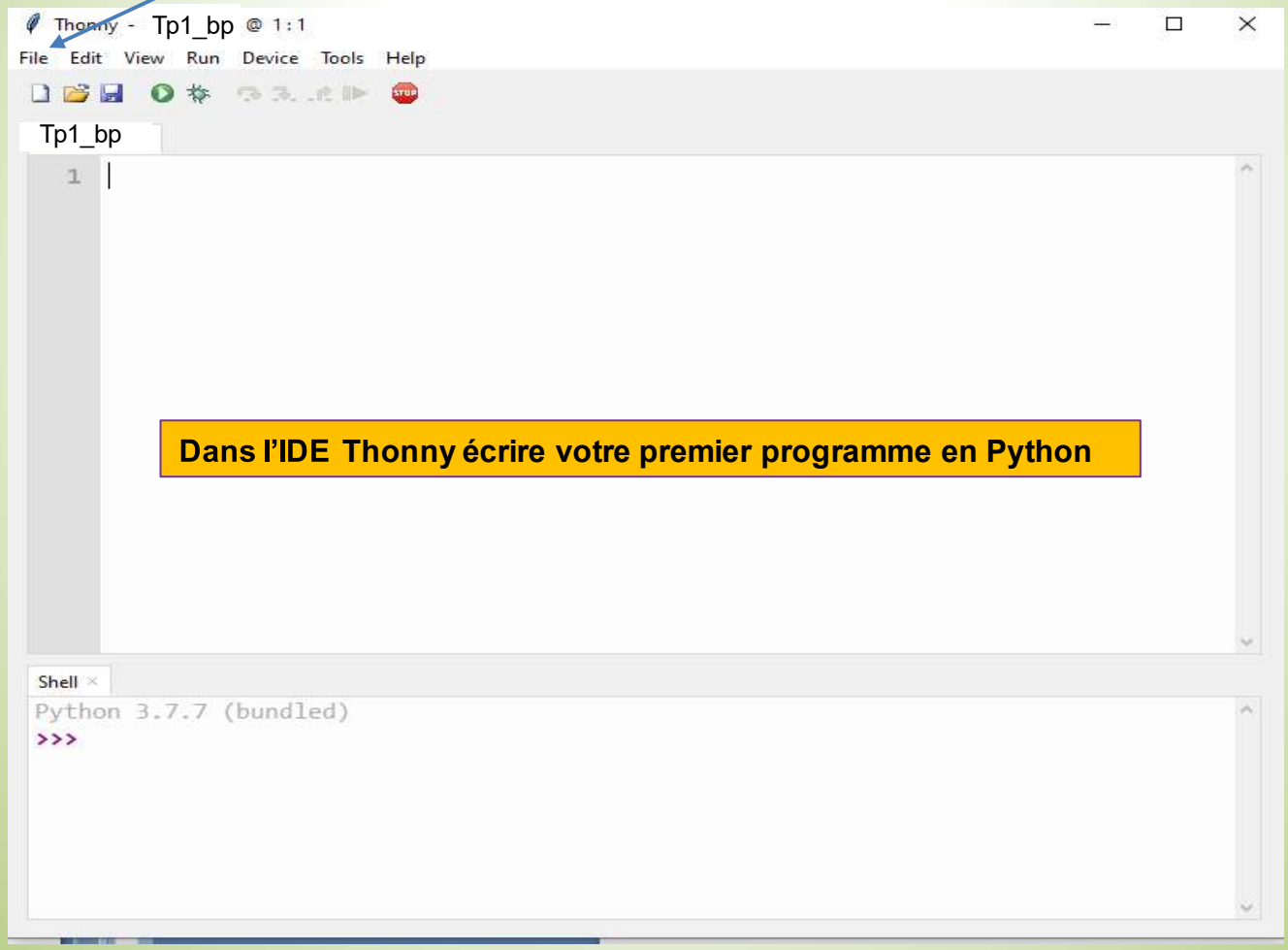
```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ touch tp1_bp.py
```

Créer un fichier vide tp1_bp.py

Python

Se mettre en mode graphique en ouvrant l'IDE Thonny

Dans « File » faire « Open » le fichier **Tp1_bp.py**



Glossaire

Sigle	
IDE	**I**ntegrated **D**evelopment **E**nvironment

IDE Thonny

1. But

Un **IDE** est un **E**nvironnement de **D**éveloppement **I**ntégré qui propose un ensemble d'outils spécifiques dédiés aux travaux de la programmation.

L'IDE **« Thonny »** a été conçu pour les programmeurs débutants qui désirent apprendre le langage **Python** Cet outil intégre son propre interpréteur Python 3.6

2. Télécharger Thonny

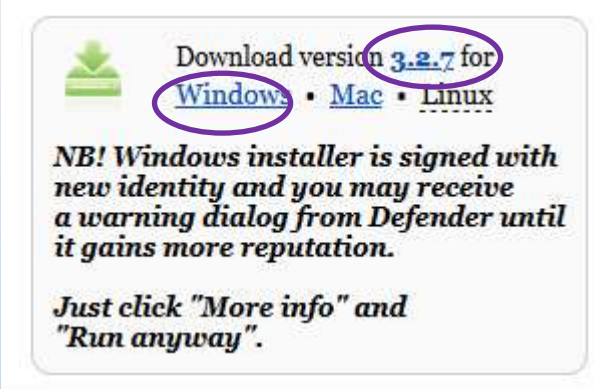
A: aller sur le site:

https://thonny.org

https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe

IDE Thonny

B: Choisir la version désirée:

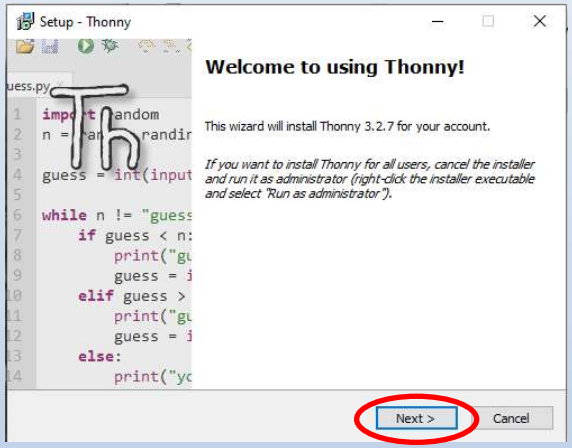


C: Télécharger la version choisie:

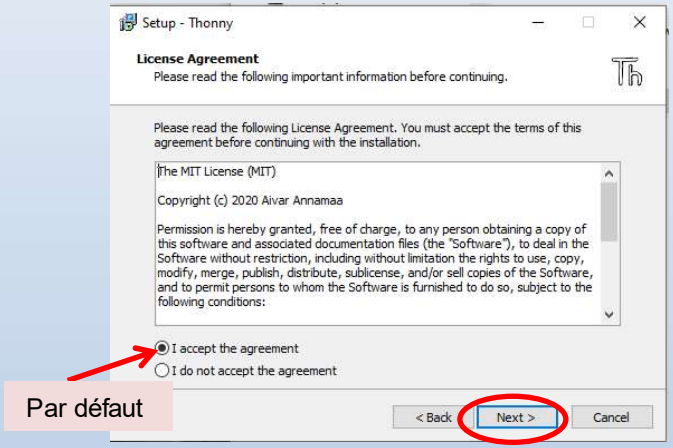
IDE Thonny

D: Installation de l'IDE sous Windows

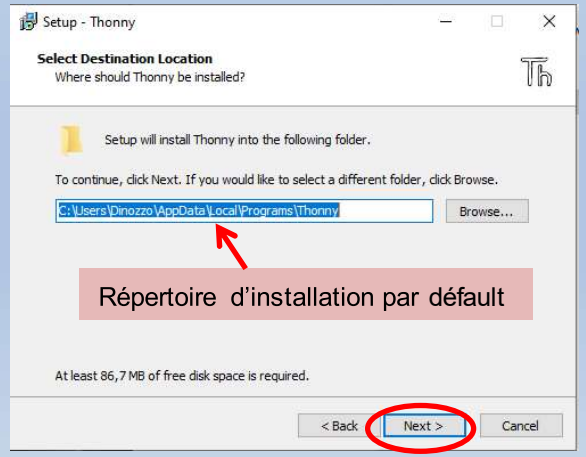
Etape 1



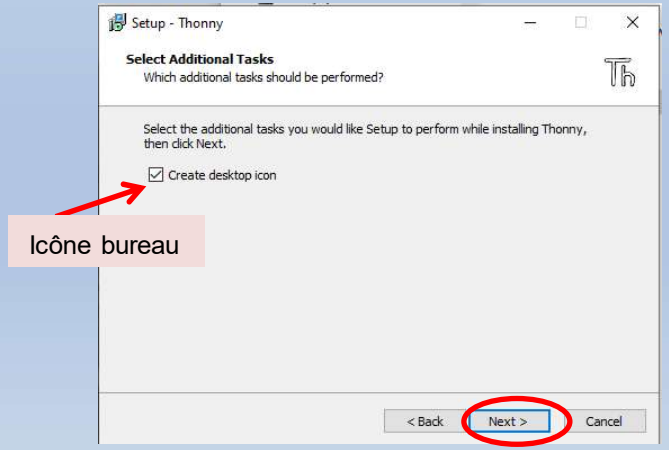
Etape 2



Etape 3

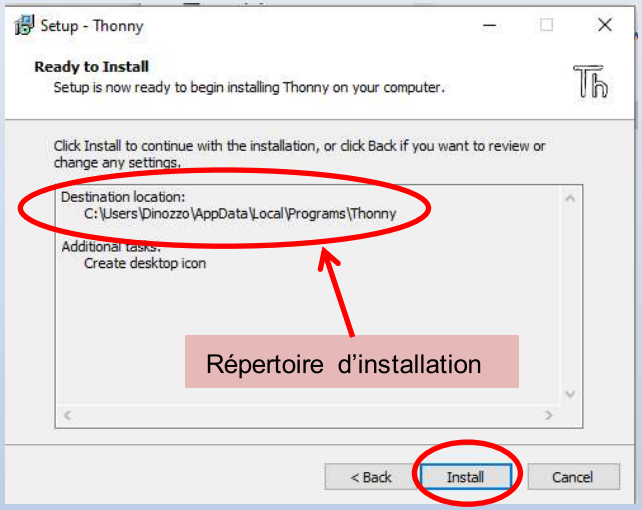


Etape 4

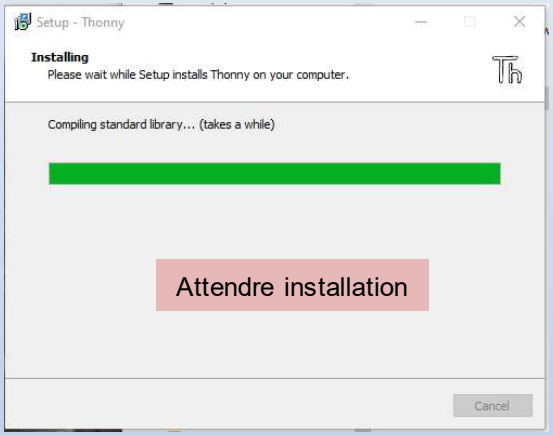


IDE Thonny

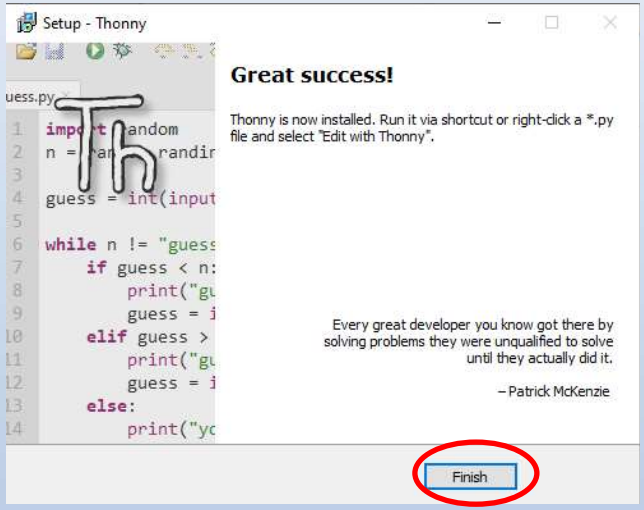
Etape 5



Etape 6



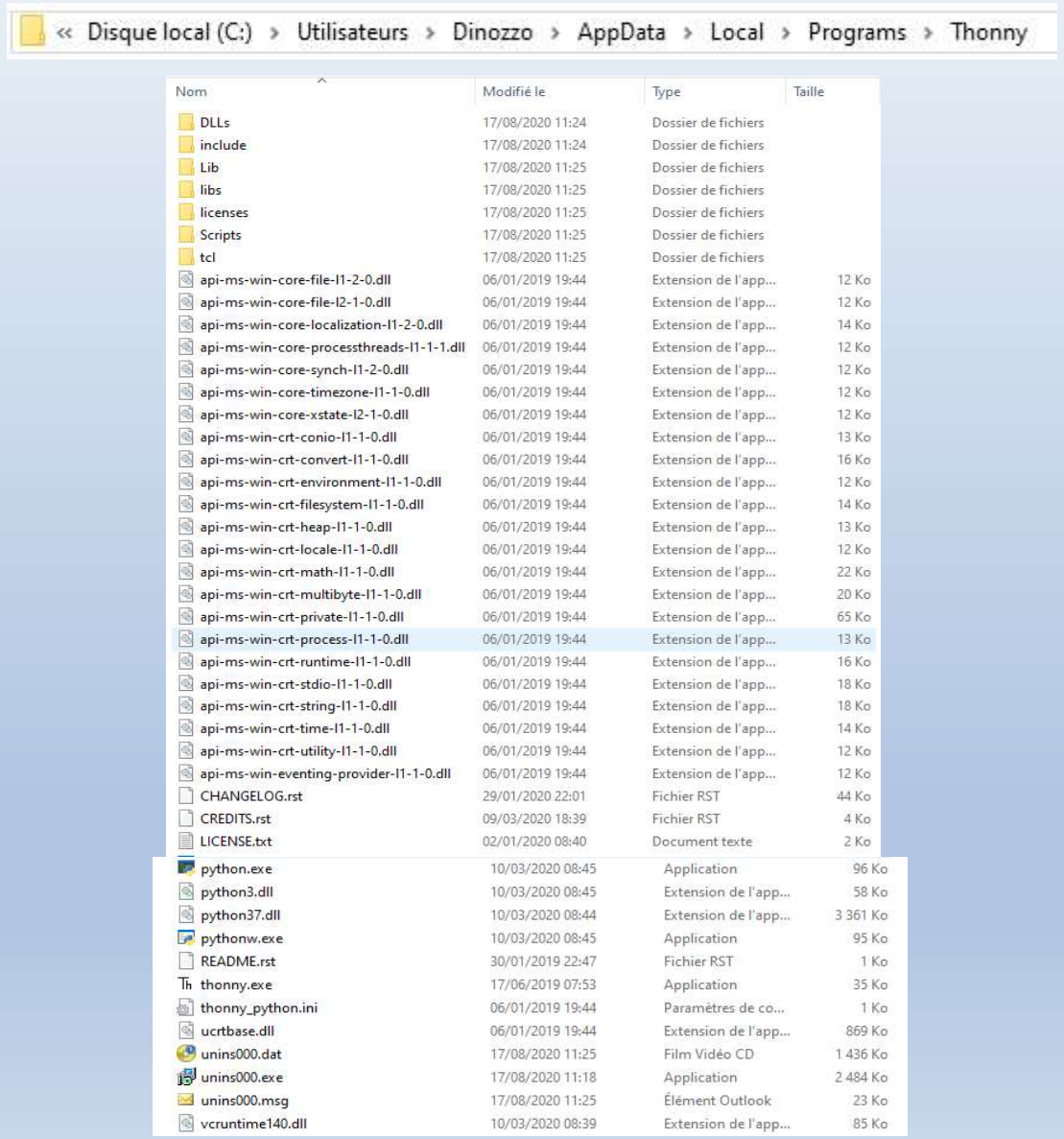
Etape 7



3. Ouvrir l'IDE Thonny

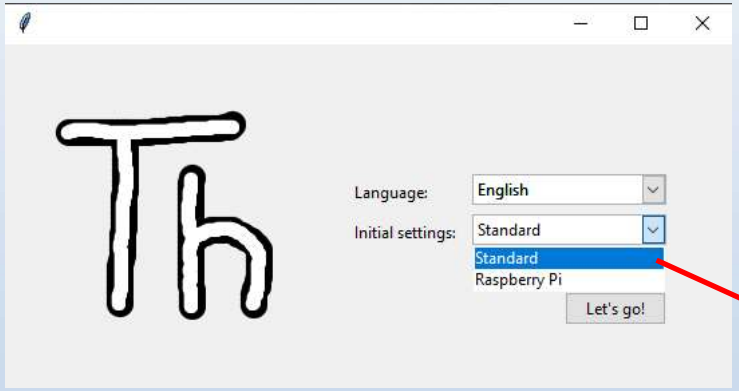
IDE Thonny

E: Détails du répertoire Thonny sous Windows

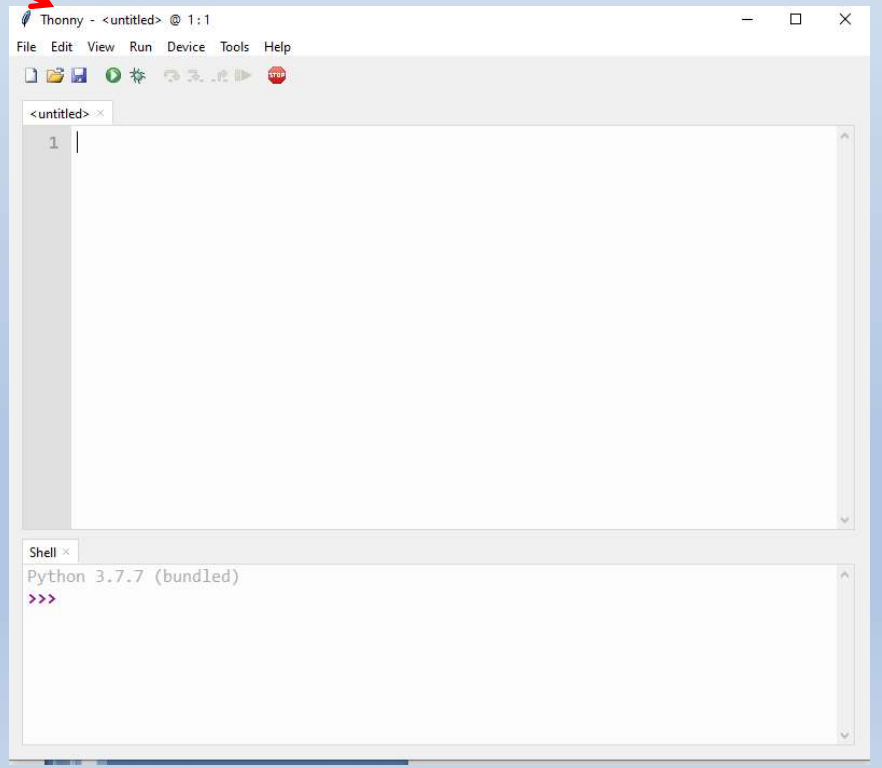


IDE Thonny

4. Paramétrage à la 1ére ouverture de Thonny



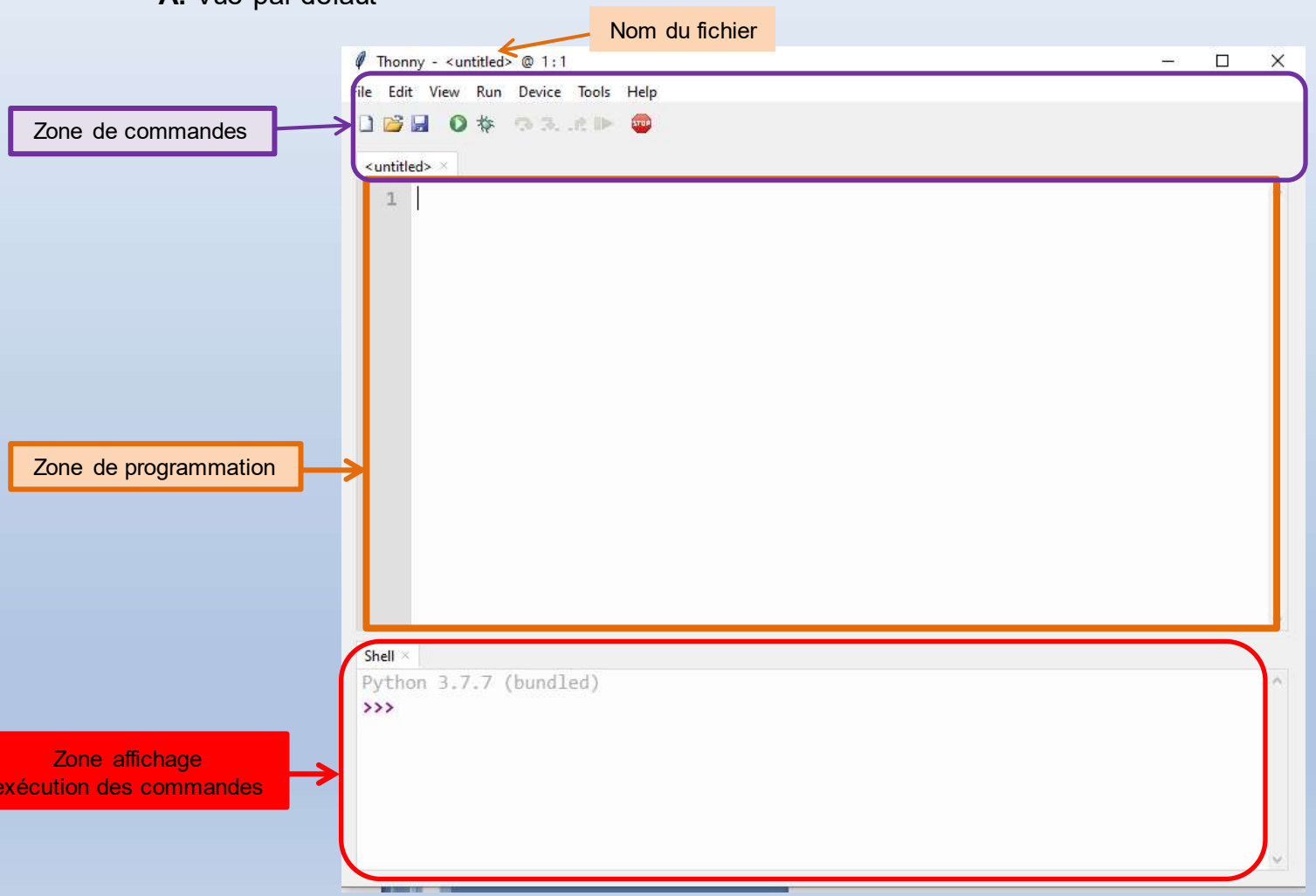
Choix Windows



IDE Thonny

5. Fenêtre de travail Thonny

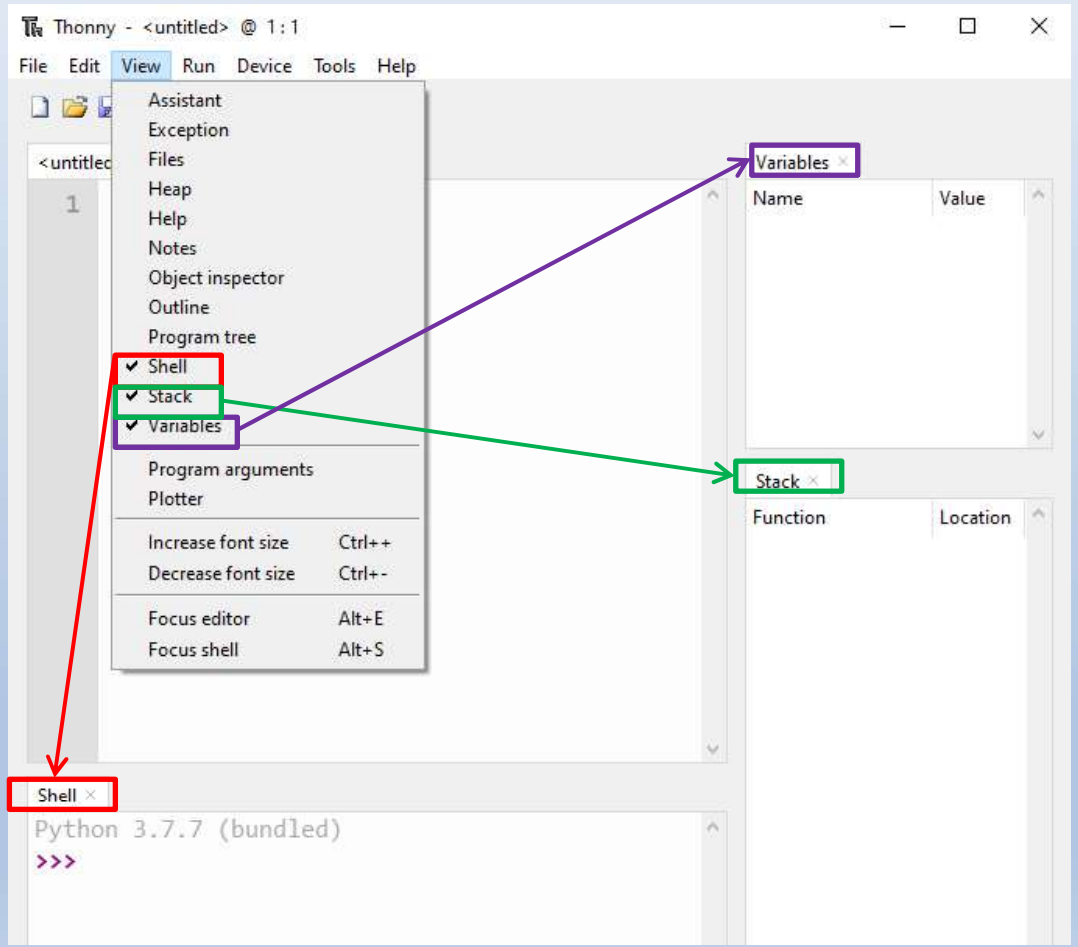
A: Vue par défaut



IDE Thonny

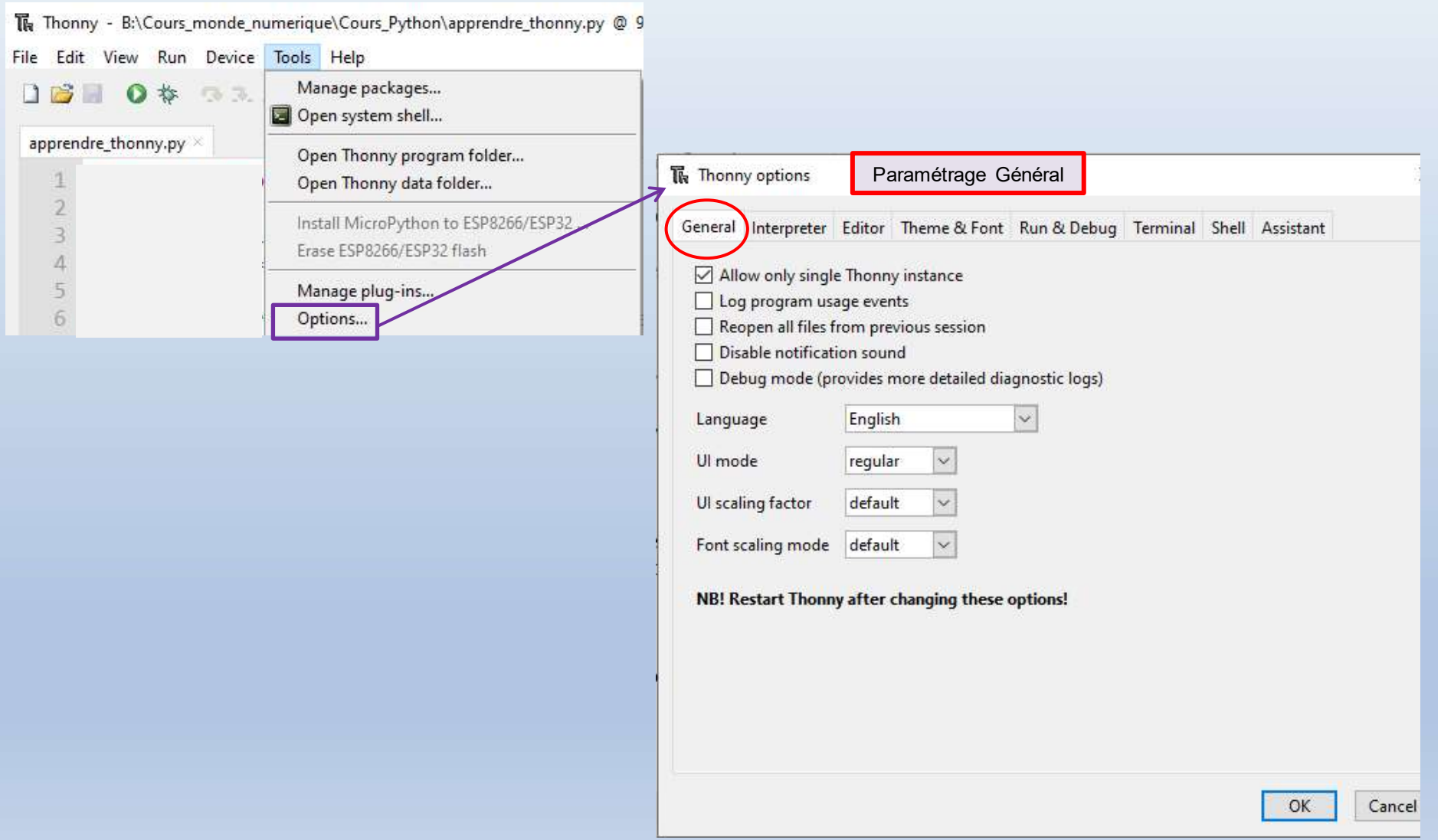
B: Vue personnalisée

Pour personnaliser la vue de travail, il faut cocher les items désirés

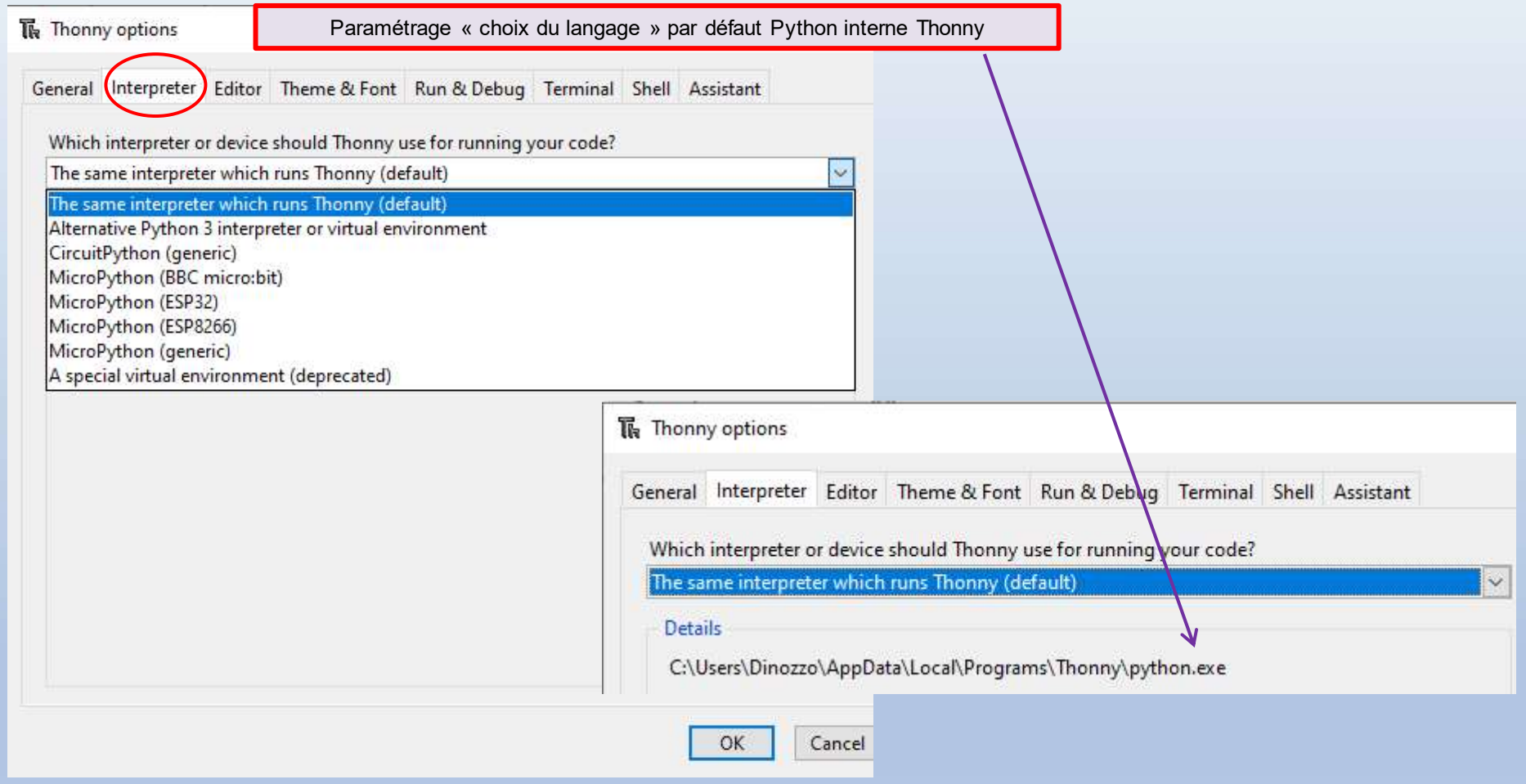


IDE Thonny

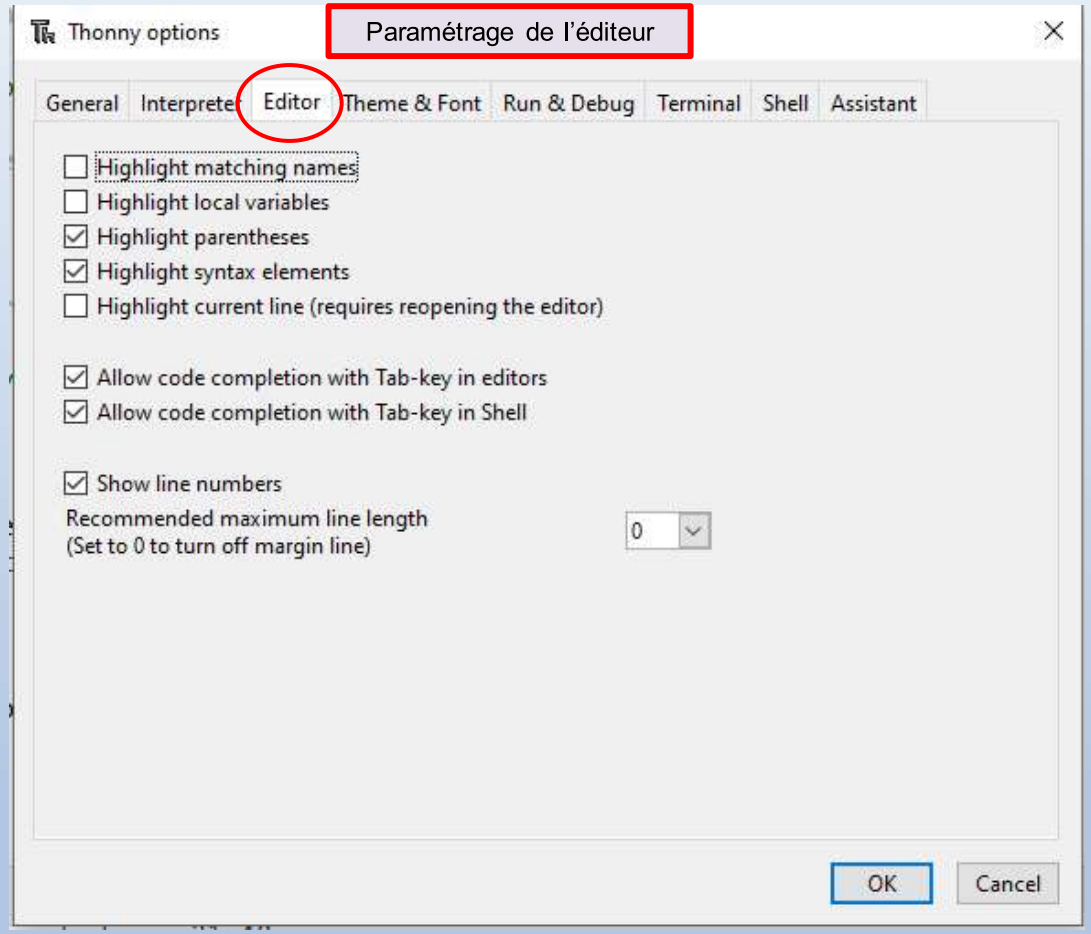
6. Paramétrage de l'IDE



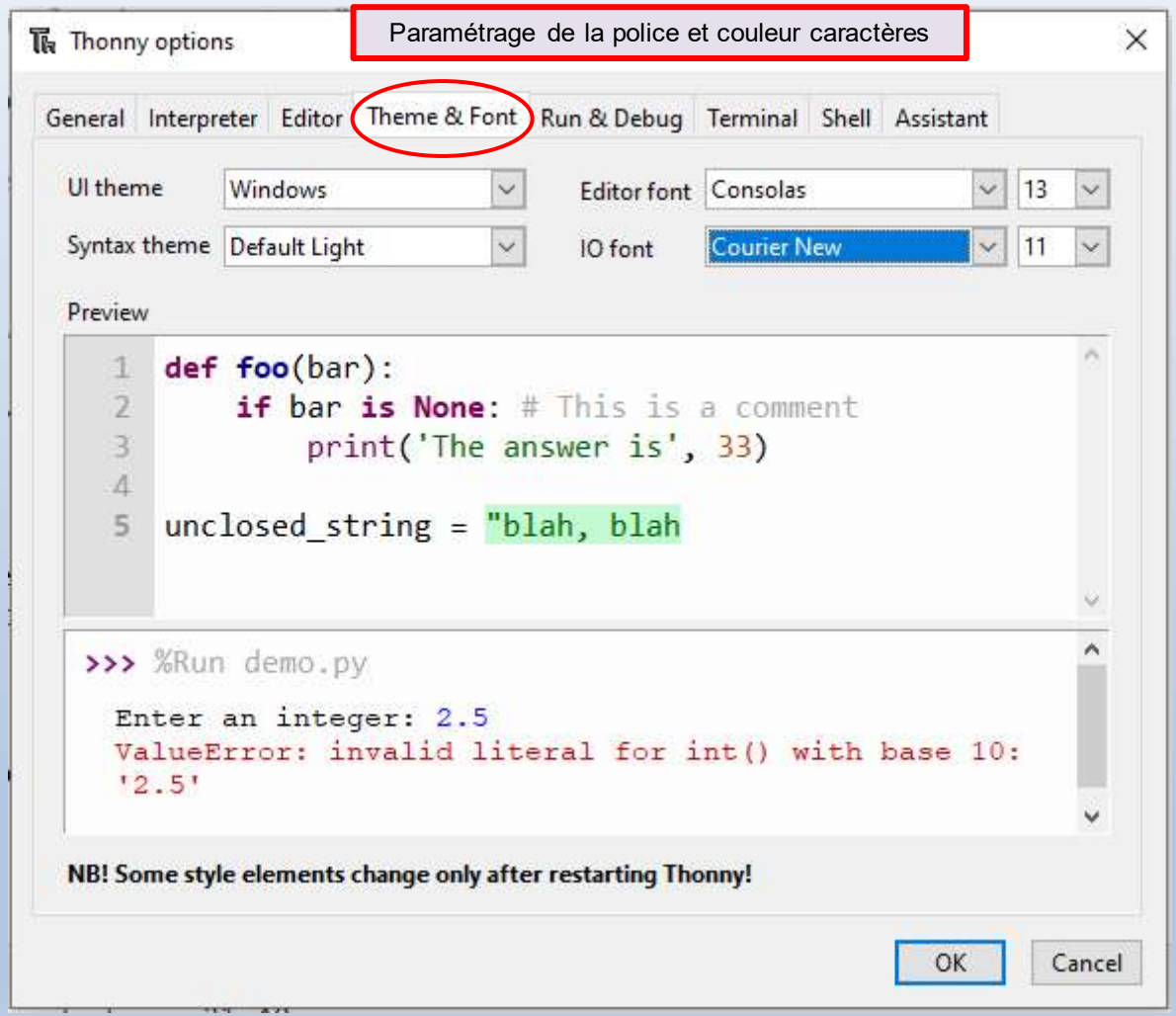
IDE Thonny


Thonny options
Paramétrage « choix du langage » par défaut Python interne Thonny
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
CircuitPython (generic)
MicroPython (BBC micro:bit)
MicroPython (ESP32)
MicroPython (ESP8266)
MicroPython (generic)
A special virtual environment (deprecated)
Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Which interpreter or device should Thonny use for running your code?
The same interpreter which runs Thonny (default)
Details
C:\Users\Dinozzo\AppData\Local\Programs\Thonny\python.exe
OK
Cancel

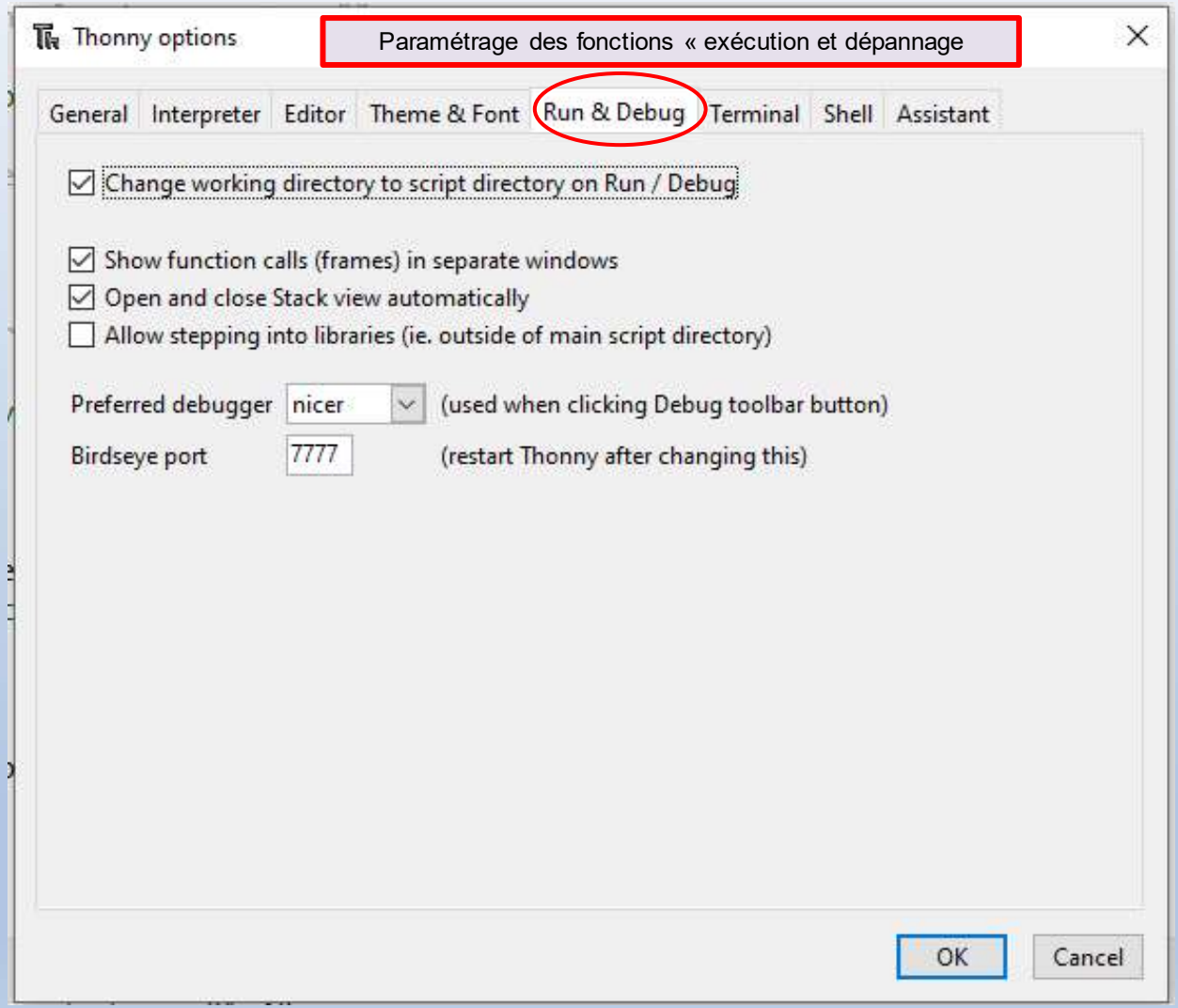
IDE Thonny


Thonny options
Paramétrage de l'éditeur
General
Interprete
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Highlight matching names
Highlight local variables
Highlight parentheses
Highlight syntax elements
Highlight current line (requires reopening the editor)
Allow code completion with Tab-key in editors
Allow code completion with Tab-key in Shell
Show line numbers
Recommended maximum line length
(Set to 0 to turn off margin line)
0
OK
Cancel

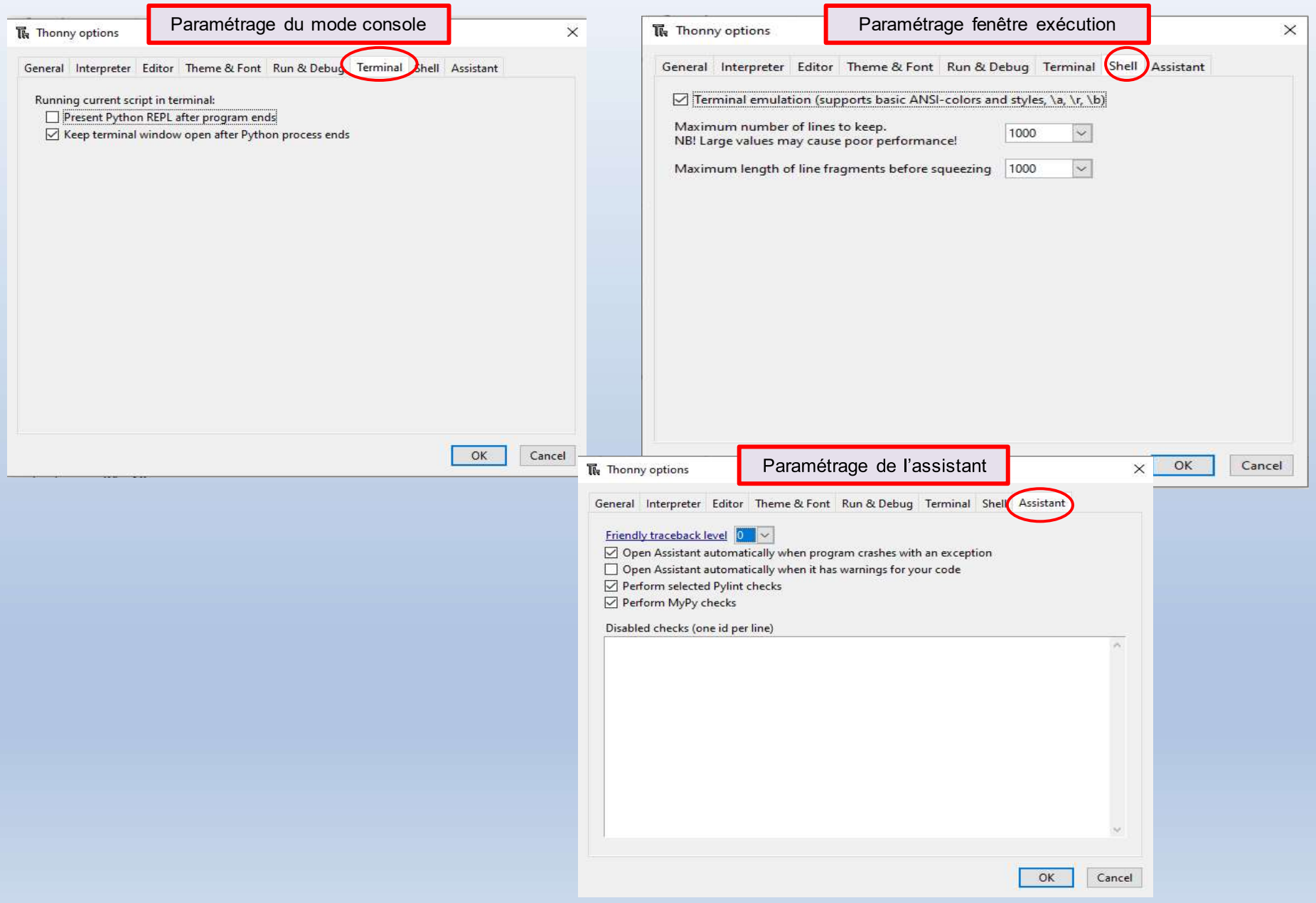
IDE Thonny


Paramétrage de la police et couleur caractères
Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
UI theme
Windows
Editor font
Consolas
13
Syntax theme
Default Light
IO font
Courier New
11
Preview
1 def foo(bar):
2 if bar is None: # This is a comment
3 print('The answer is', 33)
4
5 unclosed_string = "blah, blah
>>> %Run demo.py
Enter an integer: 2.5
ValueError: invalid literal for int() with base 10:
'2.5'
NB! Some style elements change only after restarting Thonny!
OK
Cancel

IDE Thonny


Thonny options
Paramétrage des fonctions « exécution et dépannage
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Change working directory to script directory on Run / Debug
Show function calls (frames) in separate windows
Open and close Stack view automatically
Allow stepping into libraries (ie. outside of main script directory)
Preferred debugger
nicer
(used when clicking Debug toolbar button)
Birdseye port
7777
(restart Thonny after changing this)
OK
Cancel

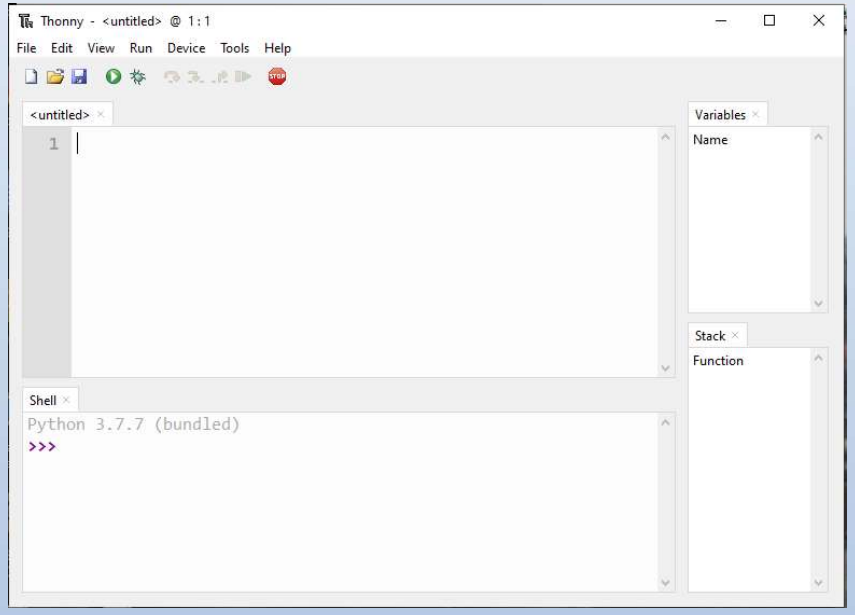
IDE Thonny


Paramétrage du mode console
Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Running current script in terminal:
Present Python REPL after program ends
Keep terminal window open after Python process ends
OK
Cancel
Paramétrage fenêtre exécution
Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Terminal emulation (supports basic ANSI-colors and styles, \a, \r, \b)
Maximum number of lines to keep.
NB! Large values may cause poor performance!
1000
Maximum length of line fragments before squeezing
1000
OK
Cancel
Paramétrage de l'assistant
Thonny options
General
Interpreter
Editor
Theme & Font
Run & Debug
Terminal
Shell
Assistant
Friendly traceback level
0
Open Assistant automatically when program crashes with an exception
Open Assistant automatically when it has warnings for your code
Perform selected Pylint checks
Perform MyPy checks
Disabled checks (one id per line)
OK
Cancel

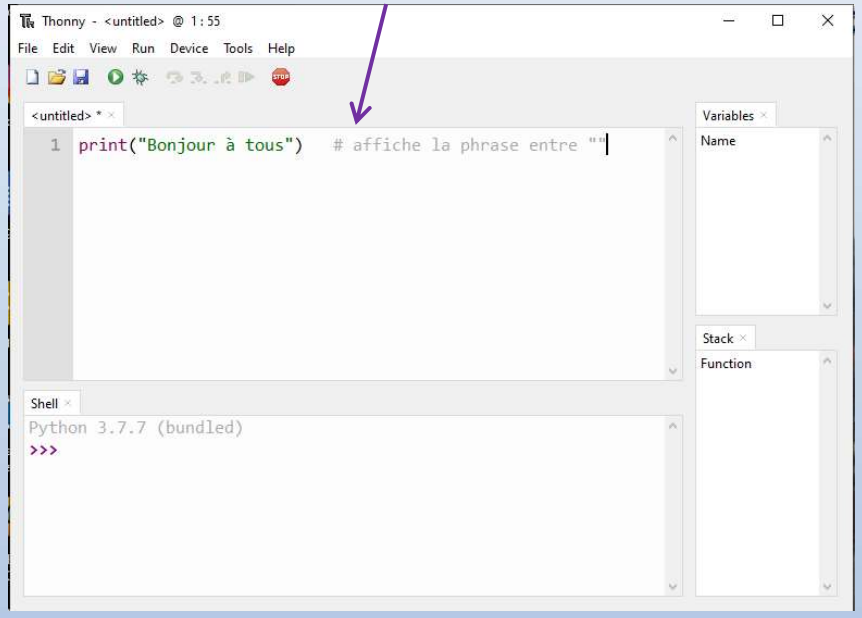
IDE Thonny

7. Prise en main

1. Ouverture de Thonny

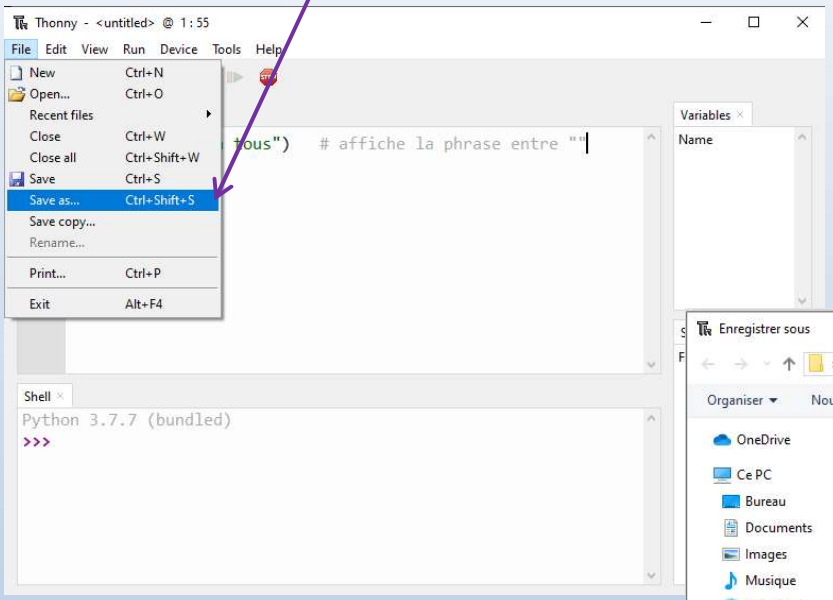


2. Ecriture d'une instruction

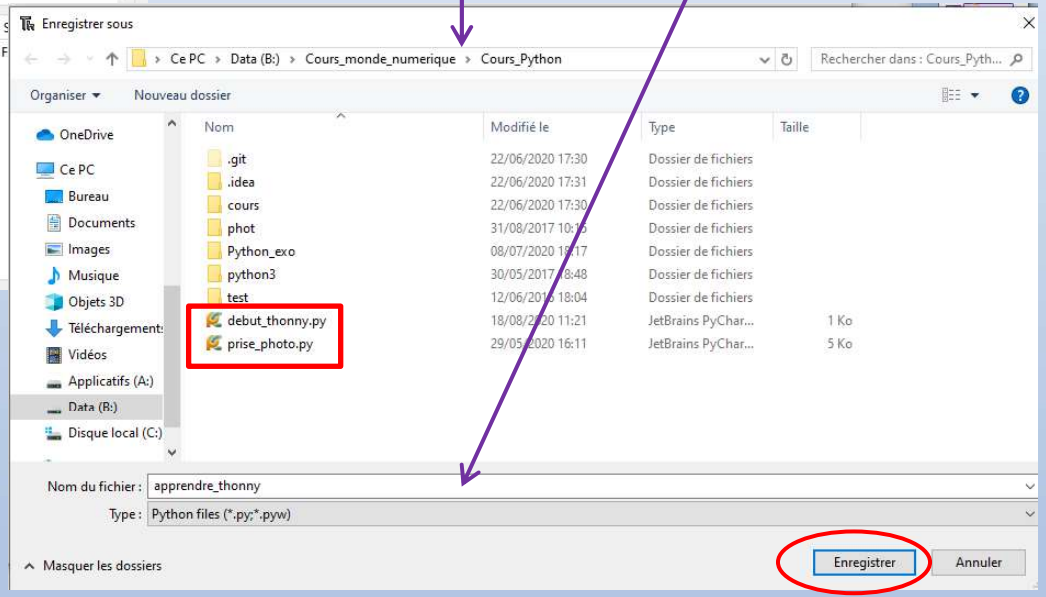


IDE Thonny

3. Sauvegarde du programme



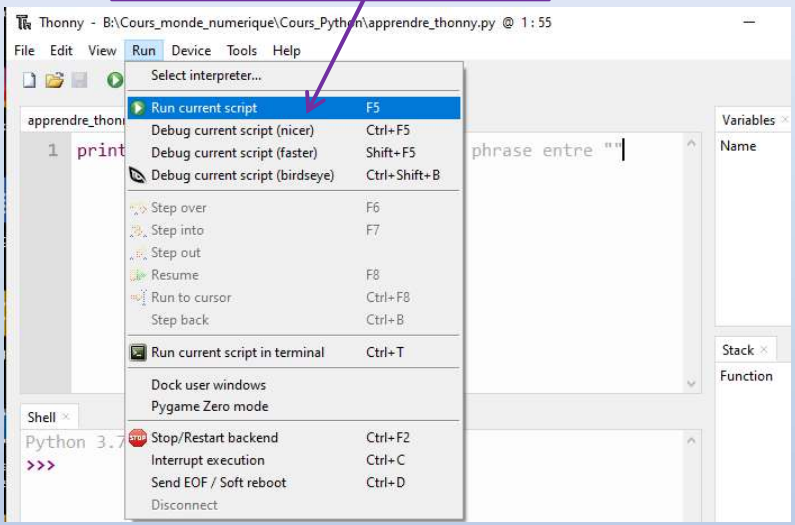
4. Répertoire de sauvegarde et nom de programme **.py**



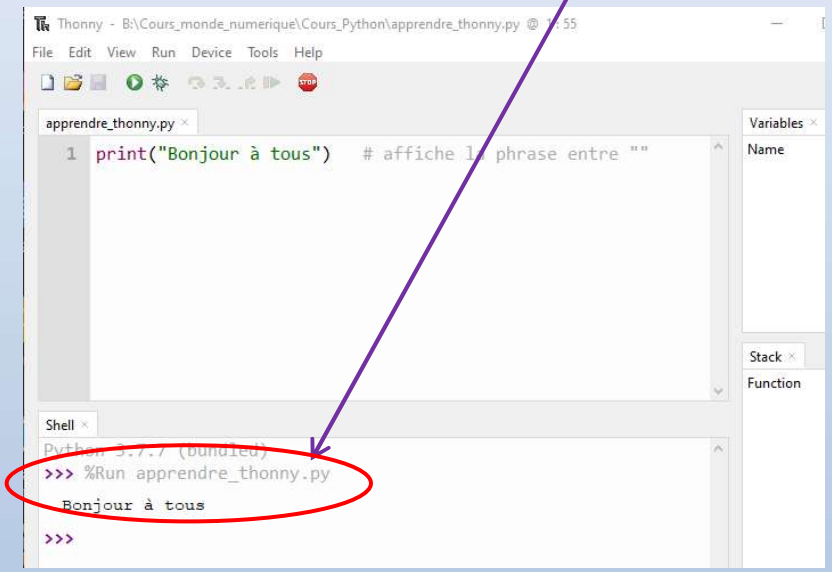
5. Prise en compte du nom

IDE Thonny

6. Exécution du programme

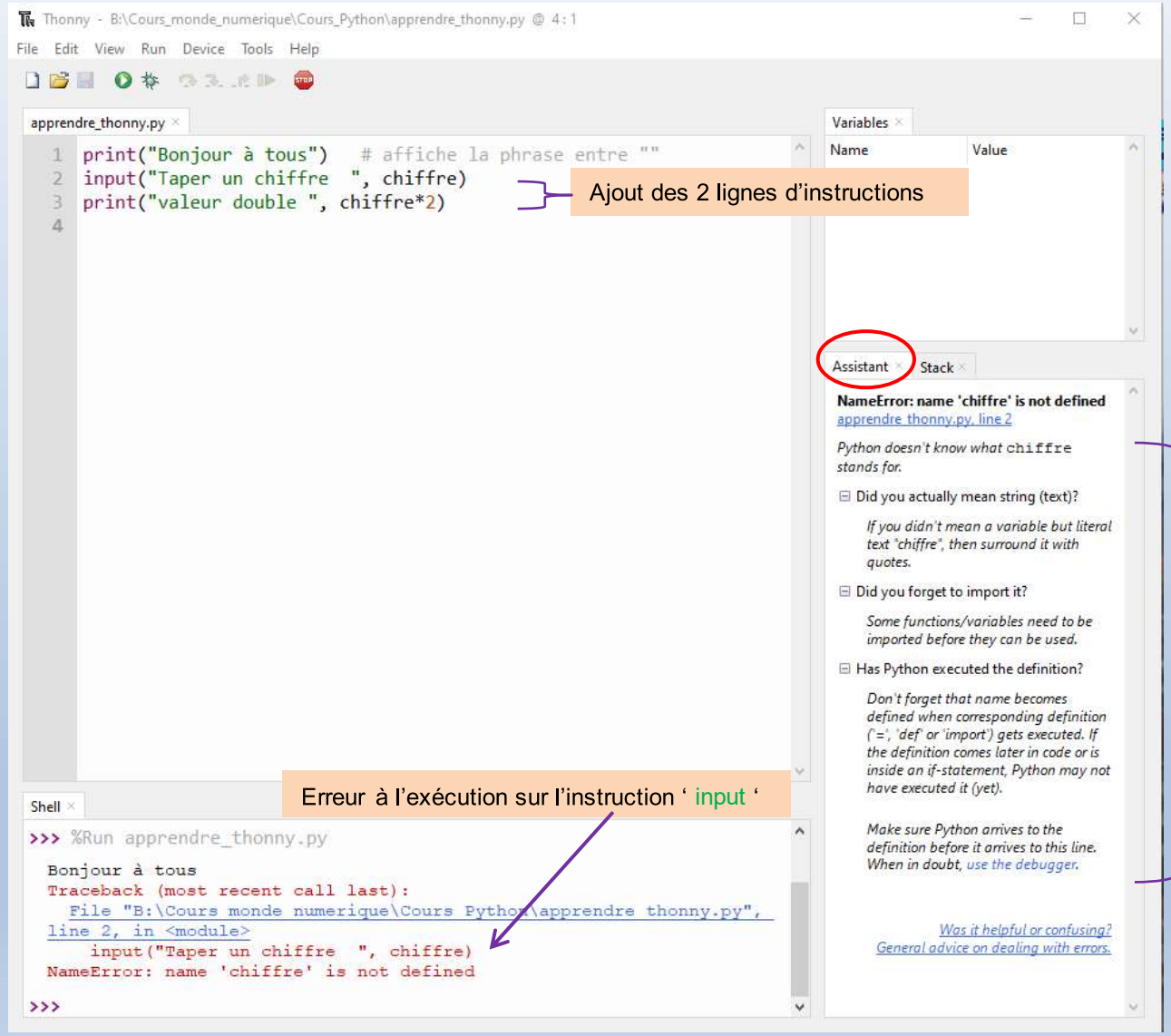


7. Affichage du résultat



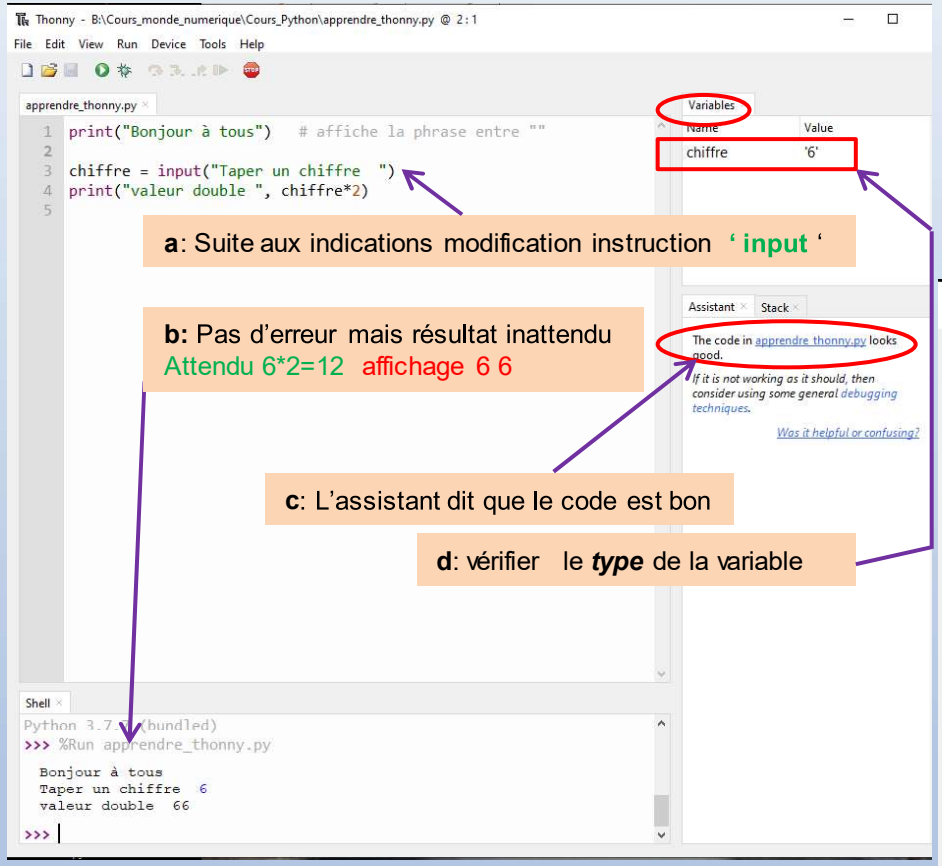
IDE Thonny

8. Affichage d'erreurs de programmation



IDE Thonny

9. Méthodologie de corrections (cas typage)



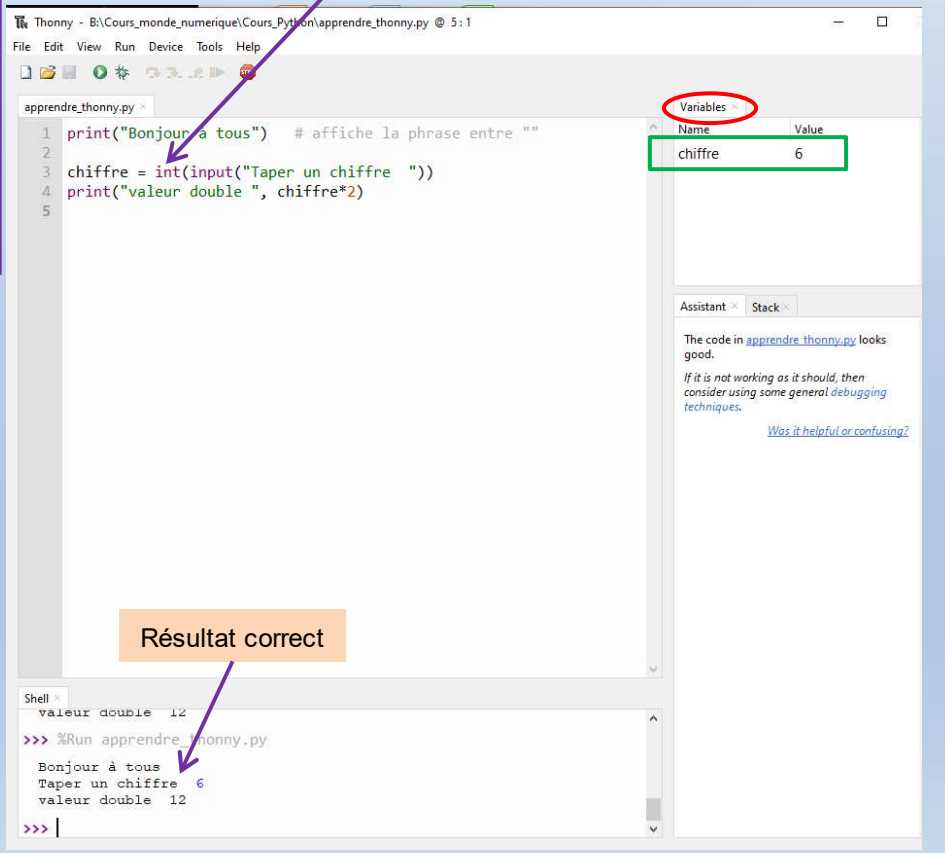
a: Suite aux indications modification instruction ' **input** '

b: Pas d'erreur mais résultat inattendu
Attendu 6*2=12 affichage 6 6

c: L'assistant dit que le code est bon

d: vérifier le ***type*** de la variable

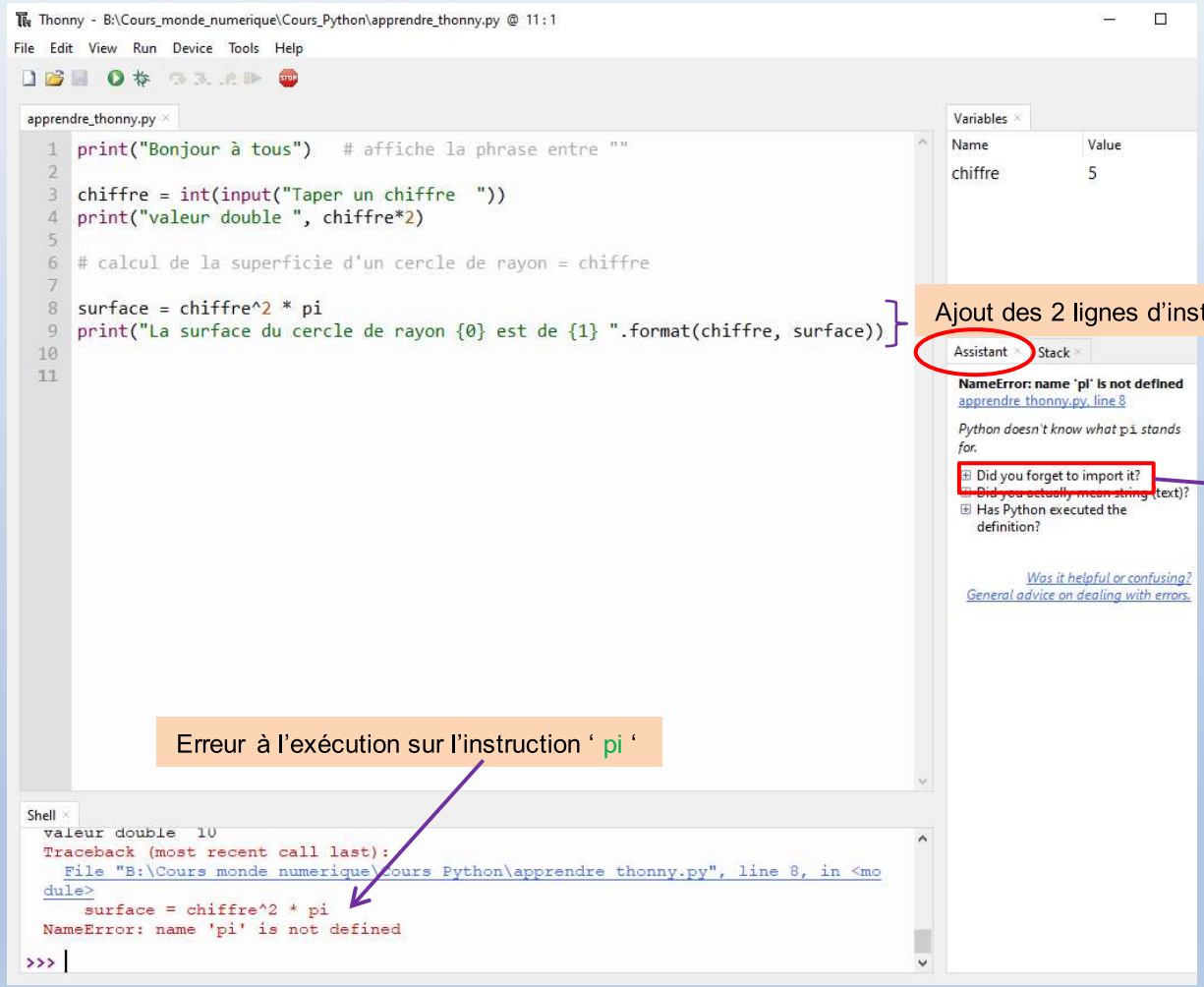
e: Affichage variable **'6'** *type* caractère
et non **6** type entier
alors redéfinir en entier par l'instruction ' **int** '



Résultat correct

IDE Thonny

9. Méthodologie de corrections (cas bibliothèque)



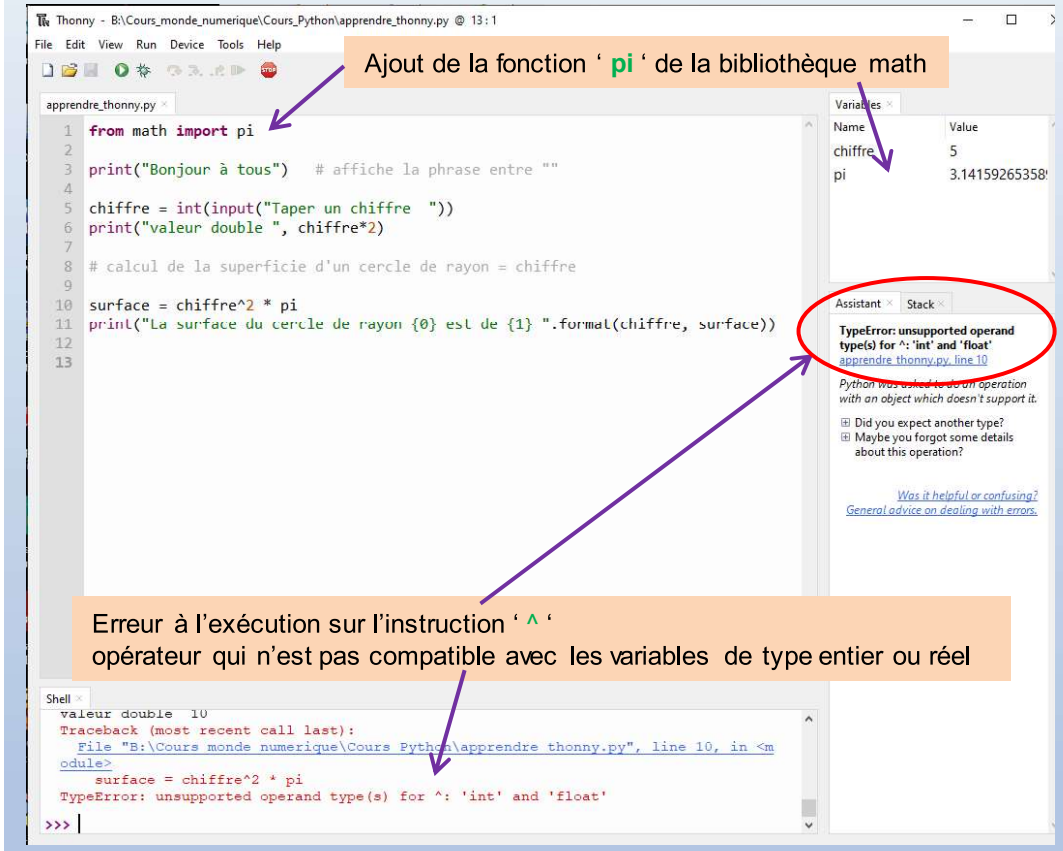
Ajout des 2 lignes d'instructions

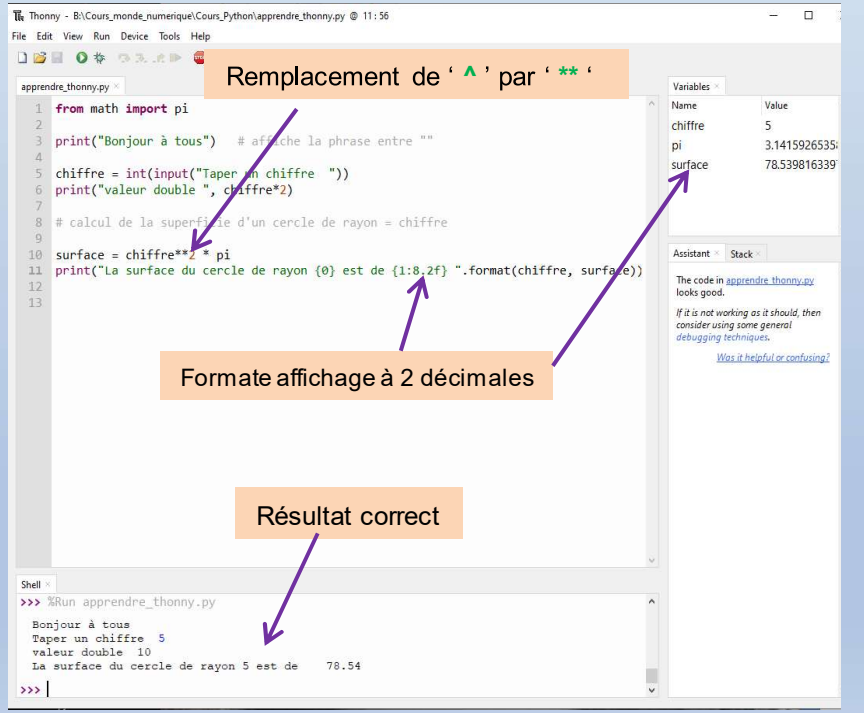
a: Assistant **import de pi**

Erreur à l'exécution sur l'instruction ' pi '

IDE Thonny

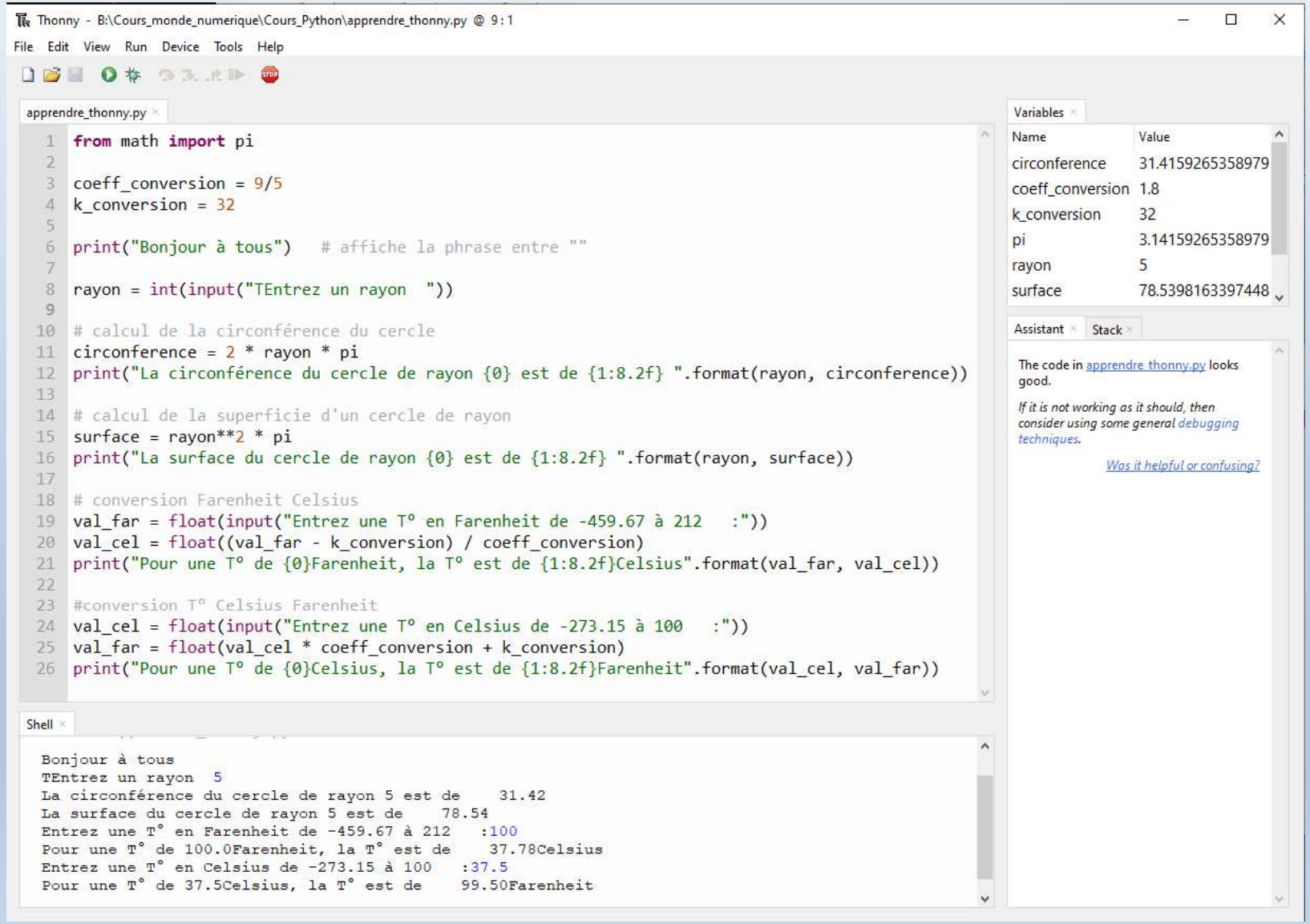
9. Méthodologie de corrections (cas bibliothèque)





IDE Thonny

10. Exemple de programme



Glossaire

Sigle	
IDE	**I**ntegrated **D**evelopment **E**nvironment
PC	**P**ersonal **C**omputer