Les cahiers d'Exercices en Programmation : Le langage Python

Apprendre à programmer et les mathématiques

Apprenez et entraînez vos acquis

- De très nombreux exercices à réaliser par vous-même

- Des corrigés regroupés à la fin du cahier et expliqués Pas à Pas.

AVANT-PROPOS

Ce livre est un cahier d'exercices : il vous propose des énoncés d'exercices et leurs corrigés. Vous allez apprendre le logiciel en vous entrainant à travers des exercices regroupés par thème.

Chaque énoncé vous présente l'exercice à réaliser. Vous trouverez à la fin du cahier le corrigé de chaque exercice. Certaines explications peuvent-être présentes.

METHODOLOGIE

Lors de la réalisation des exercices, vous pourrez remédier à certain problème à l'aide des corrections à la fin du cahier.

Après avoir réalisé tous les exercices de chaque chapitre vous allez pouvoir vérifier les compétences acquises à l'aide du tableau des objectifs.

Celui-ci sert à la cotation du professeur (grille d'évaluation).

Des **légendes ou recommandations** peuvent être présentes dans certains exercices. Celles-ci vous aideront dans vos recherches. Elles ne doivent pas être reproduites dans votre travail.

Chaque point de matière acquis dans un exercice peut être utilisé dans des exercices suivants sans explication.

Table des matières

Chapitre 1 : Introduction à Python

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Lancer l'interpréteur
Utiliser la fonction d'affichage : PRINT
Saisir des expressions (*, +, -, /, //, %, **)
Comprendre les expressions
Introduire des commentaires à l'aide du symbole : #

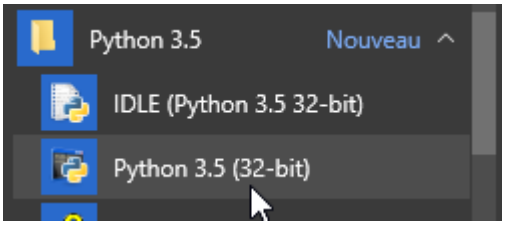
1. Qu'est-ce que le langage Python ?

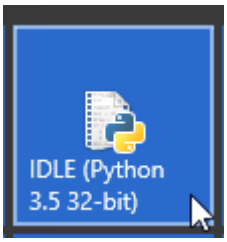
- Python est un langage de programmation interprété (il est lui traduit en quelque sorte au fur et à mesure de l'exécution par l'interpréteur), à ne pas confondre avec un langage compilé (langage qui est, avant de pouvoir l'exécuter, traduit en langage machine (binaire) par un compilateur).
- Il permet de créer toutes sortes de programmes, comme des jeux, des logiciels, des progiciels, etc.
- Il est possible d'associer des bibliothèques à Python afin d'étendre ses possibilités.
- Il est portable, c'est à dire qu'il peut fonctionner sous différents systèmes d'exploitation (Windows, Linux, Mac OS X,…).

2. Utiliser Python comme une calculette grâce à l'interpréteur de commande

Ce site peut te permettre d'utiliser Python Online (pas recommandé pour la syntaxe) : *https://www.tutorialspoint.com/execute_python_online.php*

Où tu peux aussi lancer l'interpréteur sous Windows (il doit être bien sûr installé) ou l'IDLE :

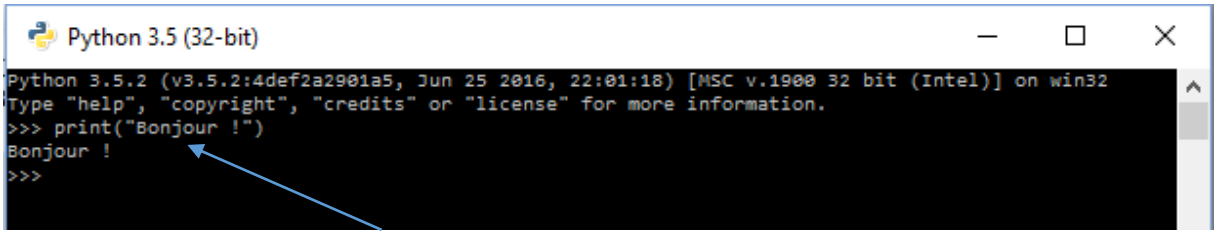




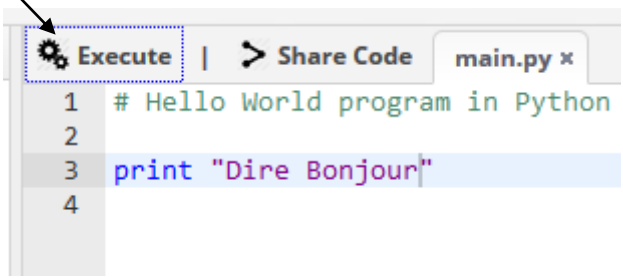
Et sous Mac OSX :

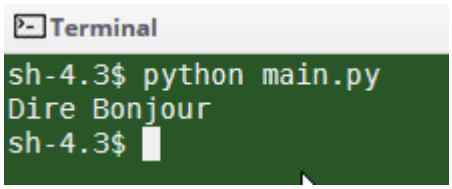
Cherchez un dossier `Python` dans le dossier `Applications`. Pour lancer Python, ouvrez l'application `IDLE` de ce dossier.

Exercice : Commençons l'apprentissage par l'affichage d'une salutation (commande print) et ensuite Enter.



Où à partir du site Python Online :





Ou avec Notepad ++. Mais le fichier doit être exécuté avec l'IDLE Python (je préfère cette manière de travailler).

La console Python (IDLE) pas le site Python Online fonctionne aussi comme une simple calculatrice.

Vous pouvez saisir une expression dont la valeur est renvoyée dès que vous appuyez sur la touche Enter.

Exercez-vous en tapant ces expressions :

```
>>> 2 * 5 + 6 - (100 + 3)
-87
>>> 7 / 2; 7 / 3
3.5
2.3333333333333335
>>> 34 // 5; 34 % 5 # quotient et reste de la division euclidienne de 34 par 5
6
4
>>> 2 ** 7  # pour l'exponentiation (et non pas 2^7 !
128
```

// : Obtenir la partie entière d'une division

% : Modulo ou reste de la division

: Introduction d'un

Chapitre 2 : Éditer, exécuter et sauvegarder un programme Python

Objectif(s) de ce chapitre.

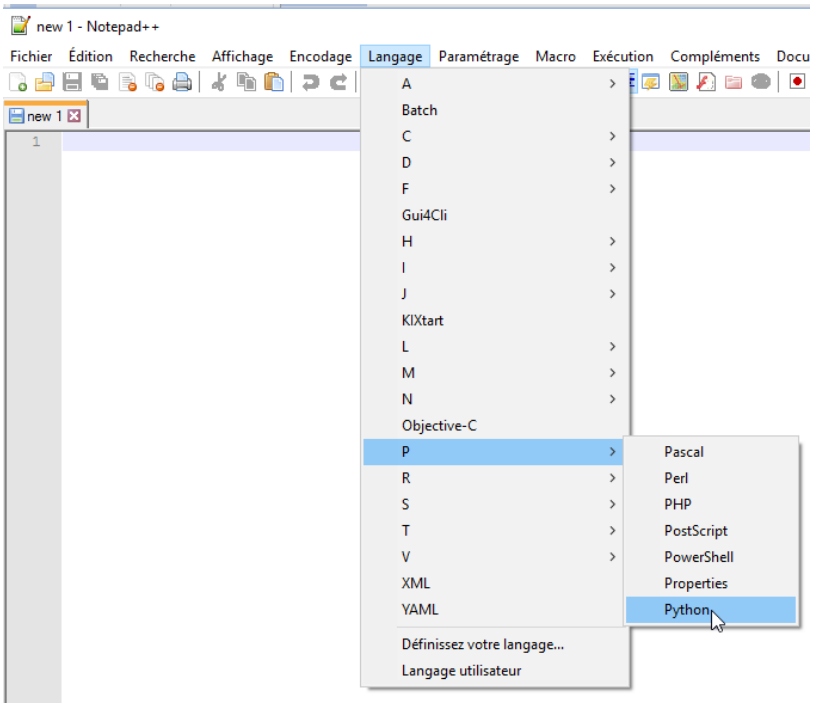
À la fin de ce chapitre, l'élève sera capable de :

Exécuter Notepad ++
Spécifier un langage
Créer et sauvegarder un programme avec une extension Python : .py
Ouvrir et lancer un programme créé

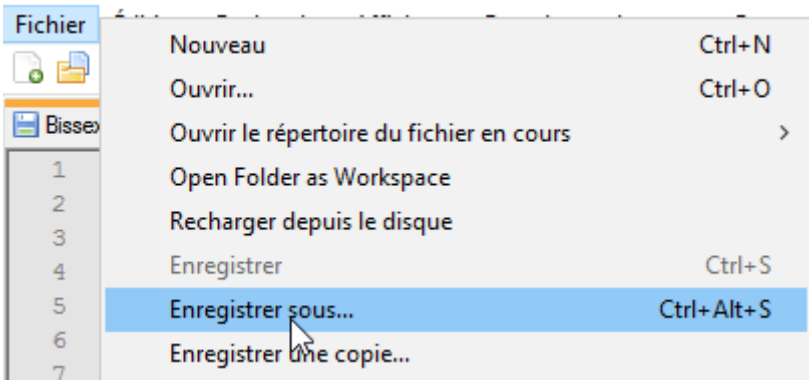
1. Utilisation de Notepad ++

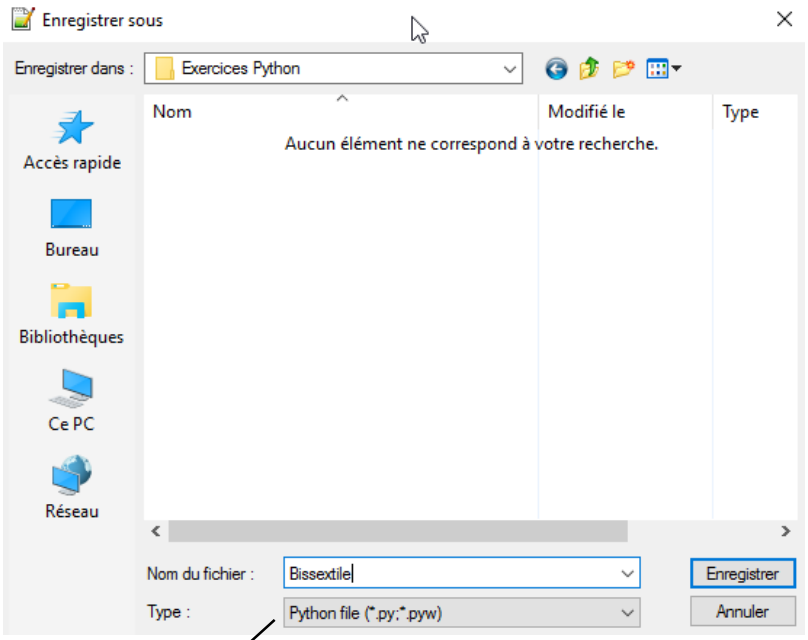
Exécuter Notepad ++

Spécifier que le langage utilisé sera en Python.



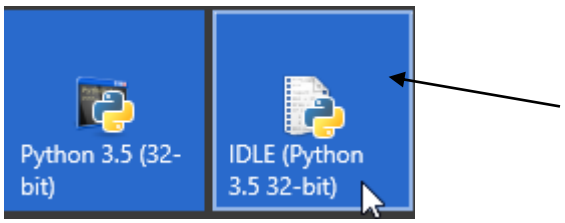
2. Sauvegarde du programme

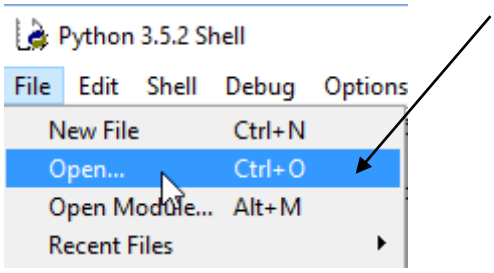


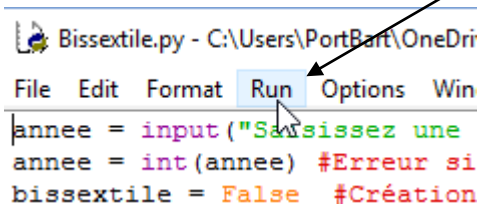


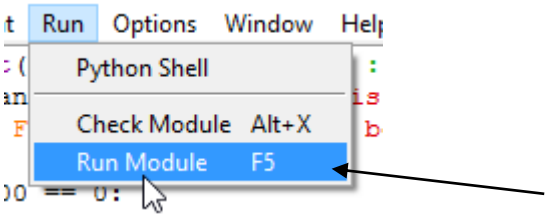
Extension Python

3. Exécuter le programme









Chapitre 3 : Le monde des variables

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Affecter une valeur à une variable
Connaître le type d'une variable : TYPE
Retourner un identifiant unique pour un objet : ID
Réaliser des opérations arithmétiques simples

1. Variables et affectations

Une affectation se fait à l'aide du symbole "=". Elle est gardée en mémoire pour une utilisation ultérieure.

Taper ces expressions dans l'IDLE Python :

```
>>> a = 128
>>> a
128
>>> a, id(a), type(a)
(128, 137180768, <class 'int'>)
>>> 2 * a
256
```

a = 128 : On affecte ici à la variable a la valeur 128. Un ordre est à garder dans l'affectation. On ne peut pas écrire comme ligne de commande 128 = a qui aboutira à une erreur dans l'interpréteur.

id(a) : retourne un identifiant unique pour chaque objet.

type(a) : savoir de quel type est une variable. a appartient donc à la classe des entiers.

En résumé :

- Les variables permettent de conserver dans le temps des données de votre programme.
- Vous pouvez vous servir de ces variables pour différentes choses : les afficher, faire des calculs avec, etc.
- Pour affecter une valeur à une variable, on utilise la syntaxe nom_de_variable = valeur.

- Il existe différents types de variables, en fonction de l'information que vous désirez conserver : int, float, chaîne de caractères etc.
- Pour connaitre le type d'une variable, on utilise la fonction `type`
- Pour afficher une donnée, comme la valeur d'une variable par exemple, on utilise la fonction `print`.

2. Exercices de synthèse

Ex1. Quel est le type de la variable 3.4 :

Ex2. Quel est le type de la variable "Un essai" :

Ex3. Trouver les lignes de commandes pour :

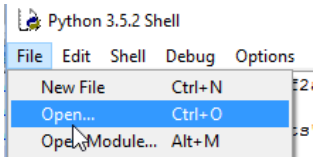
- Affecter la valeur 3 à la variable `a` ;
- Afficher la variable `a` ;
- Augmenter de 3 la valeur de la variable `a` ;
- Affecter à la variable `b` la valeur `a - 2` ;
- Afficher ceci : `a = 6 et b = 4`

Ex4. Introduit cet exercice avec Notepad++. N'oublie pas de changer le langage.

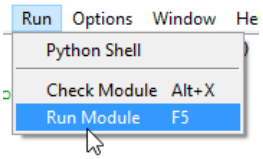
```
1  print("quatre plus quatre =")
2  print(4+4)
3  print("huit fois huit =")
4  print(8*8)
```

Enregistre-le sous le nom Opérations

Lance-le à l'aide de l'IDLE :



Une fois le fichier ouvert, tu dois l'exécuter :



Chapitre 4 : Les structures conditionnelles

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Comprendre et savoir utiliser une condition IF et un bloc d'instructions
Comprendre et savoir utiliser dans une condition les mots-clés (if, elif et else)
Savoir utiliser les opérateurs de comparaison
Comprendre la notion d'indentation
Comprendre et savoir utiliser le type booléen
Comprendre et savoir utiliser les mots-clés (and, or et not)
Comprendre et savoir utiliser le mot-clé (is)

1. Les structures conditionnelles

Les instructions ne sont plus ici linéaires : dans le chapitre précédent l'interpréteur exécutait au fur et à mesure le code que vous saisissiez dans la console.

La structure conditionnelle que nous allons inclure dans les programmes va nous permettre de réaliser des tests et d'aller plus loin dans la programmation.

Les conditions permettent d'exécuter une ou plusieurs instructions dans un cas et d'autres instructions dans un autre cas.

2. La première condition et bloc d'instructions : la forme minimale en IF

Les conditions vont vous permettre de faire une action précise si une variable est par exemple positive et une autre action si celle-ci est négative ou une troisième action si la variable est nulle.

Ex5 : Lancer la console Python et taper ce premier exemple :

```
>>> # Premier exemple de condition
>>> a=5
>>> if a >0 : print (a," est plus grand que 0")

5  est plus grand que 0
```

Taper deux fois sur la touche Enter pour que le programme se lance.

Un bloc d'instruction = série d'instructions.

Donc, on attribue à la variable a la valeur de 5.

Et on test : Si la valeur de a est strictement plus grande que 0 alors j'affiche à l'écran a est plus grand que 0 sinon on écrit rien.

Le Si = mot clé IF.

Ex6 : Tester le même programme mais en attribuant à la variable a la valeur de -2.

Votre programme écrit quoi ?

Ex7 : Taper ce second exemple comportant un bloc d'instructions :

```
>>> #Deuxième exemple avec un bloc d'instructi
>>> a = 5
>>> b = 8
>>> if a > 0:
        # On incrémente la valeur de b
        b += 1
        # On ajoute les valeurs des variables
        print("a=",a," et b=",b)


a= 5  et b= 9
```

Notion importante : l'indentation.

On entend par indentation un certain décalage vers la droite, obtenu par un (ou plusieurs) espaces ou tabulations.

C'est un moyen pour l'interpréteur de savoir où se trouvent le début et la fin d'un bloc.

3. La forme complète (if, elif et else)

La première forme de condition est assez incomplète.

Considérons, par exemple, une variable a de type entier. On souhaite faire une action si cette variable est positive et une action différente si elle est négative.

L'instruction else (Ex8) :

Le mot-clé else, qui signifie « sinon » en anglais, permet de définir une première forme de complément à notre instruction if.).

```
age=21
if age >=18:
        print("Vous êtes majeur.")
else:
        print("Vous êtes mineur.")
```

If et Else : même niveau d'indentation. Et si a valait 0 ?

L'instruction elif (Ex9) :

Le mot clé elif est une contraction de « else if », que l'on peut traduire par « sinon si ». Dans l'exemple que nous venons juste de voir, l'idéal serait d'écrire :

- si a est strictement supérieur à 0, on dit qu'il est positif ;
- sinon si a est strictement inférieur à 0, on dit qu'il est négatif ;
- sinon a ne peut qu'être égal à 0, on dit alors que a est nul.

```
1  a=0
2  if a >0:
3      print("a est positif.")
4  elif a<0:
5      print("a est négatif.")
6  else :
7      print("a est nul")
```

4. Autres opérateurs de comparaison

Opérateur	Signification littérale
<	Strictement inférieur à
>	Strictement supérieur à
<=	Inférieur ou égal à
>=	Supérieur ou égal à
==	Égal à
!=	Différent de

La condition entre le if et le : cela se nomme un prédicat.

Tester les comparaisons suivantes directement en ligne de commande à partir du programme Python 3.5.

```
>>> a = 0
>>> a == 5
False
>>> a > -8
True
>>> a != 85.54
True
```

5. Le type Booléen (bool)

Au point précédent, vous pouvez remarquer que l'interpréteur renvoie un `True` ou un `False` (Vrai ou faux).

Ce sont les 2 valeurs de type booléen.

Le `T` et le `F` majuscule sont obligatoires pour que Python les comprenne.

Voici un exemple (Ex10) :

```
1  age = 21
2  majeur = False
3  if age >=18:
4      majeur = True
```

Que vaut la valeur booléenne de la variable majeur ?

Et comment afficher sa valeur en ligne de commande ?

```
>>> print(majeur)
True
>>>
```

6. Les mots-clés and, or et not

Utilité : tester plusieurs prédicats dans une condition.

Voici un exemple d'une condition situant un entier entre deux intervalles sans les mots-clés (Ex11) :

```
1  a = 5
2  if a >= 2:
3      if a <= 8:
4          print("a est dans l'intervalle.")
5      else :
6          print("a n'est pas dans l'intervalle.")
7  else :
8      print("a n'est pas dans l'intervalle.")
```

Assez lourd comme code !

Nous allons utiliser maintenant le mot clé and (et) et transformer l'exemple précédent (Ex12).

```
a=5
if a>=2 and a <=8:
    print("a est dans l'intervalle.")
else:
    print("a n'est pas dans l'intervalle.")
```

Plus simple et plus compréhensible.

Exercice Ex13 : Essayer de transformer le programme précédent en utilisant le mot clé or (ou) (la réponse finale doit-être identique).

Enfin il existe le mot-clé qui "inverse" un prédicat.

Tester ce programme (Ex14) :

```
majeur = False
if majeur is not True:
    print("Vous n'êtes pas encore majeur.")
```

Ajout à la liste d'un nouveau mot-clé : is.

Ce mot clé teste l'égalité non pas des valeurs de deux variables, mais de leurs références.

Chapitre 5 : Votre premier programme

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Comprendre et savoir utiliser la fonction Input()
Tester si un nombre est multiple d'un autre
Comprendre et savoir utiliser la matière vue dans les chapitres précédents

1. La fonction Input()

Input() : Attente d'une valeur saisie par l'utilisateur. Cette valeur est attribuée à une variable.

Cette fonction accepte un paramètre entre ses parenthèses : message affiché à l'utilisateur.

Ce message est encadré de guillemets " ".

Taper cet exemple :

```
1    annee = input("Saisissez une année : ")
```

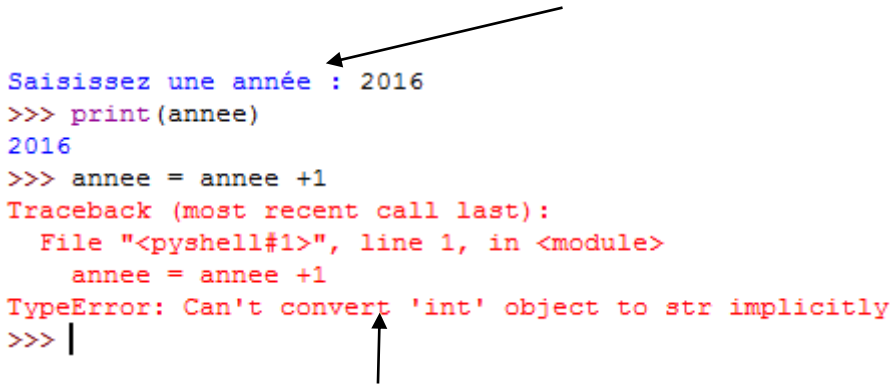
Lors de l'exécution du programme avec l'IDLE, introduisez la valeur : 2016.

Comment afficher la valeur introduite par l'utilisateur à partir du Shell Python ?

```
Saisissez une année : 2016
>>> print(annee)
2016
>>>
```

Un petit problème est quand-même présent. Le type de l'année est une chaîne de caractère et nous voulions travailler sur un entier.

Une vérification peut-être réalisée en tapant ceci :



Et vous apercevrez ce message d'erreur.

Il faudra donc convertir cette variable à l'aide de la fonction int() qui prend en paramètre la variable d'origine.

Ajouter ces lignes de commandes à la suite (toujours dans le Shell Python) :

```
>>> annee=int(annee)
>>> type(annee)
<class 'int'>
```

Type indique le type de la variable qui est bien maintenant un entier.

Réaliser un Quizz simple (Ex16)

Introduit ce code :

```
1  a=input("Quelle est la capitale de l'Angleterre ? : ")
2  if a=="Londres":
3      print("Correct")
```

La commande Si vérifie si une information est vraie. La ligne suivante s'exécute si c'est le cas.

2. Comment tester des multiples

Comment tester si un nombre a est multiple d'un nombre b ?

Il suffit de tester le reste de la division entière de b par a. Si ce reste est nul, alors a est un multiple de b.

Taper ces deux commandes (Shell Python :

```
>>> 5%2 # Le reste = 1 donc 5 n'est pas un multiple de 2
1
>>> 8%2 # Le reste = 0 donc 8 est un multiple de 2
0
```

À vous de jouer

Tous les éléments nécessaires pour réussir les exercice sont à vos dispositions. Si vous avez du mal passez à la correction et étudiez-la soigneusement. Il n'y a pas qu'une seule solution. Bonne chance !

Note : Avant de commencer à travailler, sachez qu'il vous sera plus simple de taper votre programme dans un éditeur (notepad ++) et de le sauvegarder. Celui-ci pourra être exécuté par la suite à l'aide du programme Idle (voir les explications dans le chapitre suivant).

Exercice : Ex15 : Déterminer si une année saisie par l'utilisateur est bissextile.

Règle : Une année est dite bissextile si c'est un multiple de 4 sauf si c'est en même temps un multiple de 100. Mais elle est quand même considérée comme bissextile si c'est un multiple de 400.

Démarche à suivre :

- Si une année n'est pas multiple de 4, le programme s'arrête et on affiche : L'année introduite n'est pas bissextile ;
- Maintenant si elle est multiple de 4, on regarde si elle est multiple de 100 ;
    - Si c'est le cas, on regarde si elle est multiple de 400 ;
        - Si c'est le cas, on affiche l'année est bissextile ;
        - Sinon, on affiche elle n'est pas bissextile ;
    - Sinon, on affiche l'année est bissextile.

Exercice Ex16 : Crée un Quizz en Python avec 5 questions sur les capitales – Pays.

Chapitre 6 : Les boucles

Objectif(s) des exercices de ce chapitre.

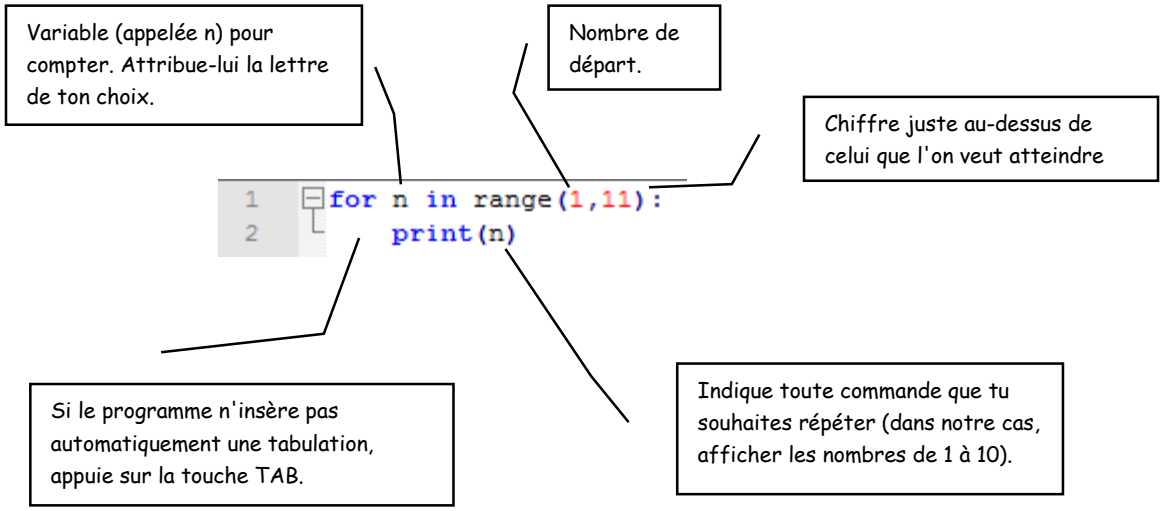
À la fin de ce chapitre, l'élève sera capable de :

Comprendre et savoir utiliser la boucle for
Définir un nombre de départ et le nombre que l'on veut atteindre

1. À quoi ça sert ?

Afficher de manière répétée des choses à l'écran.

Une boucle est une série de commande répétée un certain nombre de fois.



Comment compter alors jusque 100 (Ex17) ?

```
1 for n in range(1,101):
2     print(n)
```

2. À ton tour ?

Teste toutes ces boucles et enregistre tes travaux :

Ex18 :

```
for n in range(1,21):
    print(n)
```

Ex19 :

```
for n in range(1,51):
    print(n)
```

Ex20 :

```
for a in range(1,201):
    print(a)
```

Ex 21 :

```
for b in range(1,101):
    print(b*10)
```

Ex22 :

```
for c in range(1,101):
    print(c*100)
```

Chapitre 7 : Formes graphiques

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Importer et utiliser une bibliothèque
Utiliser une boucle pour dessiner une forme

1. Tortue tortueuse

Crée ce programme avec Notepad ++, enregistre-le et exécute le à partir de l'IDLE Python (Ex23).

```
1  from turtle import *
2  forward(200)
```

Ce programme dessine une flèche vers la gauche.

Ajoute ces lignes à ce programme.

```
1  from turtle import *
2  forward(200)
3  right(90)
4  forward(200)
5  right(90)
```

Après exécution du programme, analyse les modifications à l'écran et essaie maintenant de dessiner un carré complet.

2. Boucles et formes

Introduisons maintenant une boucle pour dessiner une forme (Ex24).

```
1  from turtle import *
2  for n in range(0,4):
3      forward(200)
4      right(90)
```

Le forme dessinée à l'aide du programme Ex23 et Ex24 est identique. Mais tu peux remarquer qu'en utilisant une boucle, le programme devient plus court.

3. Exercices

Teste toutes ces boucles et enregistre tes travaux :

Ex25 :

```
1  from turtle import *
2  for n in range(0,6):
3      forward(200)
4      right(60)
```

Ex26 :

```
1  from turtle import *
2  for n in range(0,8):
3      forward(100)
4      right(45)
```

Ex27 :

```
1  from turtle import *
2  for n in range(0,5):
3      forward(200)
4      right(72)
```

Ex 28 :

```
1  from turtle import *
2  for n in range(0,5):
3      forward(200)
4      right(144)
```

Chapitre 8 : Mode aléatoire

Objectif(s) des exercices de ce chapitre.

À la fin de ce chapitre, l'élève sera capable de :

Comprendre et savoir utiliser la commande Random (nombre aléatoire)

1. Nombre au hasard

Pour obtenir un résultat au hasard par exemple pour un lancer de dé, on utilise en informatique le mode aléatoire à l'aide de la commande Random.

```
1   from random import *
```

random : recherche d'un nombre au hasard
import * : chercher des commandes dans la bibliothèque.

Affiche maintenant un nombre aléatoire entre 1 et 6 à l'aide de ce programme (Ex29) :

```
1   from random import *
2   print(randint(1,6))
```

2. Pile ou face

Tirage du pile ou face à l'aide de ce programme (Ex30) :

```
1   from random import *
2   pièce=["pile","face"]
3   print(choice(pièce))
```

choice : permet à Python de choisir un mot au hasard dans une liste "pièce"

3. Faire un sandwich

Dans cet exercice, il faut créer deux listes d'ingrédients et ensuite faire choisir au hasard un ingrédient à partir de chaque liste (Ex31).

```
1  from random import *
2  f1=["fromage","oeuf","confiture"]
3  f2=["carotte","salade","oignon"]
4  for s in range(0,10):
5      print("Vos 10 sandwichs seront à base de ",choice(f1)," et de",choice(f2))
```

Solutions des exercices

Exercices page 5 :

2.1 : <class 'float'>

2.2 : <class 'str'>

2.3 :

```
>>> a = 3
>>> print(a)
>>> a = a + 3
>>> b = a - 2
>>> print("a =", a, "et b =", b)
```

3.4 :

```
>>> if a < 2 or a > 8:
...     print("a n'est pas dans l'intervalle.")
... else:
...     print("a est dans l'intervalle.")
...
a est dans l'intervalle.
```

4.5 :

*Bissextile.py - C:\Users\PortBart\OneDrive\Nouveaux cours 2016\Nouveaux cours\Exercices Python\...

File Edit Format Run Options Window Help

```
annee = input("Saisissez une année :") #Année saisie par l'utilisateur
annee = int(annee) #Erreur si jamais l'utilisateur n'a pas saisi un nombre
bissextile = False  #Création d'un booléen qui vaut vrai ou faux
                                    # selon que l'année est bissextile ou non
if annee % 400 == 0:
        bissextile = True
elif annee % 100 == 0:
        bissextile = False
elif annee % 4 == 0:
        bissextile = True
else:
        bissextile = False

if bissextile: #Si l'année est bissextile on affiche
        print("L'année saisie est bissextile.")
else : #Sinon on affiche
        print("L'année saisie n'est pas bissextile.")
```

Ex23 :

```
1  from turtle import *
2  forward(200)
3  right(90)
4  forward(200)
5  right(90)
6  forward(200)
7  right(90)
8  forward(200)
9  right(90)
```

Bibliographie

- Apprendre à coder niveau 3 – Vigot
- Tangente Éducation : Spécial programmation n°15