

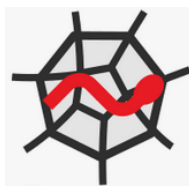


Initiation à la programmation python

Sommaire :

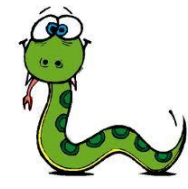
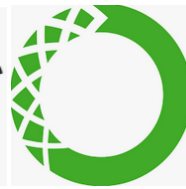
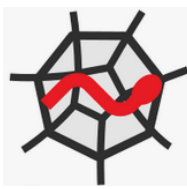


1 Environnements de développement.....	3
1.1 Utiliser un environnement autonome : pyzo.....	3
1.2 Utiliser IDLE python 3.x.....	4
1.3 Utiliser anaconda.....	5
2 Premier pas : utilisation de la console	6
2.1 La bibliothèque math	7
a) Import au cas par cas :.....	7
b) Import global :	7
c) Liste du contenu d'une bibliothèque	7
2.2 Calculs avec la console	7
2.3 Quelques opérateurs.....	8
2.4 Faire des tests, des comparaisons de variables.....	8
2.5 Un premier petit programme	10
2.6 Éléments de syntaxe : Précision sur les noms de variables.....	11
3 Plus loin avec le langage.....	12
3.1 Utilisation de fonctions.....	12
3.2 Un convertisseur distance / temps.....	13
3.3 Le formatage des données en sortie méthode recommandée si python < 3.6.....	14
3.4 fstring formatage des données en sortie méthode recommandée si python >= 3.6.....	15
a) Le principe.....	15
b) Formater les données	15
3.5 Le formatage des données en sortie 'méthode ancienne'	17
3.6 Quelques possibilités de mise en forme	20
3.7 La structure tant ... que while.....	20
3.8 Les répétitions avec des boucles	21
3.9 Retour sur les fonctions avec plusieurs paramètres	22
3.10 Les listes et les tables : conjuguons un verbe.....	23
4 Générer du hasard	25
4.1 Génération de hasard avec des nombres : fonctions seed(), random(), randint()	25
4.2 Génération du hasard avec des listes : fonctions choice(), shuffle(), sample()	26
4.3 Les dés sont pipés synthèse avec une liste et la fonction choice()	27
4.4 Les dés sont pipés synthèse avec random().....	28
5 S'amuser avec le module tortue.....	29
5.1 Une première étoile.....	29
5.2 La liste d'une partie des commandes 'tortue' disponibles.....	30
6 Réglages divers	31
6.1 Réglage du mode d'interprétation des caractères accentués norme UNICODE	31
6.2 Accès à des caractères spéciaux.....	31



7	Quelques points avancés	32
7.1	<i>Les logarithmes.....</i>	32
7.2	<i>Mesure des temps d'exécution du programme.....</i>	32
a)	Pour python < 3.8 la librairie time	32
b)	Pour python > = 3.8 la librairie process_time	33
8	Configuration de pyzo	34
8.1	<i>Configuration avec l'outil Pyzo.....</i>	34
8.2	<i>Choix du répertoire de travail dans le code.....</i>	36
8.3	<i>En cas de plantage.....</i>	36
9	Aide mémoire turtle	37
10	Installation des logiciels sous pyzo.....	38
11	Installation de Python 3.7 et des bibliothèques 'NSI'	40
11.1	<i>Installation de python37.....</i>	40
11.2	<i>Méthode pour l'ajout des modules.....</i>	40
11.3	<i>Installation des modules avec PIP.....</i>	40
11.4	<i>Installation sans modules whl</i>	41
11.5	<i>Un script python pour tester que tout est bien installé</i>	41
11.6	<i>Un traitement par lot pour l'installation des modules</i>	42
11.7	<i>Sous W10.....</i>	42
12	Ressources	43






1 Environnements de développement

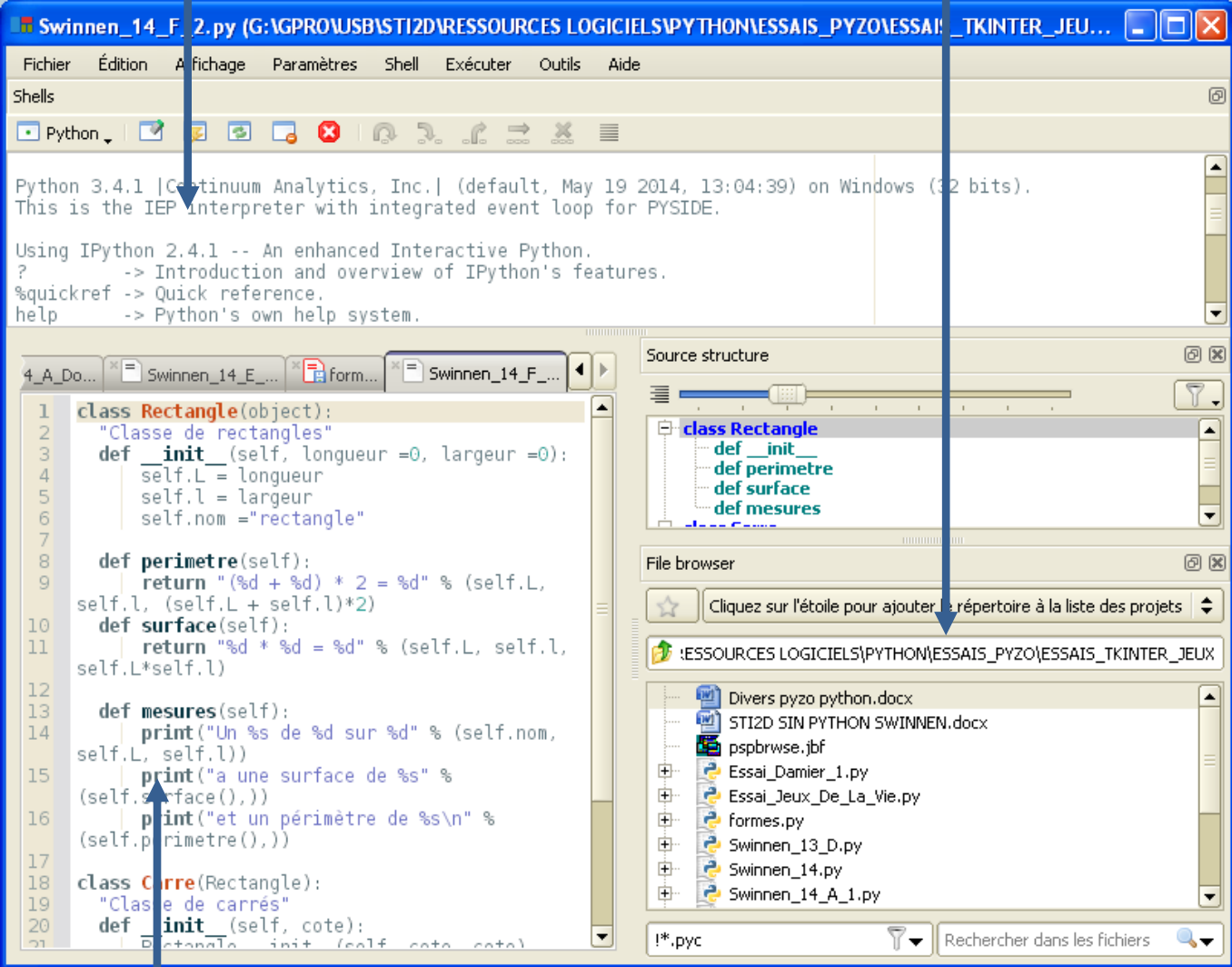
1.1 Utiliser un environnement autonome : pyzo

L'outil est très complet, et gratuit, pour l'utiliser le téléchargement c'est ici <http://www.pyzo.org/>

Lancement  La fenêtre de travail de PYZO :

Python interactif :

Le dossier de travail

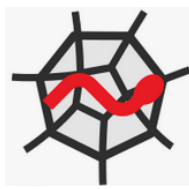


```

1 class Rectangle(object):
2     "Classe de rectangles"
3     def __init__(self, longueur =0, largeur =0):
4         self.L = longueur
5         self.l = largeur
6         self.nom = "rectangle"
7
8     def perimetre(self):
9         return "%d + %d) * 2 = %d" % (self.L,
10 self.l, (self.L + self.l)*2)
11     def surface(self):
12         return "%d * %d = %d" % (self.L, self.l,
13 self.L*self.l)
14
15     def mesures(self):
16         print("Un %s de %d sur %d" % (self.nom,
17 self.L, self.l))
18         print("a une surface de %s" %
19 (self.surface(),))
20         print("et un périmètre de %s\n" %
21 (self.perimetre(),))
22
23 class Carre(Rectangle):
24     "Classe de carrés"
25     def __init__(self, cote):
26         Rectangle.__init__(self, cote, cote)
  
```

Les fichiers de codes
CTRL + Entrée pour les exécuter

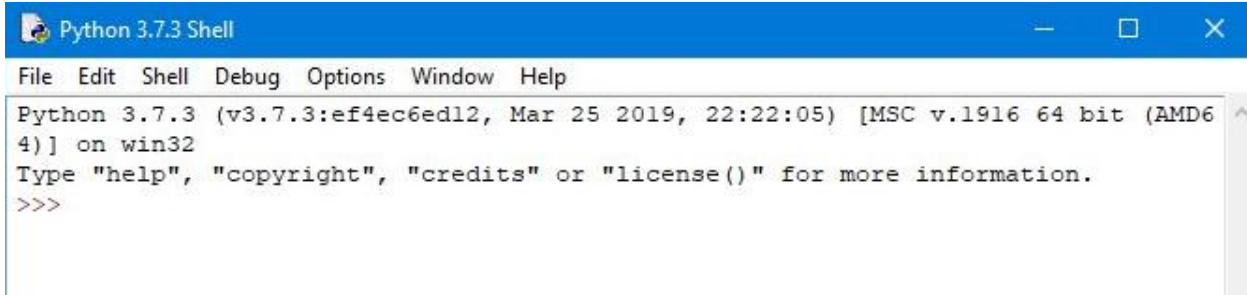




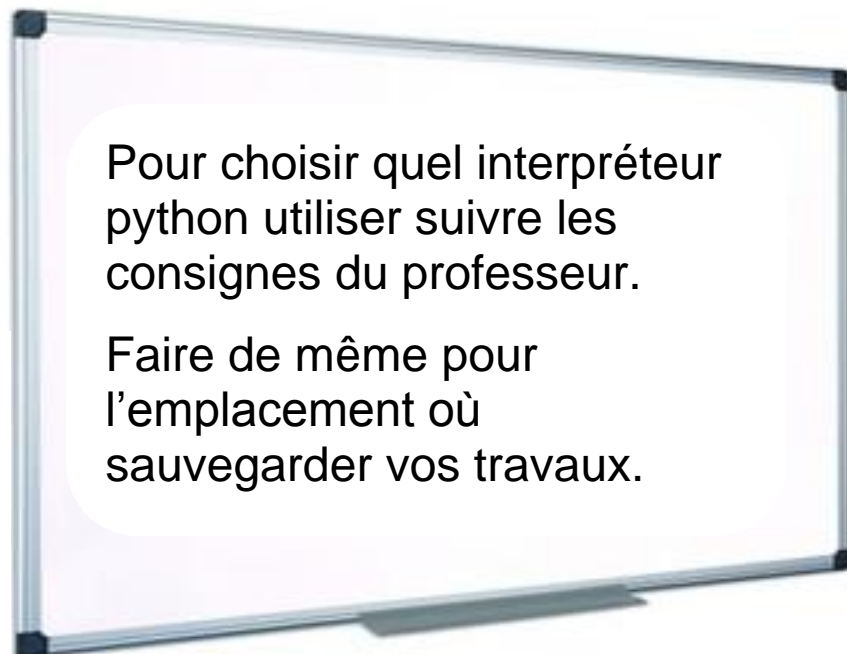
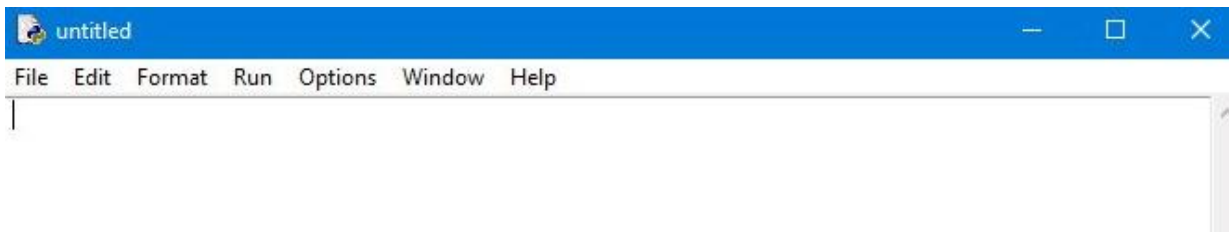
1.2 Utiliser IDLE python 3.x

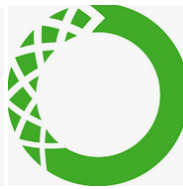
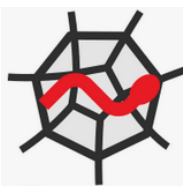


Le raccourci IDLE ouvre la console python, ici de la version 3.7.

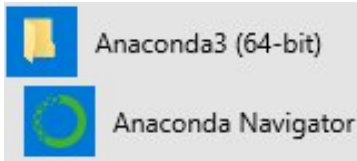


La commande File permet d'ouvrir un script existant ou bien d'en éditer un nouveau :

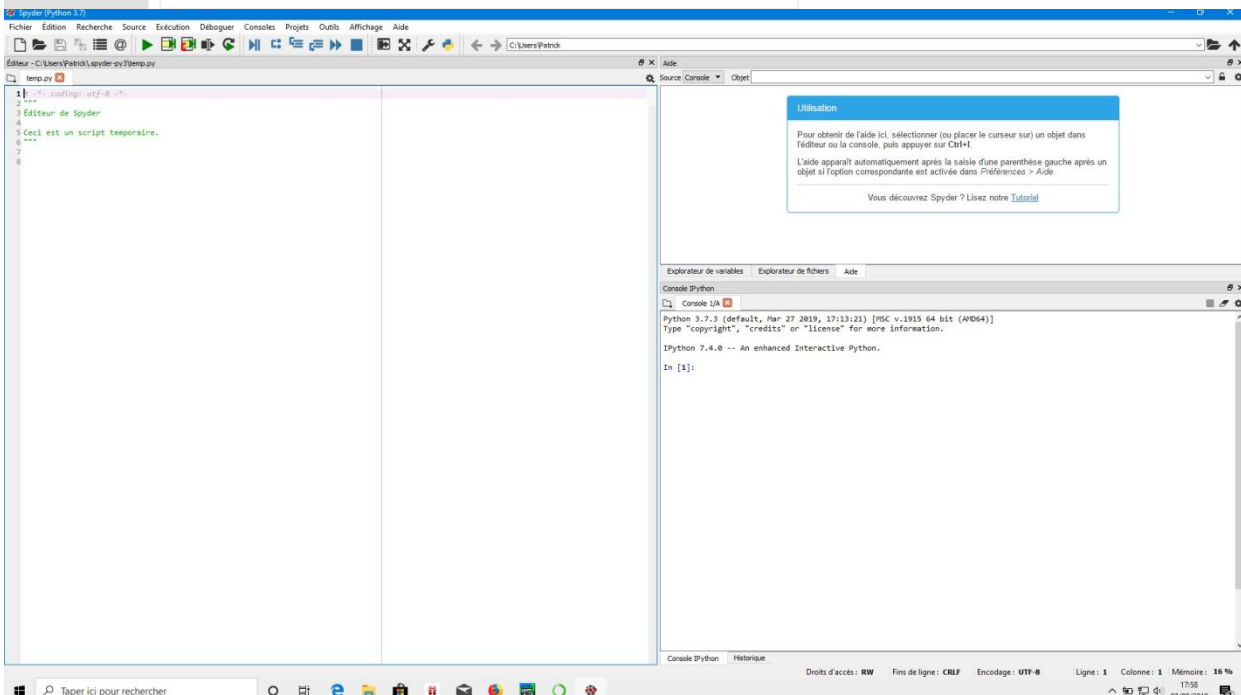
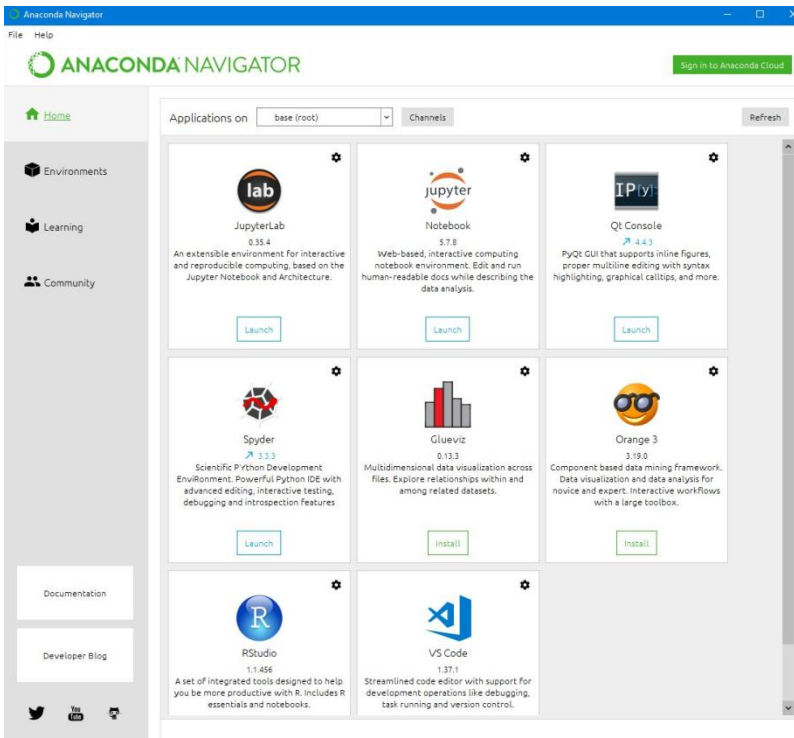


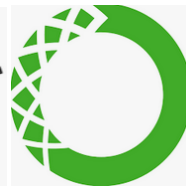
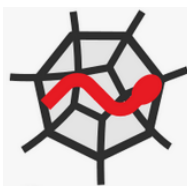


1.3 Utiliser anaconda



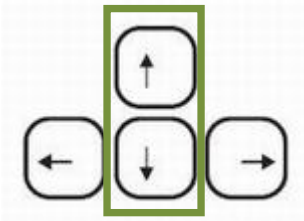
L'environnement Anaconda peut également être utilisé pour la console 'standard' lancer Spyder, pour les notebooks lancer Jupyter :



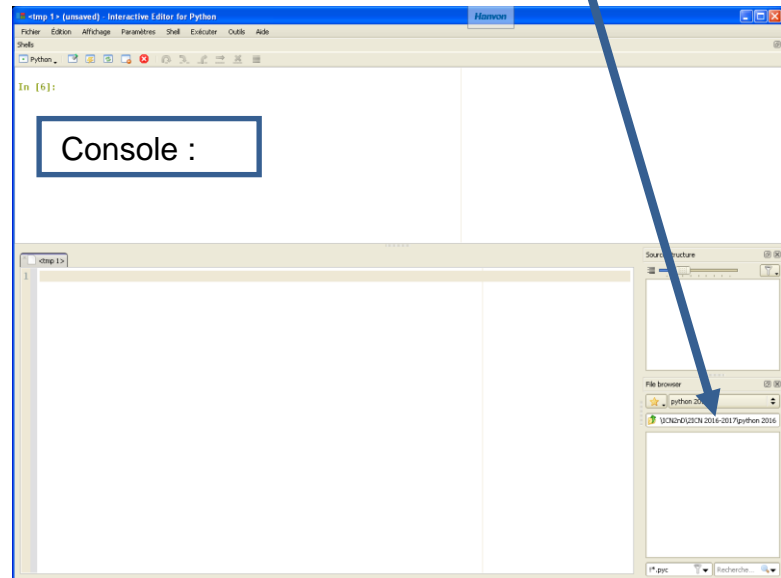


2 Premier pas : utilisation de la console

↪ Vérifier le bon positionnement du dossier de travail avant de commencer.



Les touches flèches du clavier permettent de rappeler les dernières commandes mémorisées dans la console.



↪ Premier exemple d'utilisation de la console :

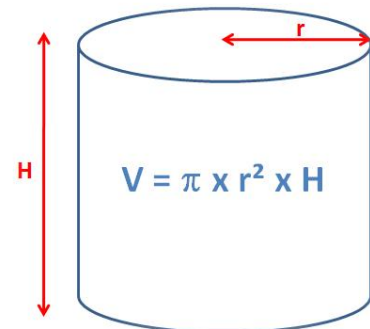
Nous avons un volume à remplir de forme cylindrique avec
 $r = 1.56$ m et $H = 3.78$ m

- Définir et initialiser les variables r , H , π

```
r,H,pi = 1.56,3.78,3.14159
```

- Calculer le volume.

```
r*r*pi*H
```



A vous de jouer quel est le résultat :

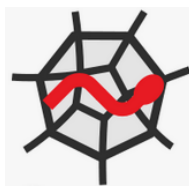
Python permet d'auto déclarer les variables. Le type est automatiquement affecté en fonction de l'initialisation qui est faite.

$r = 1.56 \Rightarrow r$ est réel

$toto = 2 \Rightarrow toto$ est entier

On peut également faire une affectation multiple en séparant les différentes affectations par des virgules.





2.1 La bibliothèque math

Nous pouvons utiliser les bibliothèques intégrées à python de plusieurs manières :

a) Import au cas par cas :

Demande de mise à disposition de la bibliothèque

```
import math
```

Appel des fonctions et variables présentes dans cette bibliothèque avec le nom complet et la syntaxe à point :

```
r*r*math.pi*H
```

b) Import global :

Mise à disposition de toute la bibliothèque dans la session

```
from math import *
```

Le calcul devient alors :

```
r*r*pi*H
```

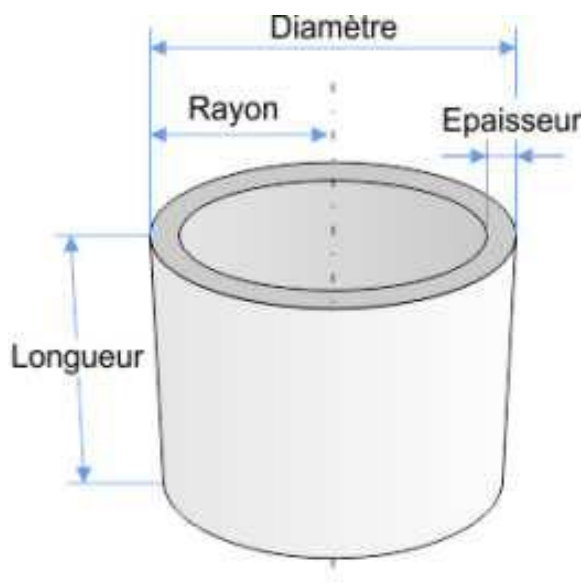
c) Liste du contenu d'une bibliothèque

La commande help permet de connaître le contenu disponible dans une bibliothèque

```
help(math)
```

2.2 Calculs avec la console

Essai Python 1.

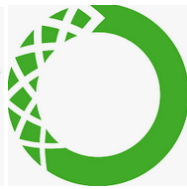
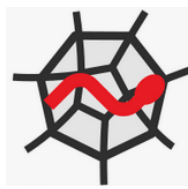


Écrire les formules ici :

Calcul du volume de la paroi

Calcul du volume intérieur





Longueur : L	Rayon : R	Diamètre : D	Épaisseur : E	Volume paroi	Volume intérieur
2.03	1.52		0.25		
7.58		3.47	0.35		

2.3 Quelques opérateurs



Essai Python 2. Comparer les opérateurs suivants avec la console

/ : 10 / 3

// : 10 // 3

% : 10 % 3

Quel est le résultat produit par chacun d'eux ?

2.4 Faire des tests, des comparaisons de variables

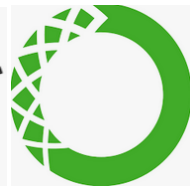
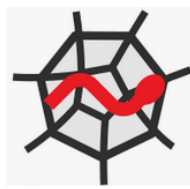
En programmation il faut fréquemment prendre des décisions. Il faut alors effectuer des tests entre plusieurs variables et agir en fonction du résultat. Les tests sont utilisés dans les instructions conditionnelles ou de sélection :

```
# Utilisation des tests simples
a=input("Entrez la valeur à tester : ")
a=int(a)
if a > 0 :
    print("La valeur est positive")
if a < 0 :
    print("La valeur est négative")
if a == 0 :
    print ("La valeur est nulle")
```

La structure de base est la suivante :

```
if METTRE LE TEST ICI :
    ACTION SI VRAI
else :
    ACTION SI FAUX
```





Les tests, comme toutes les structures algorithmiques, peuvent être imbriqués :

```


if METTRE LE 1er TEST ICI :
    ACTION SI 1er TEST VRAI
else :
    if METTRE LE 2eme TEST ICI :
        ACTION SI 2eme TEST VRAI
    else :
        ACTION SI 2eme TEST FAUX
    
```

Nous pouvons utiliser pour des chaînes de tests successifs une écriture imbriquée comme ceci :

```

# Utilisation des tests simples contractés
# avec elif ... else
a=input("Entrez la valeur à tester : ")
a=int(a)
if a > 0 :
    print("La valeur est positive")
elif a < 0 :
    print("La valeur est négative")
else :
    print ("La valeur est nulle")
    
```

 Essai Python 3. Coder les deux exemples.

 Essai Python 4. Quel est l'exemple le plus rapide i.e. qui effectue le moins de tests.

Nous avons utilisé dans notre exemple les opérateurs de comparaison :

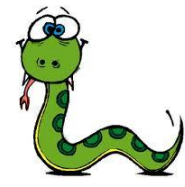
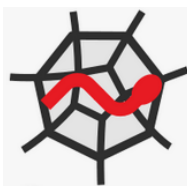
- Plus grand que >
- Plus petit que <
- est égal ==

Voilà la liste complète des opérateurs disponibles :

```

x == y # x est égal à y
x != y # y est différent de y
x > y # x est plus grand que y
x < y # x est plus petit que y
x >= y # x est plus grand ou égal à y
x <= y # x est plus petit ou égal à y
    
```





2.5 Un premier petit programme

Comment déterminer si une année est bissextile ou pas ?

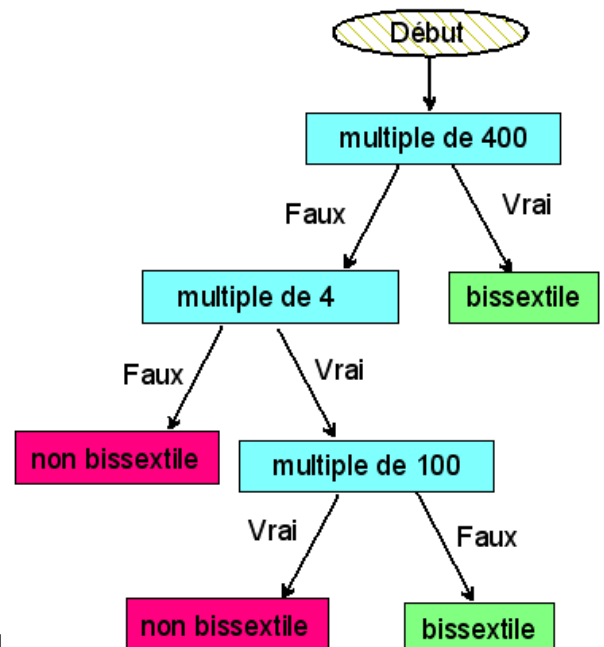
Une **année bissextile** est une **année** comportant 366 jours au lieu de 365 jours pour une **année non bissextile**. Le jour supplémentaire, le 29 février, est placé après le dernier jour de ce mois qui compte habituellement 28 jours dans le calendrier grégorien.

[Année bissextile — Wikipédia](https://fr.wikipedia.org/wiki/Année_bissextile)
https://fr.wikipedia.org/wiki/Année_bissextile

À propos de ce résultat • Commentaires

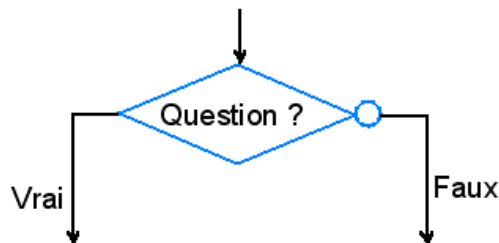
Le raisonnement est le suivant :

- ↪ Si l'année est divisible par 400
- ↪ Si l'année est divisible par 4 et non divisible par 100



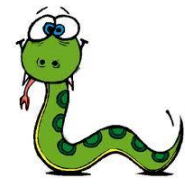
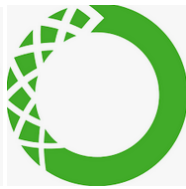
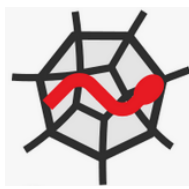
Le rectangle bleu représente un test c'est-à-dire une question à laquelle la réponse est soit oui (vrai) soit non (Faux). La représentation normalisée d'un test est donnée ci-dessous:

Essai Python 5. Représenter l'algorithme ci-dessus avec le test normalisé.



Astuce : vous mettez les trois flèches comme cela vous arrange pour obtenir le meilleur tracé d'algorithme possible.





Le programme en python :

```
# Programme testant si une année, saisie par l'utilisateur, est bissextile ou non
# Version longue identique à l'algorithme
#

# On attend que l'utilisateur fournisse l'année qu'il désire tester
annee = input("Saisissez une année : ")
# Risque d'erreur si l'utilisateur n'a pas saisi un nombre
annee = int(annee)

# Calcul pour répondre à la question année bissextile ?
# Pour faciliter la lecture l'algorithme n'est pas optimisé
# Les if ... sont complets

if annee % 400 == 0:
    bissextile = True
elif annee % 4 == 0:
    if annee % 100 != 0:
        bissextile = True
    else:
        bissextile = False
else :
    bissextile = False

if bissextile:
    print("L'année saisie est bissextile.")
else:
    print("L'année saisie n'est pas bissextile.")
```



Essai Python 6. Codez et tester ce programme.

2.6 Éléments de syntaxe : Précision sur les noms de variables

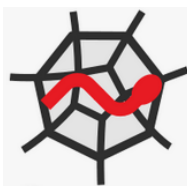
Les noms de variables doivent suivre les règles **obligatoires** suivantes :

- Un nom de variable est une séquence de lettres (a → z , A → Z) et de chiffres (0 → 9), qui doit toujours commencer par une lettre.
- Seules les lettres ordinaires sont autorisées. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que \$, #, @, etc. sont interdits, à l'exception du caractère _ (souligné).
- La casse est significative (les caractères majuscules et minuscules sont distingués).
Attention : Joseph, joseph, JOSEPH sont donc des variables différentes. Soyez attentifs !

Par convention on écrit les noms de variables en minuscules, les majuscules sont utilisées à l'intérieur du nom pour augmenter la lisibilité comme :

maVariable





Les noms de fonctions suivent les mêmes règles :

maFonctionSuperPratique

Il n'est également pas possible d'utiliser comme noms de variables les mots clés réservés du langage python :

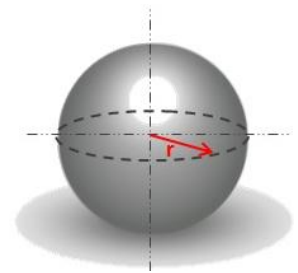
and	as	assert	break	class	continue	def
del	elif	else	except	False	finally	for
from	global	if	import	in	is	lambda
None	nonlocal	not	or	pass	raise	return
True	try	while	with	yield		

3 Plus loin avec le langage

3.1 Utilisation de fonctions

Illustrons l'utilisation de fonction avec le calcul du volume d'une sphère

$$V = \frac{4}{3} \pi R^3$$



Le script python est donné ci-dessous :

```

# Calcul du volume d'une sphère avec utilisation de fonctions

# Calcul de l'élevation à la puissance 3
def cube(n):
    return n**3

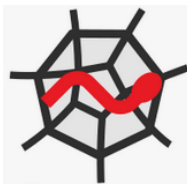
# Calcul du volume de la sphère
def volumeSphere(r):
    return 4 * 3.1416 * cube(r) / 3

# Programme principal
r = input('Entrez la valeur du rayon en m : ')
print('Le volume de cette sphère vaut', volumeSphere(float(r)), ' m3')
    
```

Annotations du code :

- Fonction n^3 : correspond à la fonction `cube(n)`.
- Fonction calcul du volume : correspond à la fonction `volumeSphere(r)`.
- Le programme principal : correspond au code de la dernière section.





Éléments de syntaxe : fonction

le ':' indique la fin d'une structure

Commentaire

```
# Calcul du volume de la sphère
def volumeSphere(r):
    return 4 * 3.1416 * cube(r) / 3
```

L'indentation indique que l'on est dans la structure indiquée par le caractère ':' au-dessus

La fonction dénommée volumeSphere reçoit la variable r en paramètre et renvoie avec l'instruction 'return' la valeur calculée. Elle utilise une autre fonction 'cube'

Essai Python 7. Codez et testez ce programme.

3.2 Un convertisseur distance / temps

Vous devez concevoir un logiciel d'aide à la navigation. L'objectif est de calculer la distance à parcourir en miles nautique et la durée du trajet en heure à partir de la donnée de la distance du trajet en km et de la vitesse du navire en nœuds.

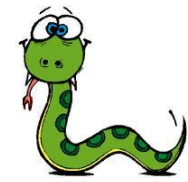
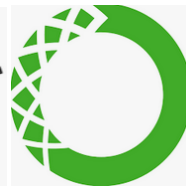
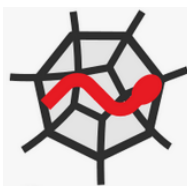
1 mile marin = 1852 m 1 nœud = 1 mile·h⁻¹

Essai Python 8. Codez et testez cet exemple.



Navire de service offshore transporteur de charges lourdes / de ravitaillement de plateformes pétrolières





L'Abeille Flandre

Remorqueur de haute mer, il sort par tous les temps secourir les navires en difficulté.

Évidemment la durée du trajet dépend aussi de l'état de la mer !



3.3 Le formatage des données en sortie méthode recommandée si python < 3.6

Dans l'exemple proposé dans le livre de Swinnen la chaîne ch est construite avec des champs réservés pour les données à formater. Ces champs sont indiqués avec les accolades { }

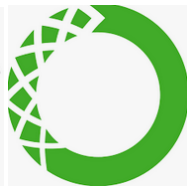
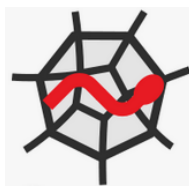
Ensuite la méthode .format est invoquée pour passer en argument les valeurs à formater :

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # Exemple de l'application de la méthode .format d'une chaîne
5  # Swinnen chapitre 10 p. 166
6
7  coul = "verte"
8  temp = 1.347 + 15.9
9  ch = "La couleur est {} et la température vaut {} °C"
10 print(ch.format(coul, temp))

```





Le champ peut contenir des indications de formatage :

```
12 ch = "La couleur est {} et la température vaut {:.2e} °C"
13 print(ch.format(coul, temp))
14
15 n = 789
16 txt = "Le nombre {:d} (décimal) vaut ${:04X} en hexadécimal et {:016b}b en binaire sur 16 bits"
17 print(txt.format(n,n,n))
```

Le résultat de cet exemple est donné ci-dessous :

```
In [10]: (executing lines 1 to 14 of "Swinnen_chapitre10_p166.py")
La couleur est verte et la température vaut 17.247 °C
La couleur est verte et la température vaut 1.72e+01 °C
Le nombre 789 (décimal) vaut $0315 en hexadécimal et 0000001100010101b en binaire sur 16 bits
```



Pour les descriptions des formats voir le paragraphe suivant.

3.4 f string formatage des données en sortie méthode recommandée si python >= 3.6

a) Le principe

Pour utiliser f-strings il suffit de mettre un f devant la chaîne de caractères et pour insérer la valeur d'une variable dans la chaîne il suffit de mettre la variable entre accolade. Si il n'y a pas de variable à substituer il n'est pas nécessaire de mettre le f devant.

```
>>> name = 'Paul'
>>> age = 23

>>> print(f'Votre nom est un {name} et vous avez {age} ans')
Votre nom est un Paul et vous avez 23 ans
```

f-strings prend également en compte les fonctions ce qui signifie que les fonctions qui se trouvent dans la chaîne de caractères sont exécutées et le résultat est affiché.

b) Formater les données

f-strings implémente également une manière de mettre en forme les nombres.

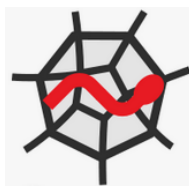
Voici la syntaxe pour utiliser le formatage (voir [Format Specification Mini-Language](#)):

```
f ' <texte> { <expression/variable> : <format> } <texte> ... '
```

Le format se trouve sous cette forme pour les nombres à virgule:

```
[alignement][signe][largeur][groupage][.précision][type]
```





- alignement : *détermine ou le nombre est aligné dans sa zone*
 - “>” aligne à droite
 - “<” aligne à gauche
 - “^” centré
 - “=” aligne le signe à gauche et le nombre à droite
- signe : *détermine l’affichage du signe*
 - “+” indique que le signe + doit être affiché ainsi que le -
 - “-” indique que le signe - doit être affiché (par défaut)
 - “ ” n’affiche pas le + mais insère un espace à la place
- largeur : *détermine la place qui doit être réservée pour l’affichage du nombre*
- groupage : *détermine le symbole de séparation tous les 3 chiffres*
 - “_”
 - “,”
- précision : *détermine le nombre de chiffres après la virgule*
- type : *détermine le mode d’affichage*
 - “e” ou “E” notation scientifique
 - “f” ou “F” affichage classique

Exemple :

```
>>> nombre = 357568.12312
>>> nombre2 = 568.568768
>>> nombre3 = -34.3432
>>> nombre4 = 23
>>> print(f'{nombre : >+20_.4f} {nombre2 : >+20_.4f}')
>>> print(f'{nombre3 : >+20_.4f} {nombre4 : >+20_.4f}')

+357_568.1231          +568.5688
-34.3432              +23.0000
```

C’est très pratique pour faire des tableaux de nombre.

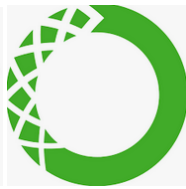
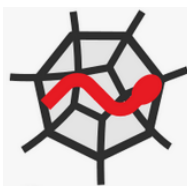
Il existe aussi des option de formattage pour les entiers il suffit simplement de mettre une lettre pour le format:

- “b” : affiche en binaire
- “c” : affiche le caractère correspondant au code unicode
- “d” : affiche en decimal (par défaut)
- “o” : affiche en octal
- “x” ou “X” : affiche en hexa

Exemple :

```
>>> chmod = 0o644
>>> f'{chmod:016b}'
'0000110100100'
```



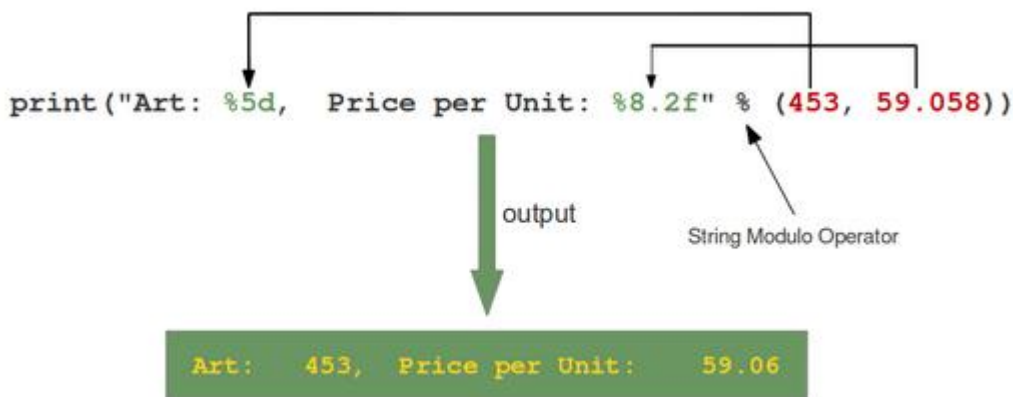


3.5 Le formatage des données en sortie 'méthode ancienne'

Une solution de l'exercice 2.2 donne les sorties suivantes sur la console :

```
Entrez la longueur du trajet en km : 2500
Entrez la vitesse moyenne de croisière prévue : 12.5
La distance en miles nautique est de 1349.8920086393089 miles
La durée du trajet prévue est de 107.9913606911447 H
A la vitesse moyenne de 12.5 miles / H
```

Nous observons que les résultats sont affichés avec un grand nombre de décimales. Il faut pour maîtriser le format de sortie utiliser des commandes de formatage. Le principe est expliqué sur le schéma ci-dessous¹ :



L'instruction print contient toutes les chaînes de caractères à afficher avec pour chacune des valeurs numériques un champ commençant par le caractère % et suivi d'un code indiquant la nature du format désiré. Les valeurs numériques sont ensuite écrites après le '%'.

Appliquons ce principe pour notre calcul de distance :

```
# Calcul de la distance en miles nautiques
distanceEnMiles = distance / 1.852
print("La distance en miles nautique est de %8.2f miles" % (distanceEnMiles))
```

Les résultats sont beaucoup plus présentables :

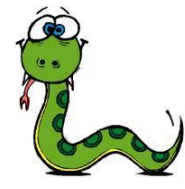
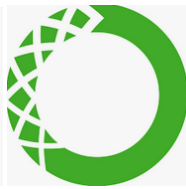
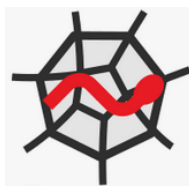
```
Entrez la longueur du trajet en km : 3500
Entrez la vitesse moyenne de croisière prévue : 14.5
La distance en miles nautique est de 1889.85 miles
La durée du trajet prévue est de 130.33 H
A la vitesse moyenne de 14.5 miles / H
```



Essai Python 9. Modifiez votre programme précédent pour y intégrer le formatage.



¹ Site http://www.python-course.eu/python3_formatted_output.php consulté le 27 février 2017.



Explication du format utilisé pour les nombres réels :

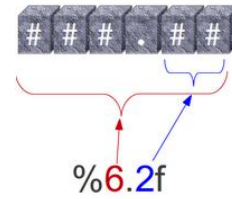
% : formatage.

6 : nombre total de digit utilisé en comptant la virgule.

2 : précision i.e. nombre de chiffres après la virgule.

Les différents formats disponibles avec quelques exemples :

```
>>> print("%#5X"% (47))
  0X2F
>>> print("%5X"% (47))
  2F
>>> print("%#5.4X"% (47))
0X002F
>>> print("%#5o"% (25))
  0o31
>>> print("%+d"% (42))
+42
>>> print("% d"% (42))
 42
```

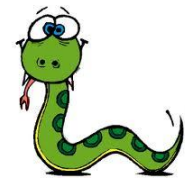
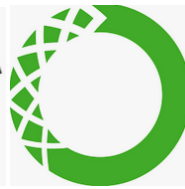
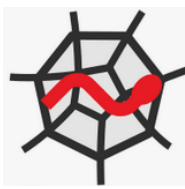


```
>>> print("%10.3e"% (356.08977))
 3.561e+02
>>> print("%10.3E"% (356.08977))
 3.561E+02
>>> print("%10o"% (25))
      31
>>> print("%10.3o"% (25))
      031
>>> print("%10.5o"% (25))
      00031
>>> print("%5x"% (47))
      2f
>>> print("%5.4x"% (47))
      002f
>>> print("%5.4X"% (47))
      002F
```

Voir la documentation officielle python :

<https://docs.python.org/fr/3.5/library/stdtypes.html#str.format>



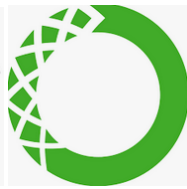
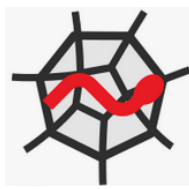


Conversion	Signification
'd'	Entier décimal signé.
'i'	Entier décimal signé.
'o'	Valeur octale signée.
'u'	Type obsolète - identique à 'd'.
'x'	Hexadécimal signé (en minuscules).
'X'	Hexadécimal signé (capitales).
'e'	Format exponentiel pour un <i>float</i> (minuscule).
'E'	Format exponentiel pour un <i>float</i> (en capitales).
'f'	Format décimal pour un <i>float</i> .
'F'	Format décimal pour un <i>float</i> .
'g'	Format <i>float</i> . Utilise le format exponentiel minuscules si l'exposant est inférieur à -4 ou pas plus petit que la précision, sinon le format décimal.
'G'	Format <i>float</i> . Utilise le format exponentiel en capitales si l'exposant est inférieur à -4 ou pas plus petit que la précision, sinon le format décimal.
'c'	Un seul caractère (accepte des entiers ou une chaîne d'un seul caractère).
'r'	String (convertit n'importe quel objet Python avec <code>repr()</code>).
's'	String (convertit n'importe quel objet Python avec <code>str()</code>).
'a'	String (convertit n'importe quel objet Python en utilisant <code>ascii()</code>).
'%'	Aucun argument n'est converti, donne un caractère de '%' dans le résultat.

Quelques caractères particuliers disponibles pour indiquer dans quelle base un résultat est affiché ou forcer l'affichage d'un signe avec quelques exemples :

Option	Signification
'#'	La conversion utilisera la « forme alternative » (définie ci-dessous).
'0'	Les valeurs numériques converties seront complétée de zéros.
'-'	La valeur convertie est ajustée à gauche (remplace la conversion '0' si les deux sont données).
' '	(un espace) Un espace doit être laissé avant un nombre positif (ou chaîne vide) produite par la conversion d'une valeur signée.
'+'	Un caractère de signe ('+' ou '-') précède la valeur convertie (remplace le marqueur « espace »).





3.6 Quelques possibilités de mise en forme

- Choix du séparateur quand on affiche plusieurs chaînes elles sont séparées par un caractère espace. Il est possible de choisir une chaîne particulière pour réaliser cette séparation :

```
In [1]: print("Bonjour","le","monde", sep=" - ")
Bonjour - le - monde
```

- Si on ne veut pas aller à la ligne après une instruction print() il suffit de le préciser avec l'option end= " "

```
# Impression avec formatage et affichage amélioré
n=0
while n<4:
    print("Mon chiffre = %2d" % (n)," ",sep=" ++ ", end = " ")
    n=n+1
```

Mon chiffre = 0 ++ Mon chiffre = 1 ++ Mon chiffre = 2 ++ Mon chiffre = 3 ++

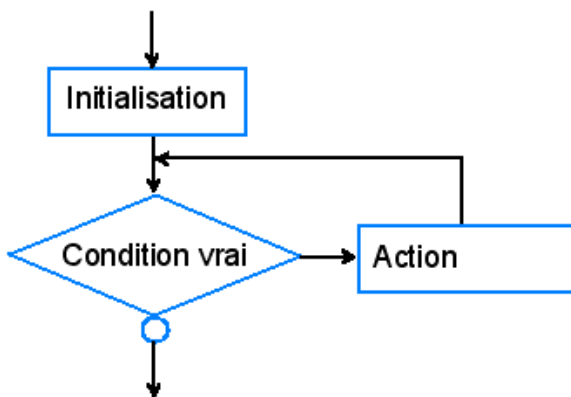


Essai Python 10.

Codez et testez cet exemple.

3.7 La structure tant ... que while

La structure tant que réalise un traitement tant qu'une condition est réalisée. L'organigramme est le suivant :



A NOTER :

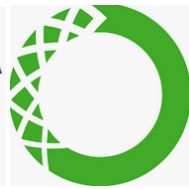
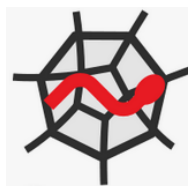
La condition est testée en premier donc le bloc action n'est jamais exécuté si la condition est fausse dès le départ.

Si la condition est vraie alors le bloc action est exécuté, il faut faire évoluer la condition si on veut sortir de la boucle.

Calcul de la suite de Fibonacci

0, 1, 1, 2, 3, 5, 8, 13, 21, ... Lecture: La **suite de Fibonacci** F_n est la succession de tous les nombres de $n = 1$ à l'infini telle que les deux premiers sont égaux à 1 et les suivants se calculent comme la somme des deux précédents. Par convention, on pose que le nombre de **Fibonacci** de rang 0 est égal à 0.





Un exemple de réalisation :

```
# Calcul de la suite de Fibonacci


# initialisation
a, b, c = 1, 1, 1

# Entrée de la valeur finale
fin=input("Nombre de valeurs : ")
fin=int(fin)
fin = fin - 1

# Ecretage de la valeur si elle est trop grande
if fin>20:
    fin=20

# Affichage de la suite
print("0 1 ", end="")
while c < fin:
    print(b, end=" ")
    a, b, c = b, a+b, c+1

    Nombre de valeurs : 12
    0 1 1 2 3 5 8 13 21 34 55 89
```

 Essai Python 11. Codez et testez cet exemple


3.8 Les répétitions avec des boucles

Programmer une boucle en informatique c'est faire une action répétitive.

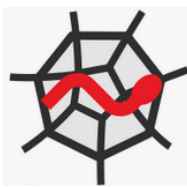
```
# Première boucle

chaine = "Bonjour le monde"
for lettre in chaine:
    print(lettre, " ",end="")

B o n j o u r   l e   m o n d e
```

 Essai Python 12. Codez et testez cet exemple





On peut choisir le nombre d'exécution de la boucle, on parle du nombre d'itérations, en utilisant une variable dédiée à compter le nombre de passage dans la boucle. Dans l'exemple ci-dessous c'est la variable `i`.

On peut contrôler précisément le nombre d'itérations avec l'intervalle `range(1,8)`, noter que cet intervalle déroule de 1 jusqu'à 7.

Deuxième boucle


```
for i in range(1,8):
    print("Le carré de %3d vaut %3d" %(i,i**2))
```

```
Le carré de 1 vaut 1
Le carré de 2 vaut 4
Le carré de 3 vaut 9
Le carré de 4 vaut 16
Le carré de 5 vaut 25
Le carré de 6 vaut 36
Le carré de 7 vaut 49
```

 Essai Python 13. Codez et testez cet exemple

 Essai Python 14. Codez l'affichage de la table de multiplication de 9

3.9 Retour sur les fonctions avec plusieurs paramètres

 Essai Python 15. En vous inspirant de l'exemple ci-dessous tester une fonction qui affiche la table de multiplication :

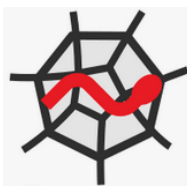
```
# Affichage des tables de multiplications
# Utilisation d'une fonction avec plusieurs paramètres

def tableMulti(base, debut, fin):
    print('Fragment de la table de multiplication par',base, ' :')
    for i in range(debut,fin+1) :
        print("%3d x %3d = %4d" % (i, base, base * i))
##
tableMulti(8, 13, 17)
```

Fragment de la table de multiplication par 8 :

```
13 x 8 = 104
14 x 8 = 112
15 x 8 = 120
16 x 8 = 128
17 x 8 = 136
```





3.10 Les listes et les tables : conjuguons un verbe

La conjugaison est un bon exemple d'utilisation de tables et de manipulation de chaînes de caractères. Voilà un premier exemple :

Utilisation des tables pour réaliser une conjugaison

```
pronom = ["je", "tu", "il", "nous", "vous", "ils"]
radical = "man"
terminaison = ["ge", "ges", "ge", "geons", "gez", "gent"]

print("Conjugaison du verbe ", radical, "ger", sep='')
for i in range(0,6):
    print(pronom[i], " ", radical, terminaison[i], sep='')
```

Conjugaison du verbe manger
je mange
tu manges
il mange
nous mangeons
vous mangez
ils mangent



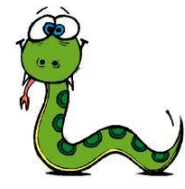
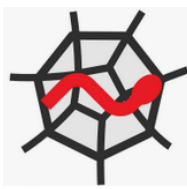
Essai Python 16. Codez et testez cet exemple

C'est bien mais nous voulons modifier notre code programme pour pouvoir traiter tous les verbes en -ger comme patauger, dégager, déjuger, mélanger Il faut donc isoler automatiquement le radical de l'écriture du verbe à l'infinitif.

```
1 # Utilisation des tables pour réaliser une conjugaison
2
3 # Saisie du verbe
4 verbe = input("Entrez votre verbe en -ger svp : ")
5 print()
6
7 # Isolation du radical
8 longueurVerbe = len(verbe)
9 radical= verbe[0:longueurVerbe-3]
10
11 # Affichage de la conjugaison du verbe
12 pronom = ["je", "tu", "il", "nous", "vous", "ils"]
13 terminaison = ["ge", "ges", "ge", "geons", "gez", "gent"]
14
15 print("Conjugaison du verbe ", radical, "ger", "\n", sep='')
16 for i in range(0,6):
17     print(pronom[i], " ", radical, terminaison[i], sep='')
18
```

Essai Python 17. Codez et testez cet exemple





Nous voyons dans cet exemple quelques traitements sur les listes. Pour rappel une chaîne de caractère est en fait une liste de caractères.


```

1 # Exemple de traitement sur des chaînes
2
3 verbe = "patauger"
4 print("verbe = ",verbe)           verbe = patauger
5 print("verbe[0] = ",verbe[0])     verbe[0] = p
6 print("longueur du verbe = ",len(verbe),"\n")  longueur du verbe = 8
7
8 extraitDuVerbe = verbe[2:5]       verbe[2:5] = tau
9 print("verbe[2:5] = ",extraitDuVerbe,"\n")
10
11 pronom = ["je","tu","il","nous","vous","ils"]
12 print("pronom[5] = ",pronom[5])   pronom[5] = ils
13 print("pronom[3][0] = ",pronom[3][0],"\n")  pronom[3][0] = n

```

 Essai Python 18. Répondez aux questions ci-dessous :

- ↳ Comment sélectionner 'il' de la liste pronom :
- ↳ Comment sélectionner 'tauger' de la variable verbe :
- ↳ Expliquez la ligne numéro 9 de l'exemple 17.

 Essai Python 19. Proposer et tester une amélioration qui n'affiche la conjugaison que si le verbe se termine par -ger.

Entrez votre verbe en -ger svp : déménager

Conjugaison du verbe déménager

```

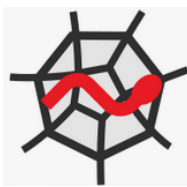
je déménage
tu déménages
il déménage
nous déménageons
vous déménagez
ils déménagent

```

Entrez votre verbe en -ger svp : écrire

Le verbe écrire n'est pas en -ger





4 Générer du hasard

Les programmes informatiques réalisent les calculs toujours de la même manière, ils sont donc prévisibles. La difficulté pour un générateur de nombre aléatoire est de le réaliser pour qu'il soit non-déterministe. Ceci est particulièrement vrai pour des applications comme les jeux par exemple. Si je relance plusieurs fois le jeu je ne dois pas avoir toujours le même comportement du joueur artificiel.

Étudions le hasard avec python².



4.1 Génération de hasard avec des nombres : fonctions seed(), random(), randint()

```
# Essai des valeurs aléatoires avec python

from random import *

# choix du germe de la séquence pseudo-aléatoire
# valeur 666, python 3
seed(666,3)

# La fonction random renvoi un nombre flottant compris entre 0 et 1
print("Tirage d'un nombre éléatoire")
print(random(),"\n")

# 100 lancer de dés
# La fonction randint(a,b) renvoi un nombre compris entre a et b
print("100 lançés de dés")
for x in range(100):
    print(randint(1,6), end=" ")
```




Résultat :

```
Tirage d'un nombre éléatoire
0.45611964897696833
```

```
100 lançés de dés
```

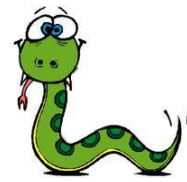
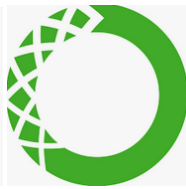
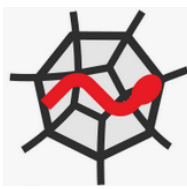
```
4 4 3 5 1 5 5 6 3 1 1 2 4 6 4 5 2 1 1 3 1 3 6 2 2 3 6 5 5 4 3 6 3 5 1 3
2 5 1 2 6 1 5 1 1 6 1 2 1 3 2 3 4 4 2 4 1 4 4 3 5 3 4 1 6 4 5 3 4 4 2 1
4 6 6 5 2 4 4 4 4 3 5 4 4 6 4 3 2 3 2 5 3 1 5 6 2 5 5 3
```

 Essai Python 20. Programmer le tirage de 3 dés à six faces, on affichera le tirage de chacun des dés et la somme totale.

```
Exemple de résultats :      Tirage numéro 0 valeur 2
                             Tirage numéro 1 valeur 3
                             Tirage numéro 2 valeur 4
                             La somme est égale à 9
```



² Site Python3 objectif jeux <https://www.apprendre-en-ligne.net/pj/hasard/index.html> consulté le 2 mars 2017.



4.2 Génération du hasard avec des listes : fonctions choice(), shuffle(), sample()

Avec des listes python permet également de faire des sélections en utilisant le hasard.

```
# La hasard avec des listes
```

```
from random import *
```

```
# Définition de la liste
```

```
liste = ['a','b','c','d','e','f','g','h','i','j','k','l']  
print ("Liste de départ : ",liste)
```

```
# La fonction choice() tire au sort un élément de la liste
```

```
print("Cinq éléments tirés au hasard : ", end='')  
for i in range(5):  
    print(choice(liste),end=' ')  
print("\n")
```

```
# On mélange la liste avec la fonction shuffle()
```

```
shuffle(liste)  
print ("Liste mélangée : ",liste,end='\n')  
print("\n")
```

```
# On tire au sort k éléments de la liste
```

```
# avec la fonction sample(liste,k)
```

```
print("Cinq éléments tirés au hasard avec sample(liste,5)")  
listeDeCinq = sample(liste,5)  
print(" => La sous liste tirée au sort : ",listeDeCinq)
```

```
i=0
```

```
print(" => Valeur des éléments : ",end='')  
for i in range(0,5):  
    print(listeDeCinq[i],end=' ')
```

```
print("\n")
```

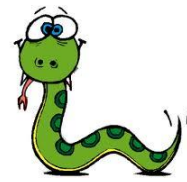
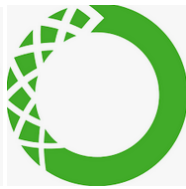
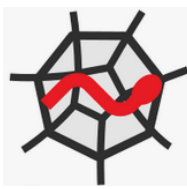
Le résultat :

```
Liste de départ : ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l']  
Cinq éléments tirés au hasard : d f j a h
```

```
Liste mélangée : ['g', 'd', 'f', 'e', 'a', 'j', 'b', 'h', 'i', 'l', 'k', 'c']
```

```
Cinq éléments tirés au hasard avec sample(liste,5)  
=> La sous liste tirée au sort : ['h', 'k', 'c', 'e', 'j']  
=> Valeur des éléments : h k c e j
```





4.3 Les dés sont pipés synthèse avec une liste et la fonction choice()

On souhaite simuler un dé pipé avec les probabilités suivantes :

1 : 10% ; 2,3,4,5 : 15% ; 6 : 30%

Pour coder le fonctionnement de ce dé on utilise une liste :

```
dePipe = [1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,6,6,6]
```

Voilà le code à compléter :




```
# La hasard avec des listes dé pipé
from random import *

# Définition des valeurs du dé pipé
dePipe = [1,1,2, à compléter ]

# Un tirage du dé
print("Tirage du dé pipé : ",choice(dePipe))
```

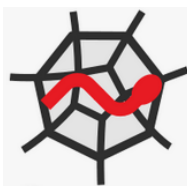
 Essai Python 21. Compléter et tester le code du dé pipé.

 Essai Python 22. Proposer et tester une méthode permettant de vérifier le dé pipé.

Exemple de résultats :

```
Entrez le nombre de tirage : 10000
Nombre de tirages : 1000
Nombre de 1 : 97 soit 0.10 au lieu de 0.10
Nombre de 2 : 149 soit 0.15 au lieu de 0.15
Nombre de 3 : 143 soit 0.14 au lieu de 0.15
Nombre de 4 : 158 soit 0.16 au lieu de 0.15
Nombre de 5 : 157 soit 0.16 au lieu de 0.15
Nombre de 6 : 296 soit 0.30 au lieu de 0.30
```





4.4 Les dés sont pipés synthèse avec random()

Si le dé est pipé mais avec des valeurs non entières, ou non multiples les unes des autres nous utilisons la synthèse avec la fonction random. Réalisons le codage d'un dé qui a les probabilités d'apparition des faces conforme à la liste ci-dessous :

- 1 apparaît dans 12.52% des cas
- 2 apparaît dans 13.09% des cas
- 3 apparaît dans 21.57% des cas
- 4 apparaît dans 19.87% des cas
- 5 apparaît dans 11.23% des cas
- 6 apparaît dans 21.72% des cas

```
# Dé pipé simulation avec la fonction random
```

```
from random import *
```

```
# Tirage de la valeur  
rnd = random()
```

```
# Affichage du résultat  
print("Le résultat du tirage est : ",end='')  
if rnd < 0.1252:  
    print(1)  
elif rnd < 0.2561:  
    print(2)  
elif rnd < 0.4718:  
    print(3)  
elif rnd < 0.6705:  
    print(4)  
elif rnd < 0.7828:  
    print(5)  
else:  
    print(6)
```



Essai Python 23.

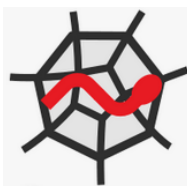
Compléter et tester le code du dé pipé.



Essai Python 24.

Proposer et tester une méthode permettant de vérifier le dé pipé.

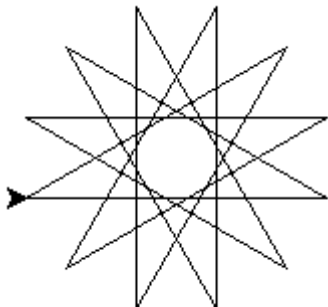




5 S'amuser avec le module tortue

5.1 Une première étoile

Comment réaliser ce dessin :



```
# S'amuser avec la 'tortue'
from turtle import *

reset()

a = 0
while a <12:
    a = a +1
    forward(150)
    left(150)
```

Le problème c'est qu'il faut fermer la fenêtre automatiquement, l'environnement python sinon pose problème sous windows. Nous allons ajouter deux instructions pour fermer automatiquement après 4 secondes par exemple³.

```
# S'amuser avec la 'tortue'
from turtle import *
from time import *

reset()

a = 0
while a <12:
    a = a +1
    forward(150)
    left(150)

# On admire notre oeuvre pendant 4 secondes
# et on ferme automatiquement la fenêtre
sleep(4)
bye()
```

Helge von Koch



1870 - 1994

Mathématicien suédois

Le flocon de von Koch
1904

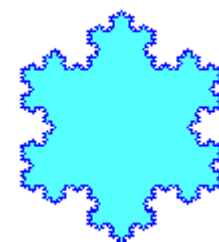


Vous pouvez tracer des formes complexes.

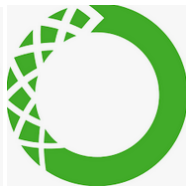
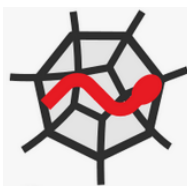


Essai Python 25.

Tester le script Web flocon von-Koch.py



³ Voir les exemples ici https://www.tutorialspoint.com/python/time_sleep.htm consulté le 2 mars.

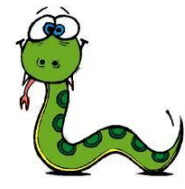
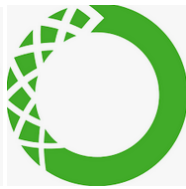
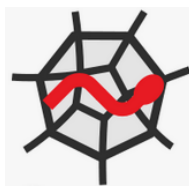


5.2 La liste d'une partie des commandes 'tortue' disponibles

Fonction	Effet
<code>fd(n)</code>	avance de <code>n</code>
<code>bk(n)</code>	recule de <code>n</code>
<code>rt(n)</code>	tourne à droite de <code>n</code> degrés
<code>lt(n)</code>	tourne à gauche de <code>n</code> degrés
<code>clear()</code>	efface l'écran
<code>penup()</code>	lève le stylo
<code>pendown()</code>	baisse le stylo
<code>reset()</code>	efface l'écran, remet la tortue au centre et réinitialise ses paramètres
<code>showturtle()</code>	montre la tortue
<code>hideturtle()</code>	cache la tortue
<code>speed(n)</code>	Change la vitesse de 1(lent) à 10 (rapide). La valeur spéciale 0 est la plus rapide.
<code>tracer(n,d)</code>	
<code>update()</code>	Force l'affichage des graphismes en attente
<code>bye()</code>	Referme la fenêtre
<code>setup(w,h)</code>	Ouvre une fenêtre de taille <code>w</code> <code>x</code> <code>h</code>

 [Une liste détaillée ici](#)





6 Réglages divers

6.1 Réglage du mode d'interprétation des caractères accentués norme UNICODE

Pour gérer les caractères accentués la directive coding en première ou deuxième ligne réglée sur la norme unicode UTF8

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

6.2 Accès à des caractères spéciaux

```
# Exemple d'accès à des caractères non présents sur le clavier
```

```
# ord(ch) renvoie le num code du caractère
```

```
# chr(num) renvoi le glyphe du caractère demandé si il existe
```

```
print("Num code du caractère 'A' :",ord('A'))
```

```
# Accès à des caractères spéciaux alphabet grec en minuscules
```

```
s = "" # chaîne vide
```

```
i = 945 # premier code
```

```
while i <= 969: # dernier code
```

```
    s += chr(i)
```

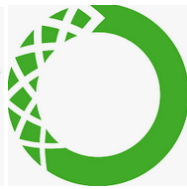
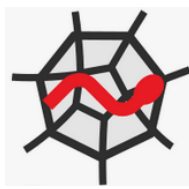
```
    i = i + 1
```

```
print("Alphabet grec (minuscule) : ", s)
```

```
Num code du caractère 'A' : 65
```

```
Alphabet grec (minuscule) : αβγδεζηθικλμνξοπρστυφχψω
```





7 Quelques points avancés

7.1 Les logarithmes

Attention il y a quelques faux amis avec les fonctions logarithmiques le $\ln(x)$ en python est $\log(x)$ **!! ET CE N'EST PAS LE LOGARITHME DECIMAL**

```
from math import *

## LES LOGARITHMES
# logarithme naturel (base e) ln(a) : log(a) !! ATTENTION
# logarithme base b : log(a,b)
# logarithme base 2 : log2(a)
# logarithme base 10 : log10(a)
a = 12
print("Logarithme néperien de a Ln(a) :",log(a))
print("Logarithme décimal de a log(a):",log10(a))
print("Logarithme base 2 de a :",log2(a))
print("Logarithme base 4 de a :",log(a,4))
```

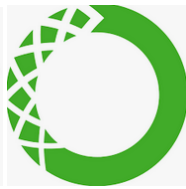
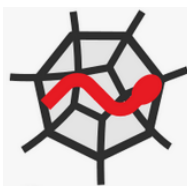
```
Logarithme néperien de a Ln(a) : 2.4849066497880004
Logarithme décimal de a log(a): 1.0791812460476249
Logarithme base 2 de a : 3.584962500721156
Logarithme base 4 de a : 1.7924812503605783
Valeur a : 1.7924
Arrondi inférieur de a: 1
Arrondi supérieur de a: 2
```

7.2 Mesure des temps d'exécution du programme

a) Pour python < 3.8 la librairie time

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 import time
5
6 # Mesure de la durée : début de la mesure
7 start_time = time.clock()
8
9 # Tâche à réaliser et à chronométrer
10 #
11 #
12
13
14
15 # Mesure de la durée : fin de la mesure
16 end_time = time.clock()
17
18 # Affichage de la durée de traitement
19 print("Le temps d'exécution : %10f secondes" % ( end_time - start_time ))
```

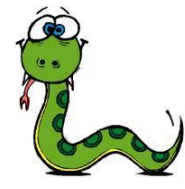
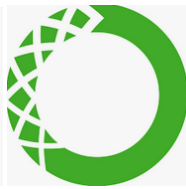
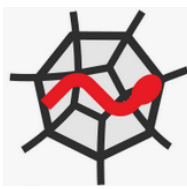




b) Pour python > = 3.8 la librairie process_time

```
8 from time import process_time
9
10 print()
11 print("CODE DE TEST")
12
13 # assigning n = 50
14 n = 50
15
16 # Start the stopwatch / counter
17 t1_start = process_time()
18
19 for i in range(n):
20     print(i, end = ' ')
21     for j in range(1000):
22         a = j**j
23 print()
24
25 # Stop the stopwatch / counter
26 t1_stop = process_time()
27
28 print()
29 print("BILAN")
30 print(f"Dates fin:{t1_stop:8.5f} S   Début:{t1_start:8.5f} S")
31 print(f"Temps passé par le programme en millisecondes: {(t1_stop-t1_start)*1000:8.5f} mS")
```





8 Configuration de pyzo

L'environnement de développement Pyzo permet de travailler avec plusieurs versions de pyzo. Il est possible également de configurer celle qui est en cours d'utilisation.

8.1 Configuration avec l'outil Pyzo

The screenshot shows the Pyzo IDE interface. The main window displays a Python shell with the following text:

```
Python 3.4.1 |Continuum Analytics, Inc.| (default, May 19 2014, 13:04:39) on Windows (32 bits).
This is the IEP interpreter with integrated event loop for PYSIDE.

Using IPython 2.4
? -> Intro
%quickref -> Quickref
help -> Python help
object? -> Detailed help for object
```

A purple arrow points to the 'Shell' menu. The menu is open, showing the following options:

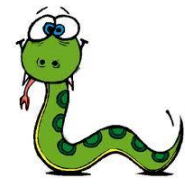
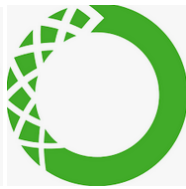
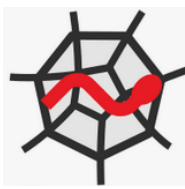
- Effacer l'écran (Ctrl+L)
- Interrompre (Ctrl+I)
- Redémarrer (Ctrl+K)
- Terminer (Ctrl+Maj+K)
- Fermer (Alt+K)
- Effacer tous les points d'arrêt
- Postmortem : déboguer à partir de la dernière pile des appels
- Débugage pas à pas principal (next)
- Débugage pas à pas entrant (step)
- Débugage pas à pas sortant
- Débugage : exécuter jusqu'à prochain point d'arrêt
- Arrêter le débogage
- Configuration des shells
- Create shell 1: (Python)
- Create shell 2: (Shell Python 3.4.0)

A yellow callout box points to the 'Configuration des shells' option with the text: "Ajoute des shells, et modifie les paramètres de l'interpréteur."

The background code in the editor shows a function named 'Cercle()' that generates a random point on a circle and prints the coordinates. The code is as follows:

```
6
7 #path = r'G:\PROJETS\BISTID\RESSOURCES LOGICIELS\PYTHON\ESSAIS_PYZO\2ICN_2016_PYTHON\FS ...
8 #chdir(path)
9
10 def Cercle()
11     """ Dessine un cercle de rayon r """
12     rayon r """
13     x = random.random() * 2 * r - r
14     y = random.random() * 2 * r - r
15     r = 10
16
17     # on dessine un cercle
18     item = Canvas.create_oval(x, y, x+r, y+r, outline='black', fill='black')
19
20     print("Création du cercle (item" , item ,)")
21     # affichage de tous les items de Canevas
22     print(Canvas.find_all())
23
24 def Undo():
```





Il est souhaitable de mettre le répertoire de travail dans le champ

Configurations du shell [?] [X]

Python Shell Python 3.4.0 [Ajouter une configuration]

nom Python

exe and Settings\Patrick\Bureau\pyzo2015a\python.exe [default]

IPython Utilise IPython s'il est disponible.

Gui PySide - LGPL licensed wrapper to Qt (recommended)

pythonPath

Configuration par défaut du système

startupScript C:\path\to\script.py

Configuration par défaut du système
 Fichier à exécuter au démarrage
 Code à exécuter au démarrage

startDir G:\GPRO\USB\STI2D\RESSOURCES LOGICIELS\PYTHON\ESSAIS_F

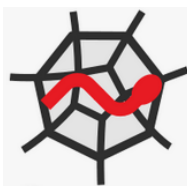
argv ;; Arguments en ligne de commande (sys.argv). arg1 arg2 "arg with spaces"

environ EXAMPLE_VAR1=value1
IEP_PROCESS_EVENTS_WHILE_DEBUGGING=1

[X] Supprimer

Cancel Done





8.2 Choix du répertoire de travail dans le code

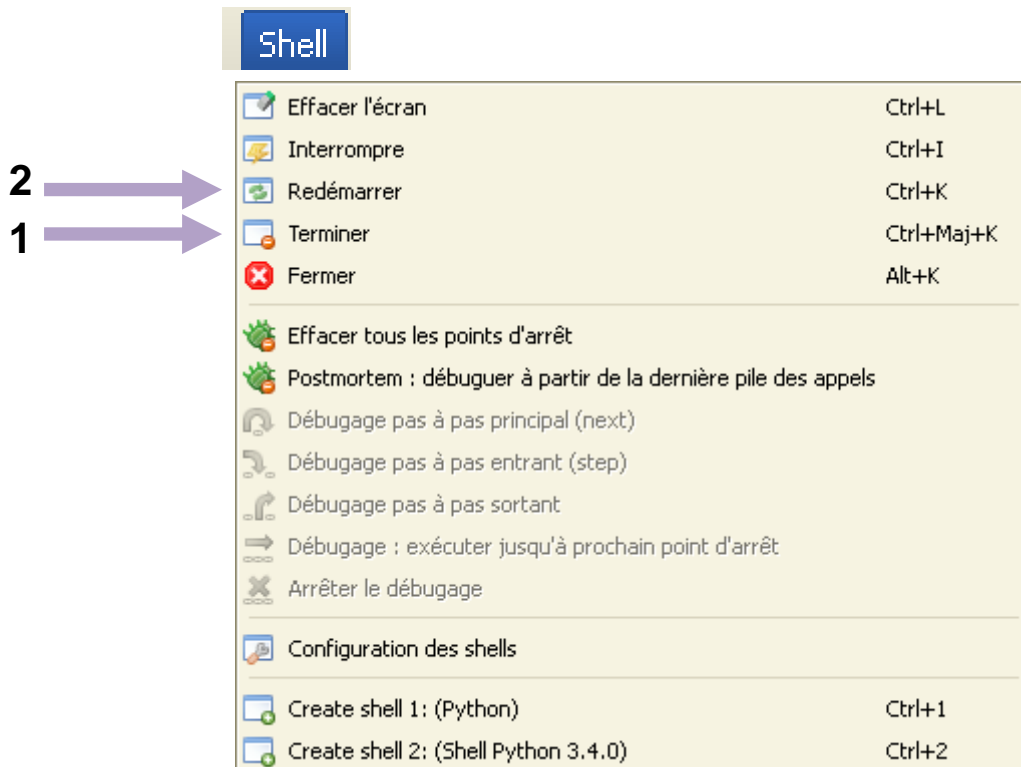
On peut modifier le répertoire courant de travail avec l'instruction `chdir(monRépertoire)` de la bibliothèque `os` (Operating Système)

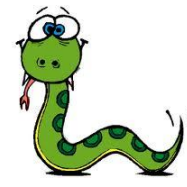
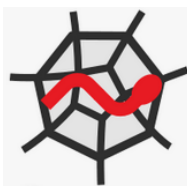
Noter le 'r' devant la chaîne de caractère, cela pour forcer une interprétation qui comprends les caractères \ de windows.

```
from os import *
path = r'G:\GPRO\USB\STI2D\RESSOURCES LOGICIELS\PYTHON\ESSAIS_PYZO\2ICN_2016_PYTHON\Ch 9 Memory'
chdir(path)
```

8.3 En cas de plantage

Si tout vas mal, si ce qui fonctionnait ne fonctionne plus c'est peut-être qu'une erreur dans un script empêche l'interpréteur de fonctionner correctement. Dans ce cas il est possible de l'arrêter et de le redémarrer. Avec les commandes dans le menu shell :



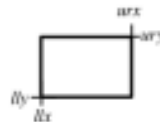
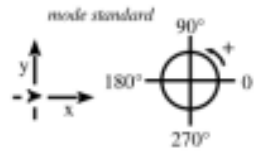


9 Aide mémoire turtle

Coordonnées / environnement

Par défaut, en mode *standard*, direction initiale vers la droite, utilisation d'un repère cartésien orthonormé, angles sens trigonométrique et en degrés.
La position 0,0 est placée au centre de la fenêtre.
En mode *logo*, la direction initiale est vers le haut et les angles sont positifs dans le sens des aiguilles d'une montre (sens inverse trigo).
En mode *world*, unités pixels, le repère n'est pas nécessairement normé (pixels non carrés).

degrees () expression des angles en degrés (tour=360°)
degrees (n) expression des angles unité au choix (tour=n)
radians () expression des angles en radians (tour=2π = 2x3.14...rad)
mode (m) fixe le mode de coordonnées : "standard", "logo", "world"
title (t) fixe le titre de la fenêtre
screensize () → (larg,hauf) dimensions de la fenêtre
screensize (l, hf, coul) fixe dimensions de la fenêtre et couleur de fond
setup (...) fixe position et dimensions de la fenêtre
window_width () → larg largeur de la fenêtre
window_height () → hauf hauteur de la fenêtre
setworldcoordinates (llx, lly, urx, ury) fixe système de coordonnées (fait un **reset ()**)
bgcolor (f[coul]) fixe/trend couleur du fond
bgpic (f[nom]) fixe/trend l'image de fond (nom fichier gif, 'nopic' pour supprimer l'image)



Formes

- ▲ "arrow"
- 🐢 "turtle"
- "circle"
- "square"
- ▲ "triangle"
- ▲ "classic"

Utilisées aussi comme tampons (cf **stamp ()**).

Couleurs

Turtle utilise les noms des couleurs de Tk, dont voici un petit extrait.

- "black"
- "white"
- "grey"
- "red"
- "orange"
- "green"
- "blue"
- "navy"
- "yellow"
- "gold"
- "tan"
- "brown"
- "sienna"
- "wheat"
- "cyan"
- "pink"
- "salmon"
- "violet"
- "purple"

Collection des couleurs sur <http://wiki.tcl.tk/37701>
Noms+valeurs RGB sur <https://www.tcl.tk/man/tcl8.6/TkCmd/colors.htm>

Codes RGB

r=rouge g=vert b=bleu
(red) (green) (blue)
Via une chaîne de valeurs hexa, composantes sur 4/8/12 bits :
"#rgb"
"#rrggbb"
"#rrrgggbbb"
Ou via tuple de 3 flottants entre 0.0...1.0 ou de 3 entiers entre 0...255:
(r, g, b)
Voir **colormode ()**

Position & Déplacements

forward (distance) avance **fd**
backward (distance) recule **bk** **back**
left (angle) tourne à gauche **lt**
right (angle) tourne à droite **rt**
goto (x, y) vas à la position x,y **setpos**
setx (x) vas à l'abscisse x **setposition**
sety (y) vas à l'ordonnée y
setheading (angle) s'oriente à l'angle **seth**
home () vas à l'origine 0,0
circle (rayon[, angle[, pas]]) cercle/arc/polygone
position () → (x, y) position courante **pos**
xcor () → x abscisse
ycor () → y ordonnée
heading () → a orientation (angle)
distance (x, y) → d distance jusqu'à x,y
distance (pos) → d distance jusqu'à pos (x,y)
towards (x, y) → a angle vers x,y
towards (pos) → a angle vers pos (x,y)
dot (f[taille], coul[f]) trace point à la position
stamp () → id trace tampon tortue à la position
clearstamp (id) efface tampon id
clearstamps (n) efface tampons (tous, n>0 premiers n<0 n derniers)
undo () annule dernier mouvement/trace



Contrôles

reset () réinitialisation complète
resetscreen
clear () effacement de la zone de tracé
clearscreen
tracer () → n périodicité animation tortue
tracer (n/, d/) fixe périodicité animation tortue
delay () → n délai (ms) entre mises à jour
delay (delay) fixe délai (ms) entre mises à jour
update () force mise à jour
speed () → n vitesse de tracé
speed (n) fixe/trend vitesse tracé n, nom ou entier [0...10]
"fastest":0 "fast":10
"normal":6 "slow":3
"slowest":1
hideturtle () masque la tortue **ht**
showturtle () affiche la tortue **st**
isvisible () → v vrai si tortue visible
shape (nom) fixe la forme de la tortue
getshapes () → /nom/ liste des noms de formes
register_shape (nomfichier) enregistre forme via fichier gif
register_shape (nom, coords) enregistre forme via liste de (x,y)
register_shape (nom, shape) enregistre forme via objet Shape
resizemode () → rmode mode redimensionnement tortue
resizemode (rmode) change le mode "auto" "user" "noresize"

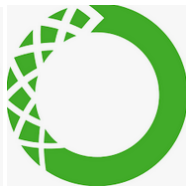
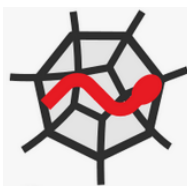
Pinceau

up () lève (pas de trace) **penup** **pu**
down () baisse (trace) **pendown** **pd**
isdown () → e/vr retourne vrai si pinceau baissé
color (cp/, cr/) fixe/trend couleur du pinceau [et du remplissage]
pencolor (coul) fixe/trend couleur du pinceau
fillcolor (coul) fixe/trend couleur du remplissage
width (larg) largeur du trait **pensize**
pen () → tp dico caractéristiques pinceau
pen (p) fixe caractéristiques pinceau via dico
filling () → sb vrai si remplissage actif
begin_fill () démarre tracés de remplissage...
instructions de déplacements
end_fill () ...termine et remplissage des tracés
colormode () → n valeur maximale pour les r g b
colormode (n) 1 ou 255 - val maxi pour les r g b

write("texte") ou **write("texte ")**



[Retour au paragraphe tortue.](#)



10 Installation des logiciels sous pyzo

Pour installer les différents programmes utiles (gratuits) il faut procéder en plusieurs étapes :
(Les exemples sont donnés pour une machine Windows 64 bits)

1. Récupération des logiciels sur les sites :

a) Pyzo : site <http://www.pyzo.org/>

pyzo_distro-2015a.win64.exe

b) Python 3.6 : site <https://www.python.org/downloads/release/python-360/>

python-3.6.0-amd64.exe

2. Téléchargement des modules complémentaires. Les modules d'extensions sont téléchargeables ici :

<http://www.lfd.uci.edu/~gohlke/pythonlibs/>

c) Pygame :

pygame-1.9.3-cp36-cp36m-win_amd64.whl

d) Matplotlib :

matplotlib-2.0.2-cp36-cp36m-win_amd64.whl

3. Installation des modules complémentaires

e) Recopier les deux fichiers .whl dans le répertoire de python 3.6 puis exécuter d'abord la commande suivante :

```
pip install wheel
```

f) Et ensuite pour chacun des modules la commande (exemple pour pygame) :

```
pip install pygame-1.9.2-cp36-cp36m-win_amd64.whl
```

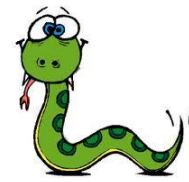
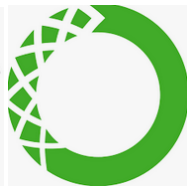
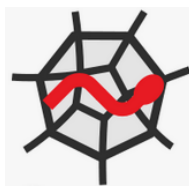
```
C:\Users\Patrick>cd c:\program files\python36
c:\Program Files\Python36>pip install wheel
Collecting wheel
  Downloading wheel-0.30.0-py2.py3-none-any.whl (49kB)
    100% |#####| 51kB 146kB/s
Installing collected packages: wheel
Successfully installed wheel-0.30.0

c:\Program Files\Python36>pip install pygame-1.9.3-cp36-cp36m-win_amd64.whl
Processing c:\program files\python36\pygame-1.9.3-cp36-cp36m-win_amd64.whl
Installing collected packages: pygame
Successfully installed pygame-1.9.3

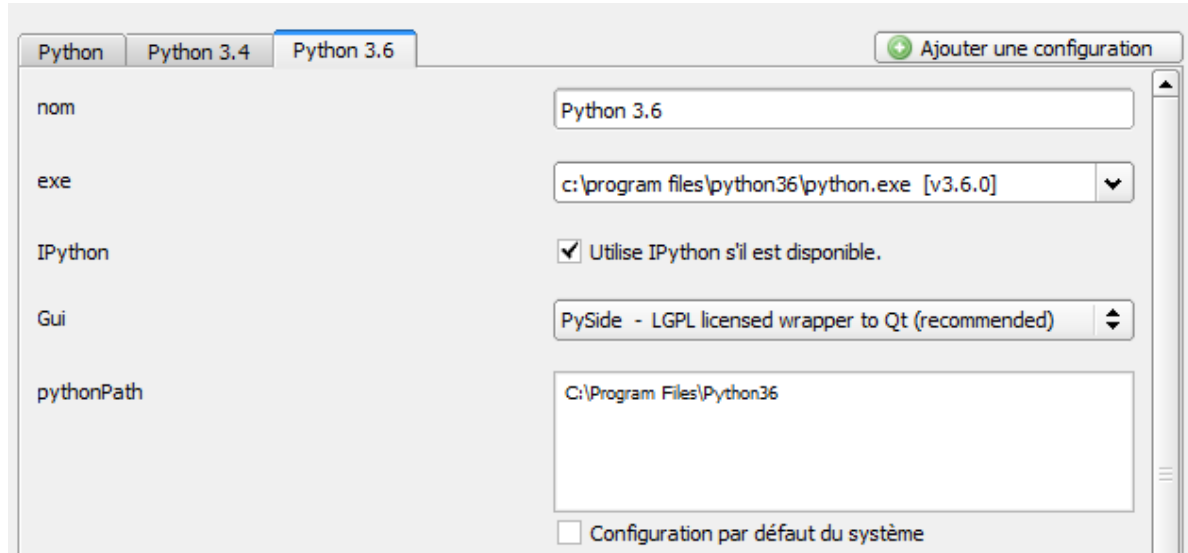
c:\Program Files\Python36>pip install matplotlib-2.1.0-cp36-cp36m-win_amd64.whl
Processing c:\program files\python36\matplotlib-2.1.0-cp36-cp36m-win_amd64.whl
Collecting six>=1.10 (from matplotlib==2.1.0)
  Downloading six-1.11.0-py2.py3-none-any.whl
Collecting python-dateutil>=2.0 (from matplotlib==2.1.0)
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
    100% |#####| 194kB 193kB/s
Collecting pytz (from matplotlib==2.1.0)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |#####| 512kB 284kB/s
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib==2.1.0)
  Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
    100% |#####| 61kB 245kB/s
Collecting cycler>=0.10 (from matplotlib==2.1.0)
  Downloading cycler-0.10.0-py2.py3-none-any.whl
Collecting numpy>=1.7.1 (from matplotlib==2.1.0)
  Downloading numpy-1.13.3-cp36-none-win_amd64.whl (13.1MB)
    100% |#####| 13.1MB 82kB/s
Installing collected packages: six, python-dateutil, pytz, pyparsing, cycler, nu
mpy, matplotlib
Successfully installed cycler-0.10.0 matplotlib-2.1.0 numpy-1.13.3 pyparsing-2.2
.0 python-dateutil-2.6.1 pytz-2017.3 six-1.11.0

c:\Program Files\Python36>
```

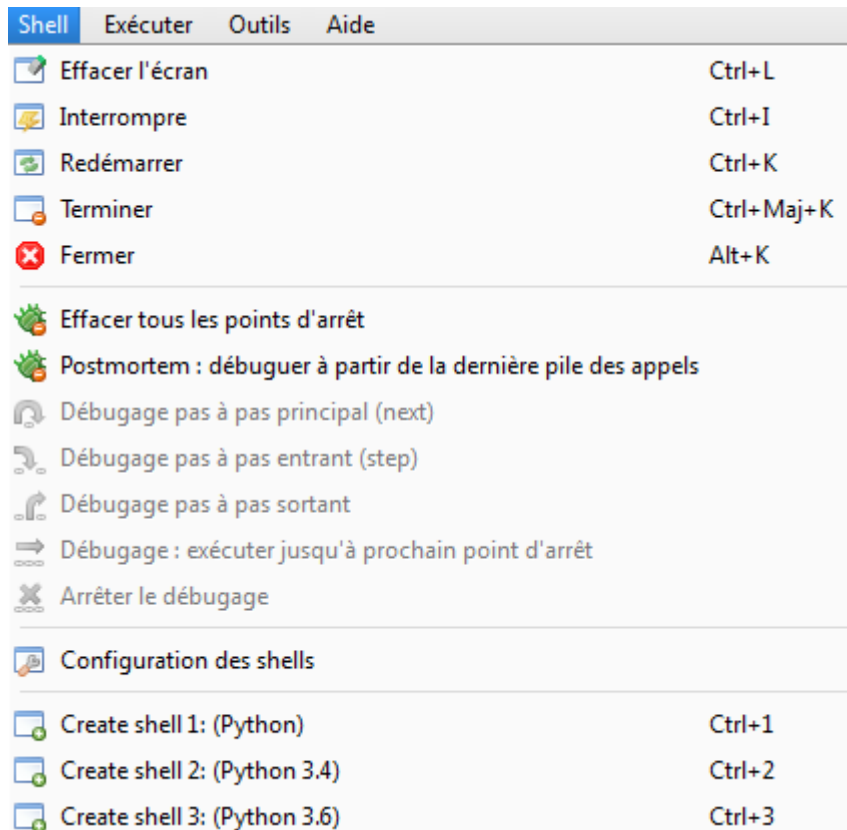


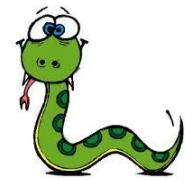
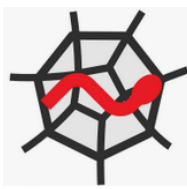


4. Ajouter un shell dans pyzo et le faire pointer sur le programme python.exe de la version 3.6 que l'on souhaite ajouter :



Les différents shell installés sont disponibles dans l'onglet shell :





11 Installation de Python 3.7 et des bibliothèques 'NSI'

11.1 Installation de python37

Sur une machine 64 bits exécuter la commande :  `python-3.7.3-amd64.exe`

Ne pas cocher la modification du Path, installer pour tous les utilisateurs.

11.2 Méthode pour l'ajout des modules

Les modules sont ajoutés avec pip, attention il peut y avoir plusieurs versions de python installées donc pour cela on utilise la commande particulière pour une installation dans le dossier de la version 37:

```
c:\Program Files\Python37>PY -3.7 -m pip install <nom_du_module>
```









```
<Dossier d'installation de python37>\PY -3.7 -m pip install <nom_du_module>
```

Avant d'installer les modules complémentaires on installe wheel

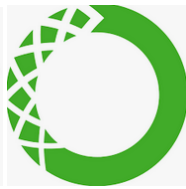
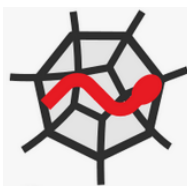
```
c:\Program Files\Python37>PY -3.7 -m pip install wheel
```

11.3 Installation des modules avec PIP

Pour trouver les modules lien : <https://www.lfd.uci.edu/~gohlke/pythonlibs/>

1	 <code>numpy-1.16.3+mkl-cp37-cp37m-win_amd64.whl</code>
2	 <code>matplotlib-3.0.3-cp37-cp37m-win_amd64.whl</code>
3	 <code>pygame-1.9.4-cp37-cp37m-win_amd64.whl</code>
4	 <code>mysqlclient-1.4.2-cp37-cp37m-win_amd64.whl</code>
5	 <code>Pillow-6.0.0-cp37-cp37m-win_amd64.whl</code>
6	 <code>pandas-0.24.2-cp37-cp37m-win_amd64.whl</code>
7	 <code>scikit_learn-0.21.1-cp37-cp37m-win_amd64.whl</code>
8	 <code>noise-1.2.3-cp37-cp37m-win_amd64.whl</code>





11.4 Installation sans modules whl

PYinstaller pour créer des exécutables autonomes

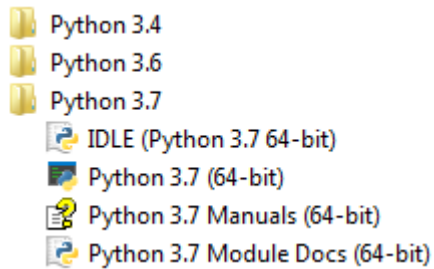
```
c:\Program Files\Python37>PY -3.7 -m pip install pyinstaller
```

Et seaborn

```
c:\Program Files\Python37>PY -3.7 -m pip install seaborn
```

11.5 Un script python pour tester que tout est bien installé

Pour lancer l'IDLE de Python



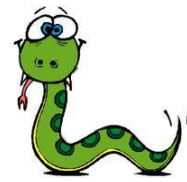
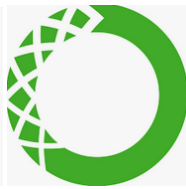
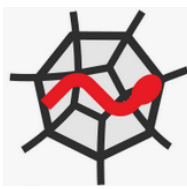
Voilà le script  Test_modules_python37.py

```
import matplotlib
import numpy
import noise
import MySQLdb as sql
import sklearn
import pandas
import seaborn
import pygame
```

Et son exécution il n'y a pas d'erreurs tout est ok :

```
===== RESTART: P:/LOGICIELS/Python/PYTHON 37/Test_modules_python37.py =====
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
>>> |
```





11.6 Un traitement par lot pour l'installation des modules

```
:: INSTALLATION COMPLETE DANS PYTHON37 DE MODULES
:: ADDITIONNELS
:: Les fichiers whl des modules devront être recopiés
:: dans le dossier d'installation de python 37
:: Ce fichier batch doit être lancé dans le répertoire
:: d'installation de Python37
```

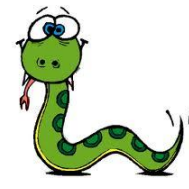
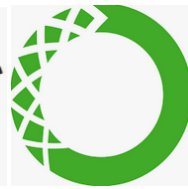
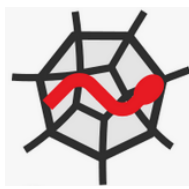
```
PY -3.7 -m pip install wheel
PY -3.7 -m pip install numpy-1.16.3+mkl-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install matplotlib-3.0.3-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install pygame-1.9.4-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install mysqlclient-1.4.2-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install Pillow-6.0.0-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install pandas-0.24.2-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install scikit_learn-0.21.1-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install noise-1.2.3-cp37-cp37m-win_amd64.whl
PY -3.7 -m pip install pyinstaller
PY -3.7 -m pip install seaborn
```

11.7 Sous W10

Créer un dossier à la racine de c:\ pour ne pas installer le logiciels dans les dossiers Programms de Windows.

Par exemple : c:\Programmes\Python37





12 Ressources

Les livres :

- apprenez_a_programmer_en_python.pdf (site du zéro)
- apprendre-python3.pdf (très complet le must)
- Ce livre disponible gratuitement sur le web ou sur <https://python.developpez.com/cours/apprendre-python3/>



Est à consulter et à pratiquer sans réserve c'est votre manuel de cours TP pour la première et la terminale.



Apprendre le langage de programmation python :

<https://www.apprendre-en-ligne.net/pj/index.html>

<http://tkinter.fdex.eu/>

<https://www.apprendre-en-ligne.net/pj/index.html>

<http://apprendre-python.com/>

http://fsincere.free.fr/isn/python/cours_python.php

http://fsincere.free.fr/isn/python/cours_python_tkinter.php

<http://www.dsimb.inserm.fr/%7Efuchs/python/python.html>

Les logiciels :

<http://www.pyzo.org/>

<https://www.python.org/downloads/release/python-360/>

<http://www.lfd.uci.edu/~gohlke/pythonlibs/>

