Options ICN-ISN

Initiation à la programmation

Sommaire :

Options ICN-ISN

1 Pour utiliser l'outil :

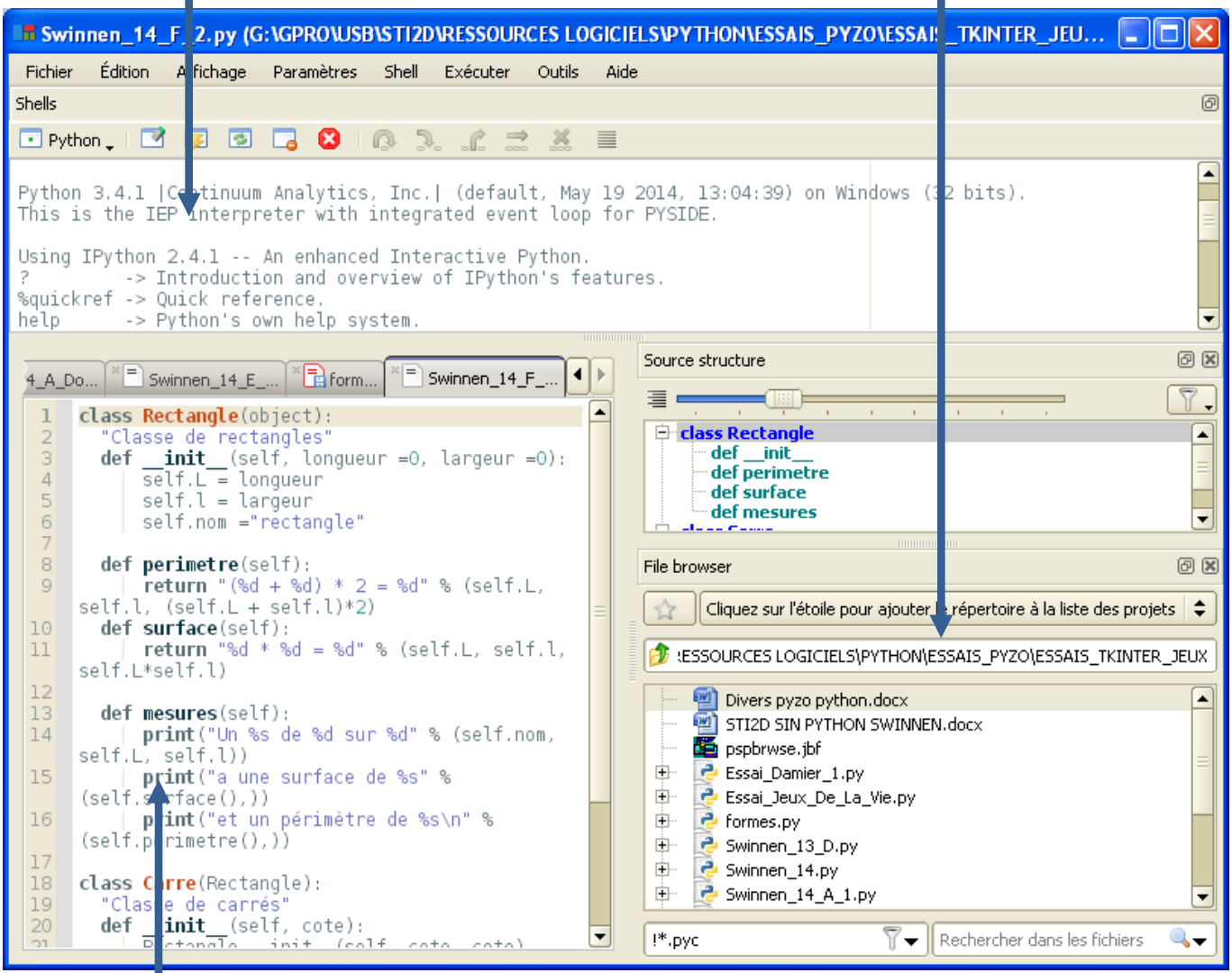
L'outil est très complet, et gratuit, pour l'utiliser le téléchargement c'est ici http://www.pyzo.org/

Lancement 

1.1 La fenêtre de travail de PYZO :

Python interactif :

Le dossier de travail



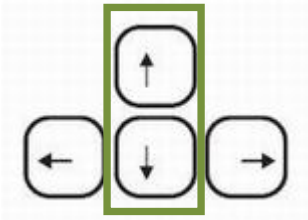
Les fichiers de codes

CTRL + Entrée pour les exécuter

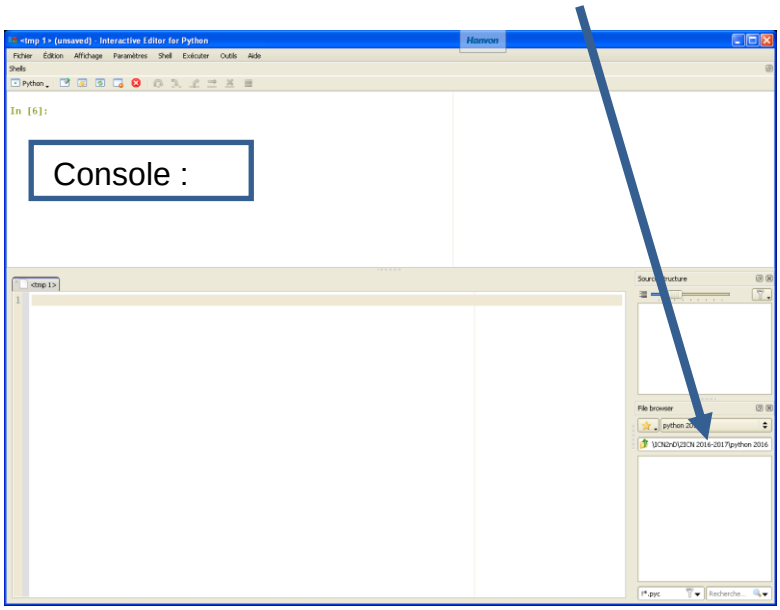
Options ICN-ISN

1.2 Utilisation de la console

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



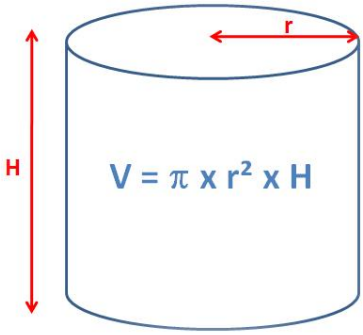
Les touches flèches du clavier permettent de rappeler les dernières commandes mémorisées dans la console.



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, p

```
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 ⇨ r est réel

toto = 2 ⇨ toto est entier

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

Options ICN-ISN

1.3 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
```

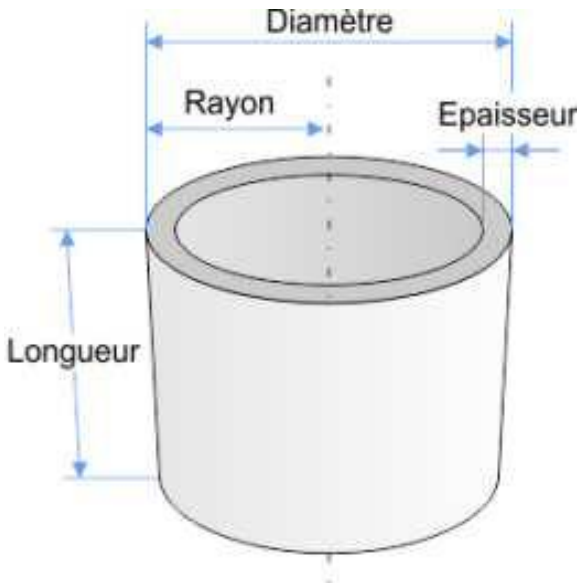
c) Liste du contenu d'une bibliothèque

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

```
help(math)
```

1.4 Calculs avec la console

Essai Python 1.



Écrire les formules ici :

Calcul du volume de la paroi

Calcul du volume intérieur

Options ICN-ISN

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

1.5 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 ?

1.6 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
```

Options ICN-ISN

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 chaines 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
```

Options ICN-ISN

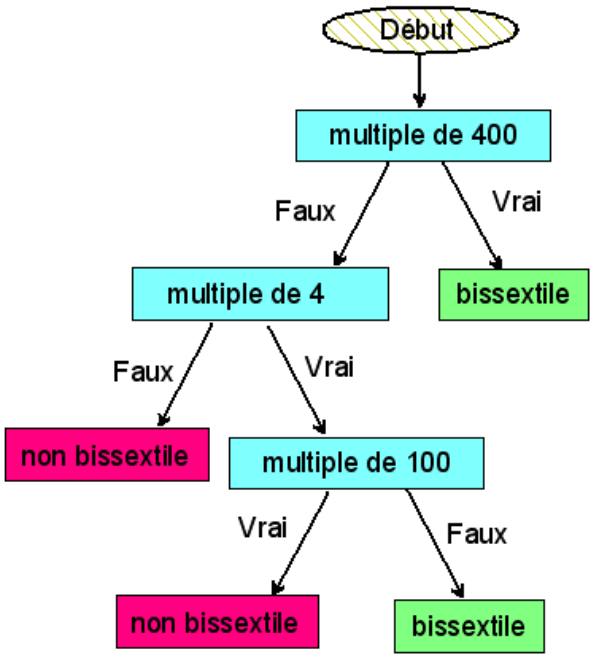
1.7 Un premier petit programme

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
>
> *À propos de ce résultat ▪ Commentaires*

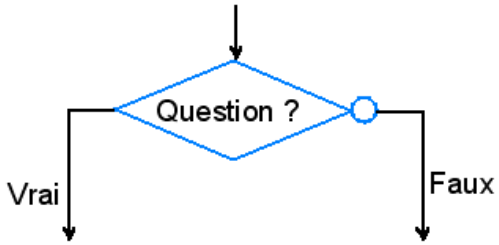
Le raisonnement est le suivant :

- 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 qu
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.

Options ICN-ISN

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.

1.8 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

Options ICN-ISN

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 :

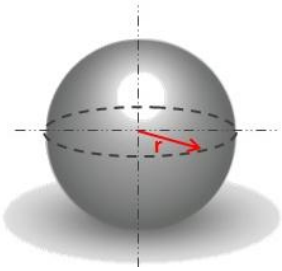
and	as	assert	break	class	continue	def
del	elif	else	except	False	finally	for
from	global	if	import	in	is	lambda
None	nonlocal	not	or	pass	raise	return
True	try	while	with	yield		

2 Plus loin avec le langage

2.1 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 :

Fonction n^3

Fonction calcul du volume

Le programme principal

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

# Calcul de l'élévation 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')
```

Options ICN-ISN

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.

2.2 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

Options ICN-ISN

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 !

2.3 Le formatage des données en sortie méthode recommandée

Dans l'exemple proposé dans le livre de Swinnen la chaine 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 chaine
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))
```

Options ICN-ISN

Le champ peut contenir des indications de formatage :

```
12 ch ="La couleur est {} et la température vaut {:6.2e} °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
```

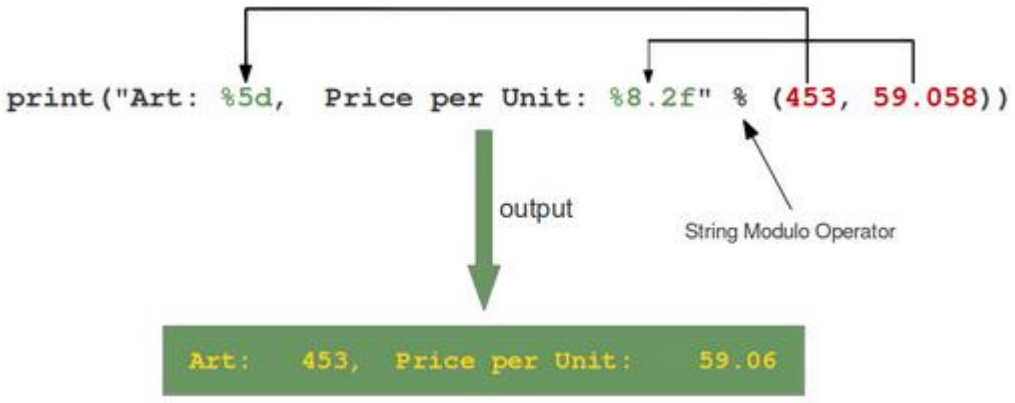
Pour les descriptions des formats voir le paragraphe suivant.

2.4 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 maitriser le format de sortie utiliser des commandes de formatage. Le principe est expliqué sur le schéma ci-dessous[1] :



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

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

Options ICN-ISN

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
```

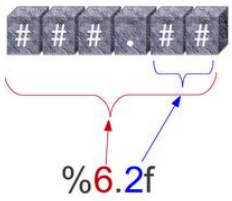
Essai Python 9. Modifiez votre programme précédent pour y intégrer le formatage.

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

Options ICN-ISN

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 *float* (minuscule).
`'E'`	Format exponentiel pour un *float* (en capitales).
`'f'`	Format décimal pour un *float*.
`'F'`	Format décimal pour un *float*.
`'g'`	Format *float*. 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 *float*. 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 `repr()`).
`'s'`	String (convertit n'importe quel objet Python avec `str()`).
`'a'`	String (convertit n'importe quel objet Python en utilisant `ascii()`).
`'%'`	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 »).

Options ICN-ISN

2.5 Quelques possibilités de mise en forme

- Choix du séparateur quand on affiche plusieurs chaines elles sont séparées par un caractère espace. Il est possible de choisir une chaine 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= " "

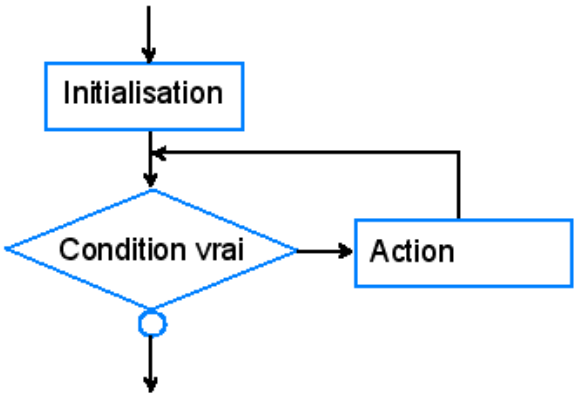
```
# Impression avec formatage et affichage amélioré
n=0
while n<4:
    print("Mon chiffre = %2d" % (n)," ",sep=" ++ ", end = "")
    n=n+1
```

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

Essai Python 10. Codez et testez cet exemple.

2.6 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.

Options ICN-ISN

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

2.7 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

Options ICN-ISN

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

2.8 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
```

Options ICN-ISN

2.9 Les listes et les tables : conjuguons un verbe

La conjugaison est un bon exemple d'utilisation de tables et de manipulation de chaines 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

Options ICN-ISN

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

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

```
verbe =  patauger
verbe[0] =  p
longueur du verbe =  8

verbe[2:5] =  tau

pronom[5] =  ils
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
```

Options ICN-ISN

3 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 python[2].

3.1 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.

Options ICN-ISN

3.2 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 tireau 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
```

Options ICN-ISN

3.3 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
```

Options ICN-ISN

3.4 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.

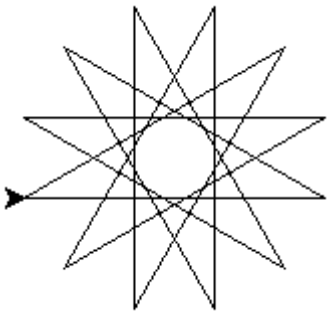
Essai Python 24. Proposer et tester une méthode permettant de vérifier le dé pipé.

Options ICN-ISN

4 S'amuser avec le module tortue

4.1 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 exemple[3].

```
# 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

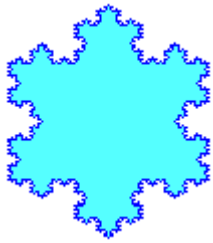
Mathématicien suédois

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.

Options ICN-ISN

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

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

Une liste détaillée ici

Options ICN-ISN

5 Réglages divers

5.1 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 -*-
```

5.2 Accès à des caractères spéciaux

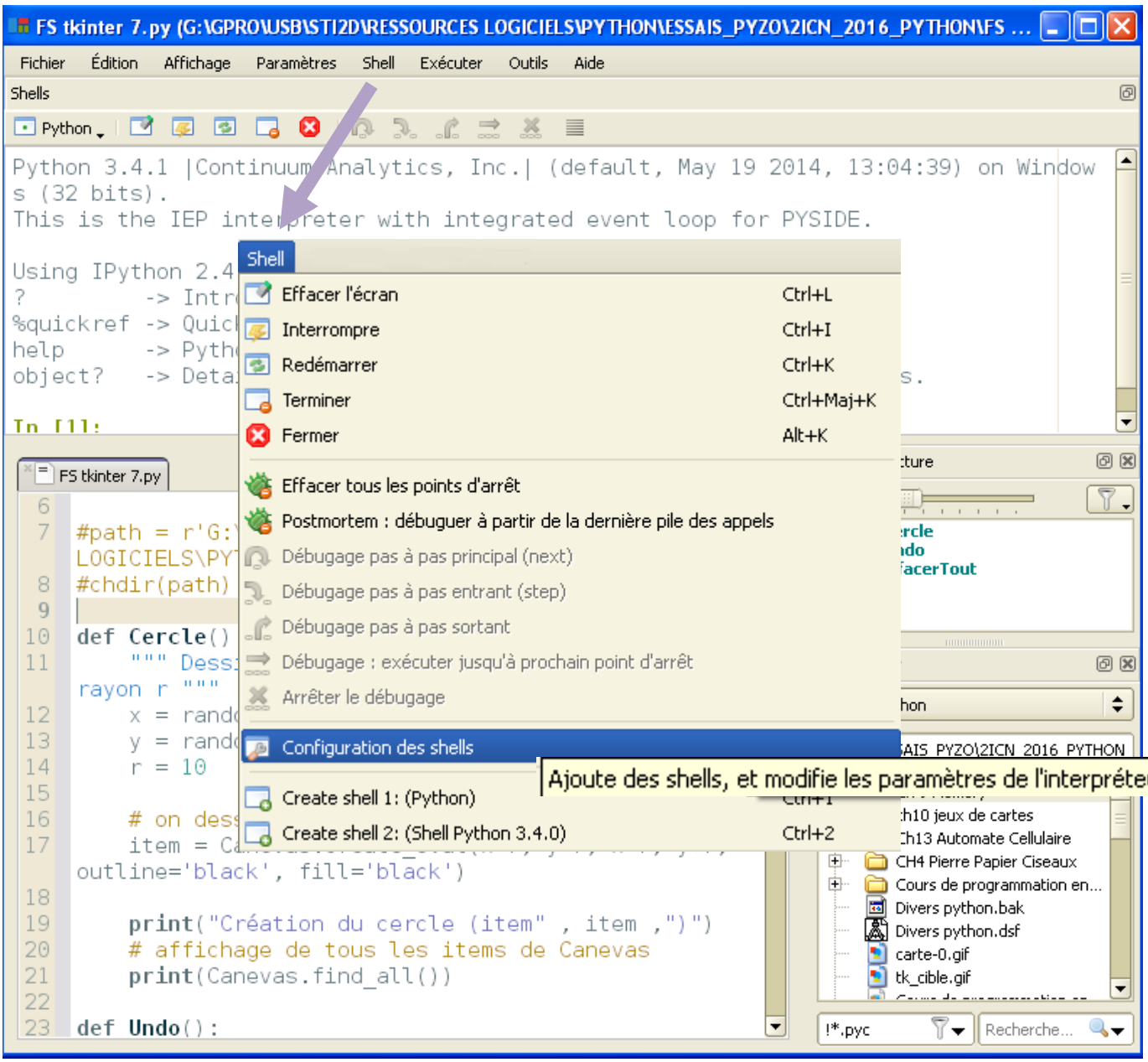
```
# Exemple d'accés à des caractères non présents sur le clavier

# 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
```

Options ICN-ISN

6 Configuration de pyzo

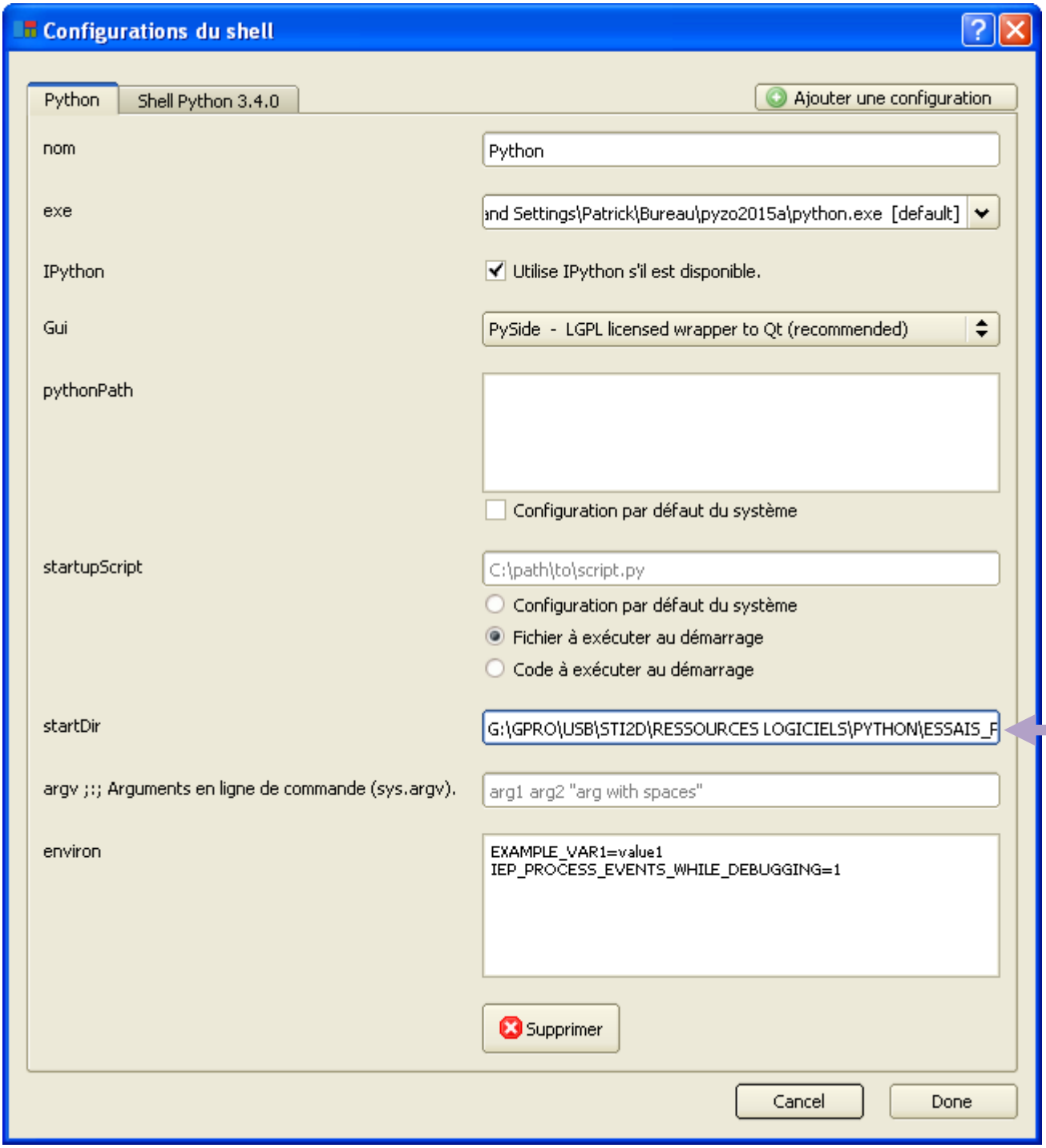
L'environnement de développement Pyzo permet de travailler avec plusieurs versions de pyzo. Il est possible également de configurer celle qui est en cours d'utilisation.

6.1 Configuration avec l'outil Pyzo



Options ICN-ISN

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



Options ICN-ISN

6.2 Choix du répertoire de travail dans le code

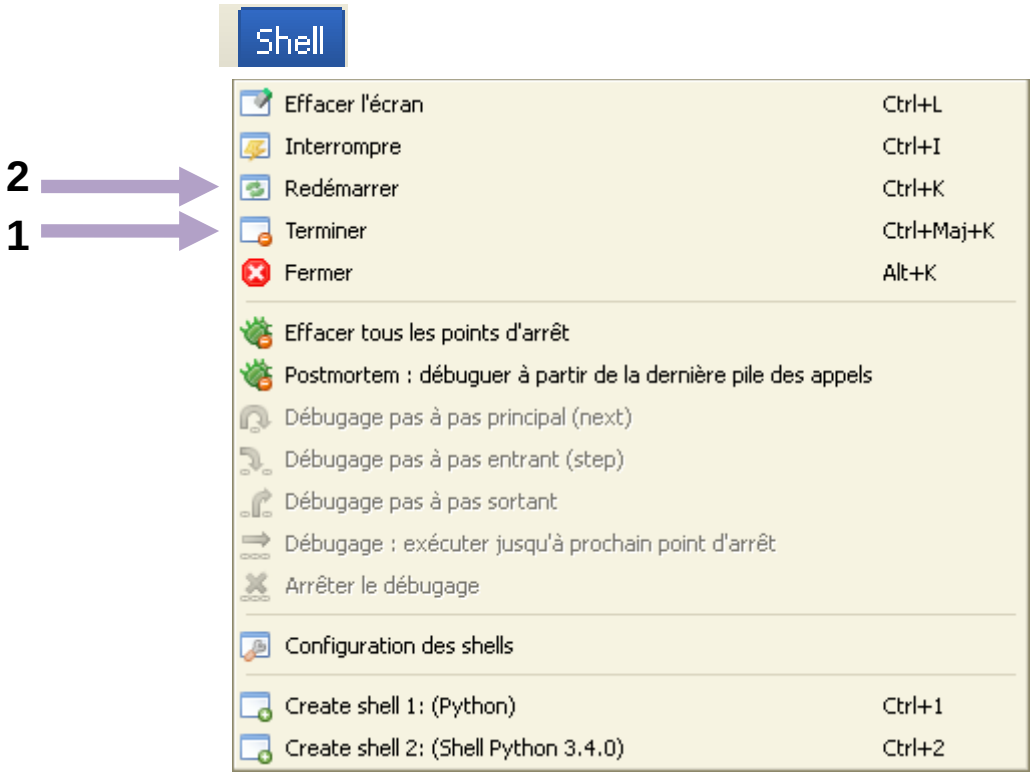
On peut modifier le répertoire courant de travail avec l'instruction chdir(monRépertoire) de la bibliothèque os (Operating Système)

Noter le 'r' devant la chaine 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)
```

6.3 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 :

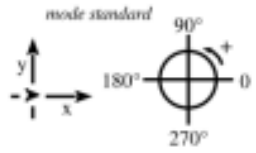


Options ICN-ISN

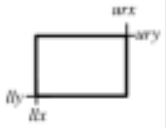
7 Aide mémoire turtle

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 = 2×3.14…rad)
`mode(m)`	fixe le mode de coordonnées : `"standard"`, `"logo"`, `"world"`
`title(t)`	fixe le titre de la fenêtre
`screensize()→(larg,haut)`	dimensions de la fenêtre
`screensize(l,h[,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()→haut`	hauteur de la fenêtre
`setworldcoordinates(llx,lly,urx,ury)`	fixe système de coordonnées (fait un `reset()`)
`bgcolor([coul])`	fixe/rend couleur du fond
`bgpic([nom])`	fixe/rend l'image de fond (nom fichier gif, `'nopic'` pour supprimer l'image)



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` `setposition`
`setx(x)`	vas à l'abscisse x
`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([taille[,coul]])`	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()`	annuler dernier mouvement/trace

Pinceau

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

`write('texte')` ou `write("texte ")`

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/rend 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"`

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.

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()`

Retour au paragraphe tortue.

Options ICN-ISN

8 Installation des logiciels

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

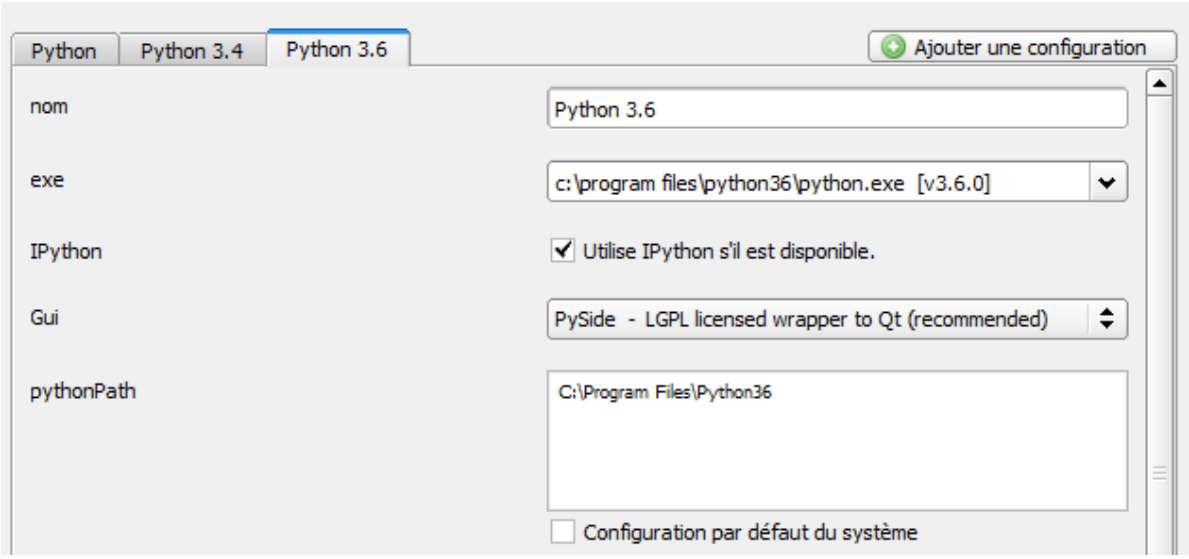
   e) Recopier les deux fichiers .whl dans le répertoire de python 3.6 puis exécuter d'abord la commande suivante :

   ```
   pip install wheel
   ```

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

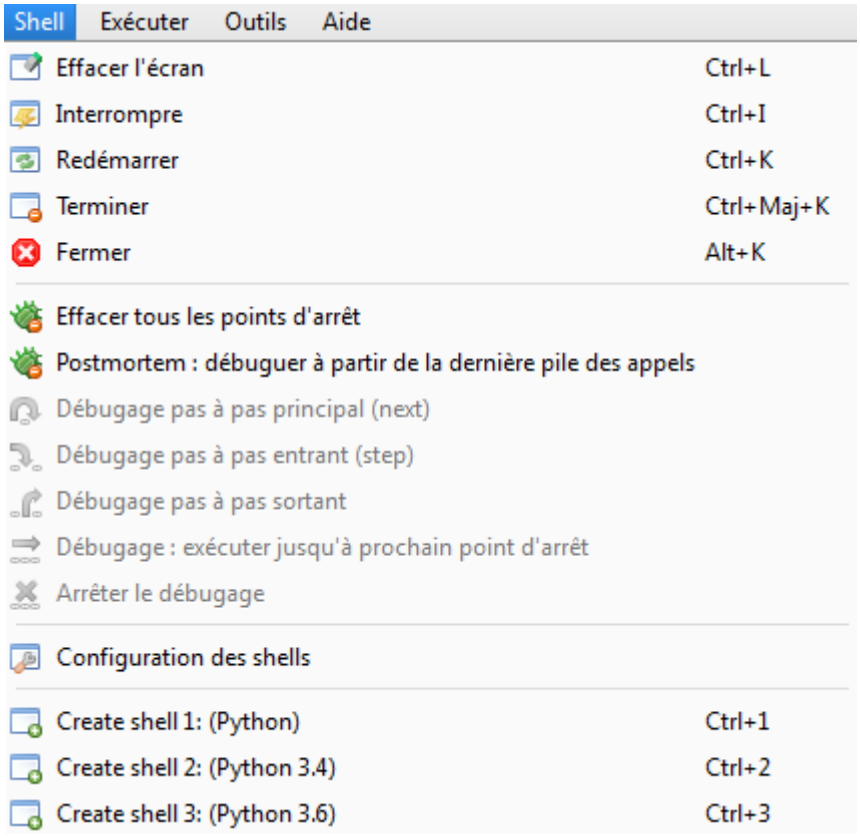
   ```
   pip install pygame-1.9.2-cp36-cp36m-win_amd64.whl
   ```

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 :



Options ICN-ISN

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



Options ICN-ISN

9 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/