Introduction à la programmation avec Python

https://www.python.org/doc/
https://docs.python.org/3.5/library/index.html
http://fsincere.free.fr/isn/python/cours_python.php
http://python.swaroopch.com/

1 Pour débuter en python

Quelques exemples d'opérations

```
print("Hello world")
a = 1
b = 2
print("a = ", a)
print("b = ", b)
# un premier commentaire
# addition:
c = a + b
print("c = ", c)
a = a + 1
print("a = ", a)
a += 1  # permet aussi d'incrémenter a de 1
print("a = ", a)

# on peut faire de même avec les autres opérations:
a -= 1

a *= 2  # correspond à
a = a * 2

a /= 2  # correspond à
a = a / 2

# pour élever un nombre à une puissance:
a**3
```

Attention la puissance ne s'écrit pas comme sur la calculatrice a^2 mais a**2. L'opération a^2 a un autre sens en python (lié à l'écriture en base deux des nombres).

Un peu d'arithmétique :

```
a = 20
b = 7
q = a // b   # division entière
r = a % b    # reste de la division euclidienne
print(a, " = ", b, "*", q, "+", r)
```

Condition : si ... alors ...

En python, le symbole = est utilisé pour l'**affectation**, il faut donc un autre code pour tester une **égalité** : "==". Rappelons qu'une assertion peut être soit vraie (True) ou fausse (False). Tester successivement les différentes lignes suivantes dans le module python directement :

```
1 + 1 == 3
1 + 1 == 2
1 + 1 == 2 and 2 * 3 == 6
1 + 1 == 1 or 2 * 3 == 5
not( 1 + 1 == 2 )
```

La condition si se dit « if » en anglais :

```
a = 1
if(a == 1):
  print("hello!")
else:
  print("Good bye")
```

On notera qu'on n'écrit pas « then » (alors) pour effectuer un test en python. Il faudra aussi veiller à bien écrire les deux-points.

Compléter le programme suivant :

```
a = 1
b = -4
if _________:
  print("a et b sont de signes opposés")
```

Tester le précédent programme avec d'autres valeurs pour a et b.

Vous avez du remarquer qu'il est pénible de devoir relancer le programme pour tester avec d'autres valeurs. La solution se trouve dans l'utilisation de la commande `input` :

```
a = input("Entrer un nombre pour a: ")
b = input("Entrer un nombre pour b: ")
# à défaut, la commande input retourne une chaîne de caractères
# il faut la convertir sous forme d'un entier:
a = int(a)
b = int(b)
if a*b < 0:
  print("a et b sont de signes opposés")
else:
  print("a et b sont de même signe")
```

L'ajout de la commande `int()` permet de convertir la chaîne de caractère entrée par l'utilisateur en nombre.

Les listes

Une liste de nombres en python s'écrit entre crochets comme suit :

```
l = [3,-5,4,7,12,1,-5]
```

Par convention, le premier terme est de rang 0 :

```
print( l[0] )
```

On accède au $i$ème terme ainsi

```
i = 3
print( l[i] )
```

Quelques opérations avec les listes :

```
l = [1,2]
print(l)
l = [1,5] + [2,4]  # concatenation de deux listes
print("concatenation: ", l)

l.reverse() # modifie l'ordre des elements de la liste en les plaçant dans l'ordre inverse
print(l)

l.sort()  # trie les elements dans l'ordre croissant
print(l)
```

Boucle : pour

La particularité du langage python est de définir la boucle **pour** à l'aide des listes. Voici un exemple :

```
l = [1,2,3,4]
for x in l:
  # x va prendre successivement toutes les
  # valeurs de la liste l
   print(x)
```

A priori pour afficher les 10 premiers entiers naturels, il faudrait définir la liste correspondante. La commande `range(n)` permet de le faire. Elle génère la liste des nombres entiers naturels de 0 à $n-1$. Ainsi, pour afficher les 10 premiers entiers naturels, le code est le suivant :

```
for i in range(11):
  print(i)
```

Exercice 1.
1. Modifier le programme précédent pour afficher les 20 premiers entiers.
2. Compléter le code suivant qui permet d'afficher le maximum des éléments de la liste l :

```
l = [-1,3,-4,5,10,4,2.3]
M = _____
for _____________:

print(M)
```

Boucle : tant que La boucle tant que s'écrit à l'aide du mot clé `while` :

```
while condition:
  # instructions ...
```

Exercice 2. Considérons la suite (u_n) définie par récurrence comme suit :

$$\begin{cases} u_0 = 100 \\ u_{n+1} = 1.01\,u_n + 10 \quad \text{pour tout entier } n \end{cases}$$

Voici un programme en relation avec cette suite :

```
u = 100
while u < 1000:
  u = 1.01 * u + 10
print(u)
```

1. Que permet de faire ce programme ?
2. Le modifier pour qu'il affiche l'indice n et non pas u_n.

Chaînes de caractères

```
s = "ceci est une chaîne de caractères "
print(s)
t = "et en voici une autre"
print(s + t)
```

Il est possible de convertir un nombre codé dans une variable en chaîne de caractères :

```
a = 2
print(str(a))
# application:
b = 3
s = "dans la variable a se trouve le nombre " + str(a) + " et dans la variable b se trouve le nombre " + str(b)
print(s)
```

Pour écrire une chaîne de caractère sur plusieurs lignes :

```
s = '''première ligne
seconde ligne'''
print(s)
# ou bien
s = "première ligne \n seconde ligne"
print(s)
```

Le code `\n` correspond à un retour à la ligne et `\t` correspond à une tabulation.

Nombres aléatoires

Pour générer un nombre aléatoire compris dans l'intervalle $[0;1[$, il faut charger le module **random** :

```
from random import *

X = random()
print(X)
```

Pour générer un nombre entier aléatoire compris entre a et b, on peut utiliser la commande `randint(a,b)` :

```
X = randint(1,6)
print (X)
```

2 Sommes des puissances entières d'entiers

Exercice 3 (Olympiades Amériques-Antilles-Guyane 2016).

Partie 1 : sommes, sommes de carrés, sommes de cubes

Les ensembles $A = \{1, 9, 11\}$ et $B = \{3, 5, 13\}$ possèdent des propriétés qui attisent la curiosité. On remarque en effet que $1 + 9 + 11 = 3 + 5 + 13$ et que $12 + 92 + 112 = 32 + 52 + 132$.

Dans la suite, on dira que la paire d'ensembles d'entiers $\{A, B\}$ possède la propriété :

- S_1 si les deux ensembles ont le même nombre d'éléments et si la somme des éléments de A est égale à la somme des éléments de B.
- S_2 si elle possède la propriété S_1 et si de plus la somme des carrés des éléments de A est égale à la somme des **carrés** des éléments de B.
- S_3 si elle possède la propriété S_2 et si de plus la somme des cubes des éléments de A est égale à la somme des **cubes** des éléments de B.

Étant donné un entier impair n, on se demande s'il est possible de partager l'ensemble $\{0, 1, 2, \ldots, n-1, n\}$ en deux parties A et B telles que $\{A, B\}$ possède une des propriétés évoquées ci-dessous. Par convention, A sera la partie qui contient 0.

1. Dans le cas $n = 3$, peut-on trouver $\{A, B\}$ possédant la propriété S_1 ?
2. Même question dans le cas $n = 5$.
3. a) Si l'ensemble $\{0, 1, 2, 3, 4, 5, 6, 7\}$ peut être partagé en deux parties A et B telles que les propriétés S_1 et S_2 soient satisfaites, quelle est la somme des éléments de A ?
   b) Quelle est la somme des carrés des éléments de A ?
   c) Déterminer les parties A et B solutions du problème.
   d) Écrire un programme en python pour tester si la partition $\{A, B\}$ vérifie la propriété S_3.
4. Lorsque l'ensemble $\{0, 1, 2, \ldots, n-1, n\}$ peut être partagé de sorte à satisfaire une ou plusieurs des propriétés S_i, on adopte le système de représentation suivant : Chaque case de la seconde ligne des tableaux ci-dessous contient 1 si le nombre appartient à A, 0 si le nombre appartient à B.
   a) Compléter les deux tableaux :

$n = 3$	0	1	2	3
	1			

$n = 7$	0	1	2	3	4	5	6	7
	1							

   b) Pour afficher le tableau $n = 3$ en python, on peut utiliser le programme suivant :

```
ligne1 = ""
ligne2 = ""
l = [1,0,0,1] # ligne du tableau codée sous forme de liste
              # si i appartient à A alors l[i] = 1
for i in range(4):
  ligne1 += str(i) + "\t"
  ligne2 += str(l[i]) + "\t"
print(ligne1)
print(ligne2)
```

   Modifier le programme pour afficher le tableau correspondant à $n = 7$.
   c) L'observation de ces deux tableaux fait naître l'idée qu'en dédoublant chaque 0 en 01 et chaque 1 en 10, on transforme une séquence intéressante en une autre, dont la taille est doublée.
      i. Écrire un programme qui permet de passer de la liste l pour $n = 7$ à la liste, notée L pour $n = 15$.
      ii. Compléter le programme pour afficher le résultat.
      iii. Écrire une programme qui vérifie que la partition $\{A, B\}$ obtenue pour $n = 15$ vérifie la propriété S_3.

Partie 2 : naissance d'une suite

Inspirés par les questions précédentes, on étudie la suite définie par $t_0 = 1$ et la relation de récurrence :

$$\begin{cases} t_{2n} = t_n \\ t_{2n+1} = 1 - t_n \end{cases}$$

pour tout entier n (suite de Prouhet-Thue-Morse).

1. Calculer $t_{2\,016}$.
2. Écrire un programme qui demande l'entier n et retourne la valeur de t_n.
3. La suite (t_n) possède-t-elle trois termes consécutifs indentiques ?
4. Cette suite est-elle périodique ?

3 Un peu de cryptographie

Les lettres et les différents caractères sont codés à l'aide de nombres dans l'ordinateur (codage ASCII) :

```
print( ord("a") )
print( ord("b") )
```

Quelle est l'entier associé à la lettre D ?

Réciproquement à partir d'un nombre, on peut retrouver le caractère associé :

```
print( chr(97) )
```

Exercice 4 (Chiffrement de César)**.**

1. Écrire un programme qui demande une lettre de l'alphabet en minuscule notée L et retourne la lettre cinq rang plus loin dans l'alphabet.
2. Tester le programme avec la lettre x. Que se passe-t-il ? Modifier si besoin le programme.
3. Nous allons utiliser une division euclidienne :

```
def chiffrement_Cesar(L):
  r = ord(L) - ord("a")  # rang de la lettre L dans l'alphabet
  r_image = ( r + 5 ) % 26
  resultat = chr( ord("a") + r_image )
  return resultat

# test:
print( "a --> ", chiffrement_Cesar("a"))
print( "x --> ", chiffrement_Cesar("x"))
```

a) Modifier le programme précédent pour chiffrer avec un décalage de 31 au lieu de cinq.

b) Tester le programme. Les résultats sont ils différents d'avant ? Justifier.

c) Avec le chiffrement de César combien y a-t-il de manières différentes de chiffrer un message ?

d) Compléter le programme suivant pour chiffrer un message entier :

```
message = "en python le chiffrement de cesar est simple"
cryptogramme = ""
for L in message:
  if ord(L) < ord("a") or ord("z") < ord(L): # si le caractère L n'est pas une
      lettre minuscule on ne la crypte pas
    cryptogramme += L
  else:
    ...
print(cryptogramme)
```

e)* Modifier la fonction `chiffrement_Cesar` afin de pouvoir crypter un message arbitraire (contenant des majuscules, des accents, ...).

f) Écrire un programme pour décrypter le cryptogramme.

4 Multiples de 3 et 5

Exercice 5. Si on dresse la liste des entiers naturels multiples de 3 ou 5 et inférieurs strictement à 10, on a 3, 5, 6 et 9. Alors la somme de ces multiples est 23.

Trouver la somme des multiples de 3 ou 5 inférieurs strictement à 1000.

5 Matrice : liste de listes

Comme on peut créer des listes de nombres, on peut aussi créer des listes de listes :

```
M = [ [1,2], [3,4] ]
# pour afficher le second terme de la première liste, on fait:
print(M[0][1])
```

Pour afficher tous les termes, on peut procéder comme suit :

```
def affiche(M):
  s = ""
  for ligne in M:
    for x in ligne:
      s += str(x) + "\t"
    s += "\n"
  print(s)

affiche(M)
```

On se rend compte que l'élément sur la ième ligne et la jème colonne est $M[i-1][j-1]$.

Exercice 6. Une liste de trois listes dont chacune des listes est constituée de trois nombres est appelée une *matrice carrée de taille 3*. Dans cet exercice, nous allons considérer de telles matrices où les valeurs sont uniquement des 0 et des 1. Un exemple :

```
M = [[1,0,0],[0,1,0],[0,0,1]]
affiche(M) #Il faut éventuellement recopier la définition de la fonction affiche() pour qu'
    elle fonctionne à nouveau.
```

1. Tester le programme suivant :

```
if M[0][0] == M[0][1] == M[0][2] == 1:
  test = True
else:
  test = False

print(test)
```

   avec la matrice M précédente et avec la matrice `M = [[1,1,1],[0,1,0],[0,0,0]]`. Que permet-il de faire ?
2. Écrire un programme qui teste si la seconde colonne de la matrice M est constituée de 1 uniquement.
3. Écrire un programme qui teste si une ligne ou une colonne est constituée de 1 uniquement.
4. Écrire un programme qui teste si une des diagonales est constituée de 1 uniquement.
5. Compléter le programme suivant qui permet de tester des trois 1 sont alignés sur une ligne, colonne ou diagonale :

```
def teste(M):
  test = False
  if M[0][0] == M[0][1] == M[0][2] == 1: test = True
  if ...
  ...
  return test
```

6. Le tester avec différentes matrices carrées M de taille 3 constituées de 1 et de 0 uniquement.

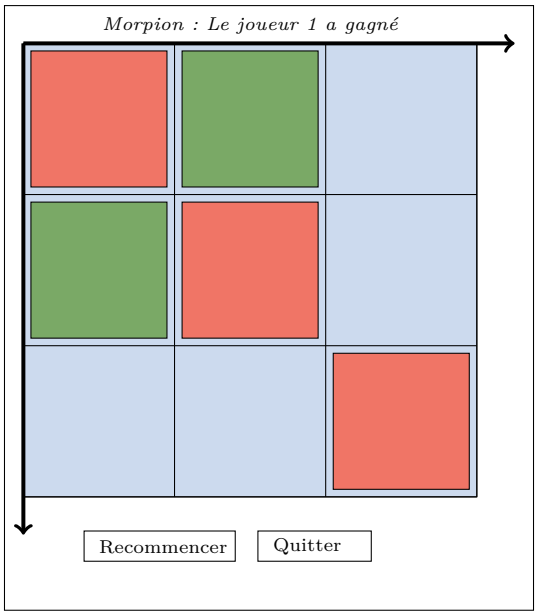
6 Interface graphique Tkinter

Tester le programme suivant :

```
from tkinter import *

# Initialisation de la fenêtre:
Mafenetre = Tk()
Mafenetre.title('Projet: Le Morpion')
Mafenetre.geometry('650x750+20+20')   # largeur fois hauteur + position(x) + position(y)
Mafenetre['bg'] = "gray63"    # changement de la couleur de fond (bg: background)

# Afficher un message (en première ligne de la fenêtre):
message= StringVar()
message.set("Cliquez dans la zone graphique de la fenêtre")
Label(Mafenetre,textvariable=message).pack(padx=1,pady=1)

# Initialisation des dimensions et de la couleur de la fenêtre graphique (nommée Canevas):
Largeur = 600
Hauteur = 600
couleur1 = "cornflower blue"
# Initialisation de la fenêtre graphique Canevas:
Canevas = Canvas(Mafenetre, width = Largeur, height = Hauteur, bg=couleur1)

# Une fonction utilisant la position de la souris qui sera appelée uniquement après un clic gauche
def Clic(event):
  # position du pointeur de la souris
  X = event.x
  Y = event.y
  # affichage des coordonnées dans le terminale:
  print("position du clic: ", X, Y)
  C = 180 # dimension du carré qu'on va tracer
  if 0 < X and X < Largeur and 0 < Y and Y < Largeur:
    # modification du message en première ligne de le fenêtre:
    message.set("début de la partie")
    # utilisation des coordonnées X et Y:
    if X + C/2 < Largeur/3:
      Carre = Canevas.create_rectangle(X-C/2,Y-C/2,X+C/2,Y+C/2,fill="red3")
    elif Largeur/3 < X - C/2:
      Carre = Canevas.create_rectangle(X-C/2,Y-C/2,X+C/2,Y+C/2,fill="green4")

# Ppour débuter le jeu ou recommencer, il faut (re)initialiser certaines données:
def initialisation():
  # On remplit la fenêtre graphique avec la couleur1:
  Canevas.create_rectangle(0,0,Largeur, Hauteur, fill=couleur1)
  # On trace une ligne verticale:
  Canevas.create_line(Largeur/3,0,Largeur/3,Hauteur, fill="black")

initialisation()

# Utilisation de bind() pour faire le lien entre l'événement: clic gauche et la fonction: Clic():
Canevas.bind("<Button-1>",Clic)

# Création d'un bouton pour quitter l'application:
BoutonQuitter = Button(Mafenetre, text='Quitter',  fg = 'red', command = Mafenetre.destroy)

# Création d'un bouton pour recommencer (à l'aide de la fonction initialisation()):
BoutonRecommencer= Button(Mafenetre, text='Recommencer',  fg = 'red', command = initialisation)

#Compilation:
Canevas.focus_set()
Canevas.pack(padx = 10, pady = 10);
BoutonRecommencer.pack(side = LEFT , padx = 10, pady = 10)
```

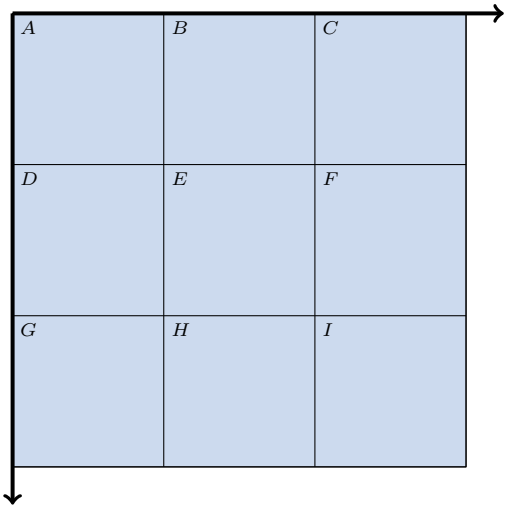
```
60 BoutonQuitter.pack(side = LEFT , padx = 10, pady = 10)
61 Mafenetre.mainloop()
```

Exercice 7. À l'aide de cet expérience, coder le jeu le Morpion :



Voici quelques indications pour mieux avancer dans ce projet. *Les références à des lignes précises du code font toujours référence au code sur la page précédente (sans aucune modification).*

1. En vous inspirant de la ligne 43 du code, dessiner les 3 autres lignes manquantes pour afficher une grille avec 9 cases.
2. Quelles sont les dimensions d'un carreau ?
3. Déterminer les coordonnées des points A, B, ... et I :

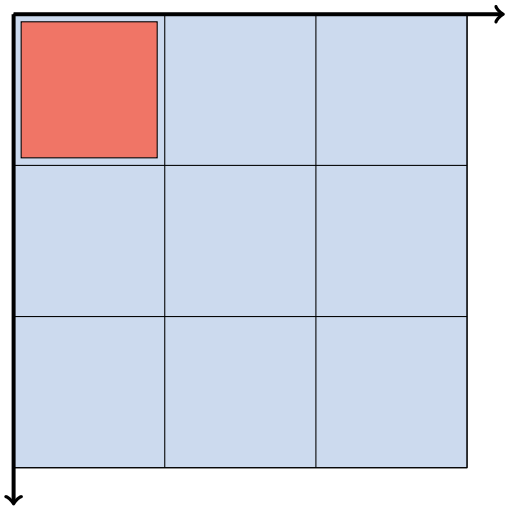


4. Ajouter les lignes suivantes à la ligne 36 du code :

```
1   i = X // 300
2   j = Y // 300
3   x = 300*i
4   y = 300*j
5   print("x = ", x, " et y = ", y)
```

   Le tester en cliquant dans différentes zones de la fenêtre.

5. En déduire le code à ajouter pour afficher un carré centré de côté 180 dans la case où l'on vient de cliquer :



6. Gestion de deux joueurs (couleur rouge et couleur verte) :
   Ajouter la ligne suivante à la ligne 17 :

```
joueur = 1 # le joueur 1 commence
```

   Remplacer les lignes 33 à 36 par :

```
  if joueur == 1:
    # si c'est au joueur 1, on trace le carré en rouge:
    ...
    joueur = 2 # on passe la main au joueur 2
  else:
    # si c'est au joueur 1, on trace la carré en vert:
    ...
    joueur = 1 # on passe la main au joueur 1
```

   Puis le compléter...

7. En utilisant la commande `message.set(" ... ")` comme à la ligne 33, compléter le programme pour qu'il affiche "joueur 1" lorsque c'est au joueur 1 de jouer et "joueur 2" dans l'autre cas.
8. Ajouter une matrice $M = [[0,0,0],[0,0,0],[0,0,0]]$ en début de programme, puis dans la fonction clic(), utilisez les variables i et j pour modifier la matrice M de telle sorte que :

$$M[i][j] = \begin{cases} 0 & \text{si la case est vide} \\ 1 & \text{si la case a été joué par le joueur 1} \\ 2 & \text{si la case a été joué par le joueur 2} \end{cases}$$

9. Modifier pour pouvoir utiliser la fonction `test(M)` crée en fin de section précédente pour pouvoir test si le joueur `joueur` a coché trois cases alignés.
10. En cas de victoire, afficher avec la commande `message.set("...")`, la joueur `joueur` a gagné.