

Apprendre à programmer avec Python

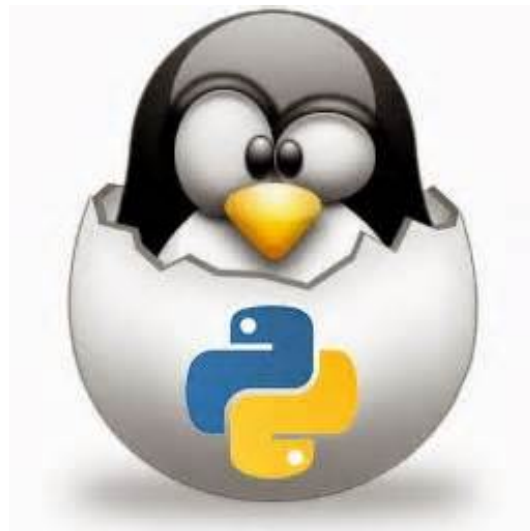


Table des matières

I	Dialoguer avec la machine	5
I.1	Afficher un message : La fonction <code>print</code>	5
I.2	Questionner l'utilisateur : La fonction <code>input</code>	5
II	Variables et opérations	6
II.1	Super calculatrice	6
II.2	Les variables numériques	6
III	Tests et indentation	8
III.1	Les tests <code>if</code> et <code>else</code>	8
III.2	Emboîtement de blocs	9
IV	Les boucles <code>while</code> et <code>for</code>	10
IV.1	Un problème	10
IV.2	Une solution : la boucle <code>while</code>	10
IV.3	Un autre exemple	11
IV.4	La boucle <code>for</code>	11
V	Les tableaux et les listes	13
V.1	Quelques opérations sur les listes :	13
V.2	Opérations sur les chaînes de caractères	15
V.3	Transformer une chaîne de caractères en une liste :	15
V.4	Transformer une liste en une chaîne de caractères :	16
V.5	Transformer un nombre en une liste de chiffres	16
V.6	Transformer une liste de chiffres en un nombre	17
VI	Définir et appeler une fonction	19
VII	Dessiner avec <code>TKinter</code>	21

Un avant-goût

Avant de commencer, copiez le dossier *python* situé dans *commun/travail* afin de le coller dans vos documents.

Un premier jeu!

Voici le contenu du programme `guessv4.py` :

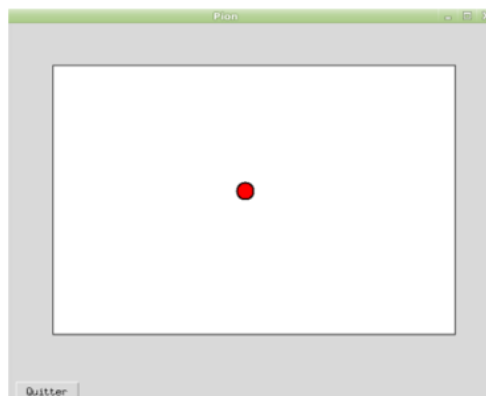
```
import random  #On importe un module complémentaire

#Le programme commence ici!

r = random.randint(1,20)  #r est un nombre entier aleatoire compris entre 1 et 20
i=1
Proposition=int(input("Trouve le nombre magique :"))
while i<3 :
    if Proposition < r :
        print("Le nombre magique est plus petit !")
    if Proposition > r :
        print("Le nombre magique est plus grand !")
    if Proposition == r :
        print("Congratulation !")
        i=2
    Proposition=int(input("Trouve le nombre magique :"))
    i=i+1
if Proposition != r:
    print("You loose !, Le nombre magique etait : ", r)
```

- ☞ Ouvrez l'éditeur *pyzo*.
- ☞ Ouvrez ce programme : */media/Poste de travail/Mon travail/Python*
- ☞ Exécutez ce programme. (*Run / Run file ou Ctrl+Enter*)
- ☞ Il y a des erreurs dans ce code, corrigez-les.

Move the pawn!



Voici le contenu du programme pawn.py :

```
from tkinter import *

def Clavier(event):
    globalPosX,PosY
    touche = event.keysym
    if touche == 'Right':
        PosY -= 20
    if touche == 'Up':
        PosY += 20
    if touche == 'Left':
        PosX += 20
    if touche == 'Down':
        PosX -= 20
    Canevas.coords(Pion,PosX -10, PosY -10, PosX +10, PosY +10)

Mafenetre = Tk()
Mafenetre.title('Pion')

PosX = 230
PosY = 150

Largeur = 480
Hauteur = 320
Canevas = Canvas(Mafenetre, width = Largeur, height =Hauteur, bg ='white')
Pion = Canevas.create_oval(PosX-10,PosY-10,PosX+10,PosY+10,width=2,outline=
    'black',fill='red')
Canevas.focus_set()
Canevas.bind('<Key>',Clavier)
Canevas.pack(padx =50, pady =50)

Button(Mafenetre, text ='Quitter', command = Mafenetre.destroy).pack(side=
    LEFT,padx=5,pady=5)

Mafenetre.mainloop()
```

- ☞ Ouvrez ce programme : */media/Poste de travail/Mon travail/Python*
- ☞ Exécutez ce programme. (*Run / Run file ou Ctrl+Enter*)
- ☞ Il y a des erreurs dans ce code, corrigez-les.

Premiers pas en Python

I Dialoguer avec la machine

I.1 Afficher un message : La fonction `print`

[Initiation interactive](#)

- ☞ Ouvrez l'éditeur `pyzo`.
- ☞ Dans la fenêtre de droite, recherchez le dossier `python` dans vos documents.
- ☞ Éditez le fichier `dialogue1.py`.
- ☞ Exécutez-le. (*Run / Run file ou Ctrl+Entrée*)

Voici le contenu du programme `dialogue1.py` :

```
print("Hello !")
Nom = "Nadia"
print("Comment vas-tu", Nom, "?")
```

Le résultat :

```
Hello !
Comment vas-tu Nadia ?
```

- ☞ Modifiez le code, changez les phrases...
- ☞ À quoi sert la fonction `print` ?

I.2 Questionner l'utilisateur : La fonction `input`

[Initiation interactive](#)

Voici le contenu du programme `dialogue2.py` :

```
nom = input("Quel est ton nom ? ")
print("Bonjour", nom)
print(nom, "est un joli nom !")
print("Passe une bonne journee Nadia")
```

Le résultat :

```
Quel est ton nom ? Nadia
Bonjour Nadia
Nadia est un joli nom !
Passe une bonne journee Nadia
```

- ☞ Exécutez ce programme, changez de prénom...
- ☞ À quoi sert la fonction `input` ?
- ☞ Il y a une erreur dans le code, corrigez-la.
- ☞ Modifiez le code en ajoutant des questions et des réponses.

II Variables et opérations

II.1 Super calculatrice

[Initiation interactive](#)

Voici le contenu du programme exo4.py :

```
a=7
b=2
print("Regardez les resultats suivants :")
print(a + b * 2, a / b, a // b, a % b, a ** b)
```

Le résultat :

```
Regardez les resultats suivants :
11 3.5 3 1 49
```

- ☞ Éditez ce programme, changez les nombres a et b et essayez de comprendre les opérations.
- ☞ Le langage Python respecte-t-il les priorités ?
- ☞ Que font les opérations : /, //, % et ** ?

- ☞ Vous avez 75 bonbons à vous partager entre élèves. Le reste sera pour nous.
À l'aide des opérations // et %, calculez le nombre de bonbons que chaque élève va recevoir ainsi que nombre de bonbons pour moi.

II.2 Les variables numériques

[Initiation interactive](#)

Voici le contenu du programme variables2.py :

```
a = input("Donnez un premier nombre : ")
b = input("Donnez un autre nombre : ")
print("La somme de ", a, "et de", b, "est : ", a + b)
```

Le résultat :

```
Donnez un premier nombre : 17
Donnez un autre nombre : 16
La somme de 17 et de 16 est : 1716
```

- ☞ Exécutez ce programme. Changez les nombres. Changez l'opération. Quel est le problème ?
- ☞ Donnez des mots à la place des nombres lors de l'exécution. Avez-vous une explication ?

Les variables peuvent avoir des types différents (des nombres, des chaînes de caractères...).

La fonction `input()` renvoie des chaînes de caractères : 'String'.

Pour transformer une chaîne de caractères en nombre entier on utilise la fonction `int()` ('Integer').



Voici une solution : exo10.py :

```
a = int(input("Donnez un premier nombre : "))
b = int(input("Donnez un autre nombre : "))
print("La somme de ", a, "et de", b, "est : ", a + b)
```

Le résultat :

```
Donnez un premier nombre : 17
Donnez un autre nombre : 16
La somme de 17 et de 16 est : 33
```

À vous de jouer !

LEVEL 1 :

Écrivez un programme qui demande trois nombres entiers puis renvoie leur somme et leur produit.

=====

LEVEL 2 :

Écrivez un programme qui demande un nombre entier puis qui renvoie sa table de multiplication jusqu'à 9.

Voici un exemple de résultat :

(On a entré le nombre 7.)

```
Donnez un nombre : 7
7 x 0 = 0
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
```

=====

LEVEL 3 :

Écrivez un programme qui :

- ☞ demande un nombre entier
- ☞ affiche ce nombre et les deux nombres qui le suivent
- ☞ affiche le résultat de la somme de ces trois nombres
- ☞ affiche le quotient de cette somme par 3.

```
Donnez un nombre : 34
34 35 36
34 + 35 + 36 = 105
Cette somme divisee par 3 : 35.0
```

Que pouvez-vous observer ?

=====

Pydéfis

III Tests et indentation

III.1 Les tests if et else


[Initiation interactive](#)

Voici le contenu du programme exo11.py :

```
a = int(input("Premier nombre : "))
b = int(input("Deuxieme nombre : "))
o = input("Operation ? (entrez * ou +) : ")
if o == "*":
    c = a * b
else:
    c = a + b
print("Le resultat est ",c)
```

Le résultat :

```
Premier nombre : 5
Deuxieme nombre : 6
Operation ? (entrez * ou +) : *
Le resultat est 30
```

- ☞ Testez ce programme, changez les nombres et les opérations.
- ☞ Que se passe-t-il si vous entrez autre chose que '*' ou '+' pour le choix de l'opération ?
- ☞ Corrigez ce problème à l'aide d'un premier test.
- ☞ À l'aide de la touche tabulation  modifiez la dernière ligne du code comme ceci :

```
a = int(input("Premier nombre : "))
b = int(input("Deuxieme nombre : "))
o = input("Operation ? (entrez * ou +) : ")
if o == "*":
    c = a * b
else:
    c = a + b
print("Le resultat est ",c)
```

- ☞ Testez plusieurs fois ce programme. Qu'observez-vous ? Expliquez pourquoi.
- ☞ Modifiez le programme afin qu'il affiche : "La somme des deux nombres est : ..." ou "Le produit des deux nombres est : ..." en fonction de l'opération choisie.

Après un if, le bloc à exécuter, si le test est vrai, doit être décalé vers la droite !

Il ne faut pas oublier les ':' qui suivent le test.

Pour sortir d'un bloc, il faut arrêter le décalage.

Pour tester une égalité, il faut utiliser '=='.



III.2 Emboîtement de blocs

Voici le contenu du programme test1.py :

```
a = int(input("Donnez un premier nombre : "))
b = int(input("Donnez un autre nombre : "))
if a < b :
    print("Le plus grand des deux nombres est ", b)
else :
    print("Le plus grand des deux nombres est ", a)
```

Le résultat :

```
Donnez un premier nombre : 7
Donnez un autre nombre : 13
Le plus grand des deux nombres est 13
```

- ☞ Testez ce programme, changez les nombres.
- ☞ Que se passe-t-il si les deux nombres entrés sont égaux? Expliquez pourquoi.

- ☞ Modifiez ce code afin qu'il teste si les deux nombres sont égaux puis, dans le cas contraire, qu'il renvoie le plus grand des deux.

À vous de jouer !

LEVEL 1 :

Écrivez un programme qui demande trois nombres puis qui renvoie le plus grand des trois.

=====

LEVEL 2 :

Écrivez un programme qui :

- ☞ demande deux nombres Donnez un nombre : 7
- ☞ teste si l'un est multiple de l'autre Donner un autre nombre : 42
- ☞ renvoie la conclusion du test. 42 est un multiple de 7.

(On a entré les nombres 7 et 42.)

(Indication : Testez les opérations $27 \% 9$; $27 \% 10$; $27 \% 3$; $27 \% 4$)

=====

IV Les boucles `while` et `for`

IV.1 Un problème

Voici le contenu du programme `boucle1.py` :

```
i=0
print(i)
i=i+1
print(i)
i=i+1
print(i)
i=i+1
print(i)
i=i+1
print(i)
```

Le résultat :

```
1
2
3
4
```

- ☞ Que fait ce programme ?
- ☞ Que veut dire la ligne `i=i+1` ?
- ☞ Modifiez le code afin d'afficher les 10 premiers nombres entiers.
- ☞ Affichez les 100 ? les 1000 premiers nombres ?

IV.2 Une solution : la boucle `while`

[Initiation interactive](#)

Voici le contenu du programme `boucle2.py` :

```
i=0
while i<5:
    print(i)
    i=i+1
```

- ☞ Exécutez ce programme. Quelle est la traduction de `while` ?
- ☞ Quelle est la valeur de `i` à la fin de l'exécution ?
- ☞ Modifiez le code afin d'afficher les 100 premiers nombres entiers.



Dans une boucle `while`, il ne faut pas oublier d'initialiser la variable utilisée (ici `i=0`).
La variable doit être incrémentée pour avancer dans la boucle (ici `i=i+1`).
Le contenu de la boucle doit être décalé d'une tabulation vers la droite.

IV.3 Un autre exemple

Voici le contenu du programme exo13.py :

```
i=1
while i<=100 :
    print(i)
    i=i*2
```

Le résultat :

```
1
2
4
8
16
32
64
128
```

- ☞ Que fait ce programme ?
- ☞ Que veut dire la ligne `i=i*2` ?
- ☞ Quelle est la valeur de `i` à la fin de l'exécution ?
- ☞ Remplacez les nombres 1 et 128 dans les deux premières lignes par des valeurs de votre choix.

IV.4 La boucle for

[Initiation interactive](#)

Voici le contenu du programme exo14.py :

```
for i in range(0,8) :
    print(2**i)
```

- ☞ Exécutez ce programme. Que pouvez-vous observer ?
- ☞ Quelle est la valeur de 2^7 ? de 2^8 ?
- ☞ Modifiez ce programme pour afficher les puissances successives de 3.

Voici le contenu du programme for1.py :



```
Mot="Pyzo"
for c in Mot:
    print(c)
```

- ☞ Exécutez ce programme. Que pouvez-vous observer ?
- ☞ Changez le mot et testez ce programme.

À vous de jouer !

LEVEL 1 :

Écrivez un programme qui demande un nombre puis renvoie les puissances de ce nombre de 0 à 20.

=====

LEVEL 2 :

Écrivez un programme qui demande un nombre entier puis qui renvoie sa table de multiplication jusqu'à 9.

Utilisez une boucle !

Voici un exemple de résultat :

(On a entré le nombre 7.)

Donnez un nombre : 7

$$7 \times 0 = 0$$

$$7 \times 1 = 7$$

$$7 \times 2 = 14$$

$$7 \times 3 = 21$$

$$7 \times 4 = 28$$

$$7 \times 5 = 35$$

$$7 \times 6 = 42$$

$$7 \times 7 = 49$$

$$7 \times 8 = 56$$

$$7 \times 9 = 63$$

=====

LEVEL 3 :

Soit n un nombre entier positif.

On appelle **factorielle** de n , notée $n!$ le résultat du produit :

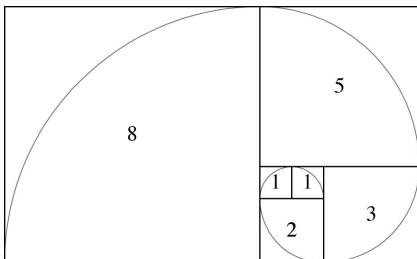
$$n! = 1 \times 2 \times 3 \times \dots \times (n-1) \times n.$$

Par exemple : $4! = 1 \times 2 \times 3 \times 4 = 24$.

À l'aide d'une boucle, calculez $23!$.

=====

LEVEL 4 :



Voici les sept premiers termes de la suite de **Fibonacci** :
0, 1, 1, 2, 3, 5, 8...

Chaque terme est la somme des deux termes qui le précède.

Par exemple, le huitième terme sera le résultat de $5 + 8$, soit 13.

Quel sera le centième terme ?

=====

[Plus de défis sur PyDéfis](#)

V Les tableaux et les listes

V.1 Quelques opérations sur les listes :

[Initiation interactive](#)

Créer une liste :

```
MaListe = [1,4,3,7,5,4]
print("Voici une liste :", MaListe)
```

Le résultat :

```
Voici une liste : [1, 4, 3, 7, 5, 4]
```

Extraire un élément d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Prendre un element d'une liste : (Le premier element est indexe par 0)
print("Le troisieme element de ma liste est :", MaListe[2])
```

Le résultat :

```
Le troisieme element de ma liste est : 3
```

Retourner l'index d'un élément d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Retourner l'index d'un element avec la methode .index() :
Index=MaListe.index(7)
print(Index)
```

Le résultat :

```
3
```

Retourner la longueur d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Calculer sa longueur avec la fonction len() :
Longueur = len(MaListe)
print("La longueur de ma liste est : ",Longueur)
```

Le résultat :

```
La longueur de ma liste est : 6
```

Ajouter un élément à la fin d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Ajouter un element a la fin avec la methode .append(element) :
MaListe.append(9)
print(MaListe)
```

```
[1, 4, 3, 7, 5, 4, 9]
```

Ajouter une liste à une liste :

```
MaListe = [1,4,3,7,5,4]
#Ajouter une liste a une liste avec la methode .extend(liste) :
MaListe.extend([2,5,4])
print(MaListe)
```

```
[1, 4, 3, 7, 5, 4, 2, 5, 4]
```

Retirer un élément d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Enlever une valeur d'une liste avec la methode .remove(element) :
MaListe.remove(4)
print(MaListe)
```

```
[1, 3, 7, 5, 4]
```

Retourner la valeur maximum d'une liste :

```
MaListe = [1,4,3,7,5,4]
#Prendre la valeur maximum d'une liste avec max(liste) :
LeMax = max(MaListe)
print("Le maximum est :", LeMax)
```

```
Le maximum est : 7
```

Trier par ordre croissant une liste :

```
MaListe = [1,4,3,7,5,4]
#Trier les elements d'une liste avec la methode .sort() :
MaListe.sort()
print(MaListe)
```

```
[1, 3, 4, 4, 5, 7]
```

Inverser une liste :

```
MaListe = [1,4,3,7,5,4]
#Inverser une liste avec la methode .reverse() :
MaListe.reverse()
print(MaListe)
```

```
[7, 5, 4, 4, 3, 1]
```

Compter le nombre d'apparitions d'un élément d'une liste :

```
MaListe=[1,4,3,7,5,4]
#Compter le nombre d'apparition d'un element d'une liste
# avec la methode .count(element) :
Nombre = MaListe.count(4)
print("Le nombre de 4 dans la liste est : ", Nombre)
```

```
Le nombre de 4 dans la liste est : 2
```

V.2 Opérations sur les chaînes de caractères

Certaines des méthodes précédentes fonctionnent aussi sur des chaînes de caractères.

- ☞ Faites fonctionner toutes ces méthodes sur une chaîne.
(Utiliser la chaîne "Hello Ana 452" par exemple.)
- ☞ Indiquez les méthodes qui fonctionnent avec les chaînes.



Il est possible de transformer une chaîne de caractères en une liste afin de lui appliquer des méthodes supplémentaires.

V.3 Transformer une chaîne de caractères en une liste :

Transformer une chaîne en une liste pour la trier par exemple :

Voici une fonction à retenir! list()

```
MonMot="Hello Ana 452"
# Transformer une chaîne en une liste de caracteres :
MaListe=list(MonMot)
# On peut maintenant trier cette liste :
MaListe.sort()
print(MaListe)
```



Le résultat :

```
[' ', ' ', ' ', '2', '4', '5', 'A', 'H', 'a', 'e', 'l', 'l', 'n', 'o']
```

- ☞ De quelle façon la méthode sort() classe-t-elle les caractères?

V.4 Transformer une liste en une chaîne de caractères :



```
MaListe=['B', 'y', 'e', ' ', 'A', 'n', 'a']
MaPhrase="" #On creer une chaine vide au depart
#On creer une boucle qui va ajouter chaque element de la liste a la chaine :
i=0
while i<len(MaListe): #Tant que i ne depasse pas la longueur de "MaListe"
    Lettre=MaListe[i] #On extrait la lettre d'index "i"
    MaPhrase= MaPhrase + Lettre #On ajoute la lettre a "MaPhrase"
    print(MaPhrase) #On affiche le resultat etape par etape
    i=i+1
```

Le résultat :

```
B
By
Bye
Bye
Bye A
Bye An
Bye Ana
```

☞ Modifiez simplement le code pour afficher seulement "Hello Ana !"

☞ Essayez ceci :



```
MaListe=['B', 'y', 'e', ' ', 'A', 'n', 'a']
MaPhrase="".join(MaListe)
print(MaPhrase)
```

V.5 Transformer un nombre en une liste de chiffres

Une fonction utile :



```
n=7658 #Voici un nombre
Liste=[] #On initialise une liste (vide)
n=str(n) #On transforme le nombre en chaine de caracteres

i=0
while i<len(n): #Tant que i ne depasse pas la longueur de "n"
    Chiffre=int(n[i]) #On extrait le chiffre d'index "i", ne pas oublier int()
    Liste.append(Chiffre) #On ajoute le Chiffre a la fin de "Liste"
    i=i+1
print(Liste) #On affiche le resultat
```

Le résultat :

```
[7, 6, 5, 8]
```

☞ Il y a plus simple ! Modifiez ce code pour utiliser une boucle for.

Une vision mathématiques :

```
n=7658 #Voici un nombre
Liste=[] #On initialise une liste (vide)

while n>0: #Tant qu'il reste des chiffres a extraire dans "n"
    Chiffre=n%10 #On extrait le dernier chiffre de "n"
    n=n//10 #On Retire le dernier chiffre de "n"
    Liste.append(Chiffre) #On ajoute le Chiffre a la fin de "Liste"
print(Liste) #On affiche le resultat etape par etape
```

Le résultat :

```
[8]
[8, 5]
[8, 5, 6]
[8, 5, 6, 7]
```

- ☞ Analysez bien ce code.
- ☞ Avez-vous remarqué le "problème" dans le résultat ?
- ☞ Modifiez le code afin que le programme renvoie simplement [7, 6, 5, 8].
(Utilisez la méthode `reverse()` ou `insert(0,Chiffre)`)

V.6 Transformer une liste de chiffres en un nombre

```
MaListe=[1,4,3,7,5,4]
MonNombre=""

i=0
while i<len(MaListe):
    Chiffre=MaListe[i]
    MonNombre=MonNombre+str(Chiffre)
    i=i+1
print("Le nombre est ",int(MonNombre))
```

Ou encore :

```
MaListe=[1,4,3,7,5,4]
MonNombre=""

for c in MaListe:
    MonNombre=MonNombre + str(c)
print("Le nombre est ", int(MonNombre))
```



Le résultat :

```
Le nombre est 143754
```

- ☞ Commentez ces codes afin d'expliquer les étapes.

À vous de jouer !

LEVEL 1 :

Écrivez un programme qui demande quatre nombres puis les renvoie du plus grand au plus petit.

=====

LEVEL 2 :

Voici une phrase étrange : "6Q9994u1e2l997le9 39h9e99u9r99e0 e999s9t99-9999i999l99 999?"

Écrivez un programme qui renvoie la phrase épurée des chiffres.

(Indications : Pensez à utiliser les listes et les méthodes `remove()` et `count()`)

=====

LEVEL 3 :

Écrivez un programme qui demande un nombre puis renvoie le plus grand nombre possible composé des chiffres du nombre de départ.

(Par exemple, si le nombre de départ est 586 alors le programme doit renvoyer 865.)

=====

LEVEL 4 :

Le théorème de Pythagore :

« Si dans un triangle, le carré du plus grand côté est égal à la somme des carrés des deux autres côtés alors ce triangle est rectangle. »

Pour faire simple :

⇒ Un triangle dont les côtés mesurent 3cm, 4cm et 5cm est rectangle. En effet :

$$3^2 + 4^2 = 9 + 16 = 25 \quad \text{et} \quad 5^2 = 25.$$

⇒ Un triangle dont les côtés mesurent 4cm, 5cm et 6cm **n'est pas** rectangle. En effet :

$$4^2 + 5^2 = 16 + 25 = 41 \quad \text{et} \quad 6^2 = 36.$$

☞ Un triangle dont les côtés mesurent 5cm, 13cm et 12cm est-il rectangle ?

☞ Écrivez un programme qui demande trois longueurs entières puis renvoie :
« Ce triangle est (ou n'est pas) rectangle » en fonction.

=====

Pydéfis : Opérations sur les collections

VI Définir et appeler une fonction

Voici un exemple très utile :



```
#Voici une fonction
def NombreVersListe(Nombre1):
    Nombre1=str(Nombre1)
    Liste1=[]
    for c in Nombre1 :
        Liste1.append(int(c))
    return Liste1

#Voici une autre fonction :
def ListeVersNombre(Liste2):
    Nombre2=""
    for c in Liste2:
        Nombre2=Nombre2 + str(c)
    return int(Nombre2)

#Program start here!

Nombre=56746
Liste=NombreVersListe(Nombre)
print(Liste)
NombreDeDepart=ListeVersNombre(Liste)
print(NombreDeDepart)
```

Le résultat :

```
[5, 6, 7, 4, 6]
56746
```

☞ Que font ces deux fonctions ?

☞ Quel est l'intérêt de définir et d'appeler des fonctions ?

À vous de jouer !

LEVEL 1 :

Écrivez un programme qui demande un nombre puis renvoie le plus grand nombre possible composé des chiffres du nombre de départ.

(Par exemple, si le nombre de départ est 586 alors le programme doit renvoyer 865.)

=====

LEVEL 2 :

Un nombre entier est un **palindrome** si ses écritures de gauche à droite et de droite à gauche sont identiques.

Par exemple, 34843 est un palindrome.

Dans cet exercice, on veut créer un programme qui, à partir d'un nombre entier, renvoie si ce nombre est un palindrome ou non.

Pour cela, il faut donc :

- ⇒ Transformer le nombre en liste de chiffres
 - ⇒ Inverser cette liste
 - ⇒ Transformer cette nouvelle liste en nombre
 - ⇒ Tester si ce nouveau nombre est égal au nombre de départ.
- =====

LEVEL 3 :

Voici le contenu du programme whitejack.py :

```
import random #On importe un module complémentaire

def Carte(Score) :
    r = random.randint(1,10)
    Score = Score + r
    return Score

Score = 0
Banque = random.randint(12,21)
Question = "y"
while Question == "y" :
    Question = input("Voulez vous une carte supplémentaire ? (y or n) :")
    if Question == "y":
        Score = Carte(Score)
        print(Score)

print("Score de la banque : ", Banque)
if Banque < Score and Score < 22 :
    print("You win !!")
else :
    print("You loose !!")
```

☞ Sans exécuter ce programme, écrivez les règles du jeu.

=====

VII Dessiner avec Tkinter

Voici le contenu du programme graph1.py :

```
# On importe Tkinter
from tkinter import *

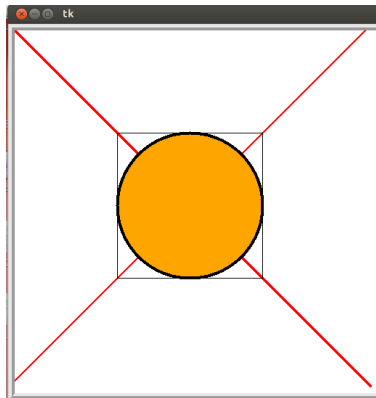
# On cree une fenetre, racine de notre interface
Fenetre = Tk()

# Dans Fenetre nous allons creer un objet type Canvas qui se nomme zone_dessin
# Nous donnons des valeurs aux proprietes "width", "height", "bg", "bd"
zone_dessin = Canvas(Fenetre, width=500, height=500, bg='white', bd=8)
zone_dessin.pack() #Affiche le Canvas

# Nous allons maintenant utiliser quelques methodes du widget "zone_dessin"
zone_dessin.create_line(0,0,500,500,fill='red',width=4) # Dessine une ligne
zone_dessin.create_line(0,500,500,0,fill='red',width=2) # Dessine une ligne
zone_dessin.create_rectangle(150,150,350,350) # Dessine un rectangle
zone_dessin.create_oval(150,150,350,350,fill='orange',width=4) # Dessine un
cercle

# On démarre la boucle Tkinter qui s'interrompt quand on ferme la fenetre
Fenetre.mainloop()
```

Résultat :



- ☞ Testez ce programme, changez les coordonnées et les couleurs.
- ☞ Sur le résultat, placez les points $A(0;0)$, $B(0;500)$, $C(500;0)$, $D(500;500)$ et $E(150;150)$.