# Python : Un premier jeu

1. Une première interface :

Les commandes suivantes permettent de créer une fenêtre pour afficher notre jeu :

```
from tkinter import *
fenetre=Tk()
fenetre.title("Jeu Python")
n=15
c=50
can=Canvas(fenetre,width=n*c,height=n*c, bg="black")
can.pack()
```

1. Comment changer la couleur du fond ?

2. Comment changer la taille de la fenêtre ?

2. Le personnage :

Pour se faciliter la tâche, on représente notre personnage par une forme géométrique simple,

La fonction suivante permet de créer un rectangle :

```
can.create_rectangle(0,0,50,50,fill="red")
```

Ici le coin supérieur gauche du rectangle est aux coordonnées 0,0 et le coin inférieur droit aux coordonnées 50,50. De même pour un cercle :

```
can.create_oval(50,50,100,100,fill="red")
```

Nous pouvons aussi afficher du texte avec la commande :

can.create_text(100,100,text="Bonsoir",fill="blue")

Et enfin des images :

- Au début du fichier :

img = PhotoImage(file= ‘image.gif’)

- Dans notre code :

can.create_image(300, 300, image=img)

3. Le déplacement :

La commande suivante permet de lancer la fonction Test si on appuie sur la touche « d » :

fenetre.bind("<d>", Test)

Les fonctions Test, la fonction rondIn et la ligne ci-dessous permettent de gérer le déplacement de notre personnage vers la droite.

```
def Test(event = None):
global pos
pos[0]+=50
rondIn()
```

```
def rondIn():
global c,posx,posy,Mt
can.delete(ALL)
rond = can.create_oval(pos[0],pos[1],pos[0]+c,pos[1]+c,fill='#fff')
```

1. recopiez ces deux fonctions en bas de votre code ainsi que la commande fenetre.bind ci-dessus et l’instruction suivante en haut de votre code :

pos=[0,0]

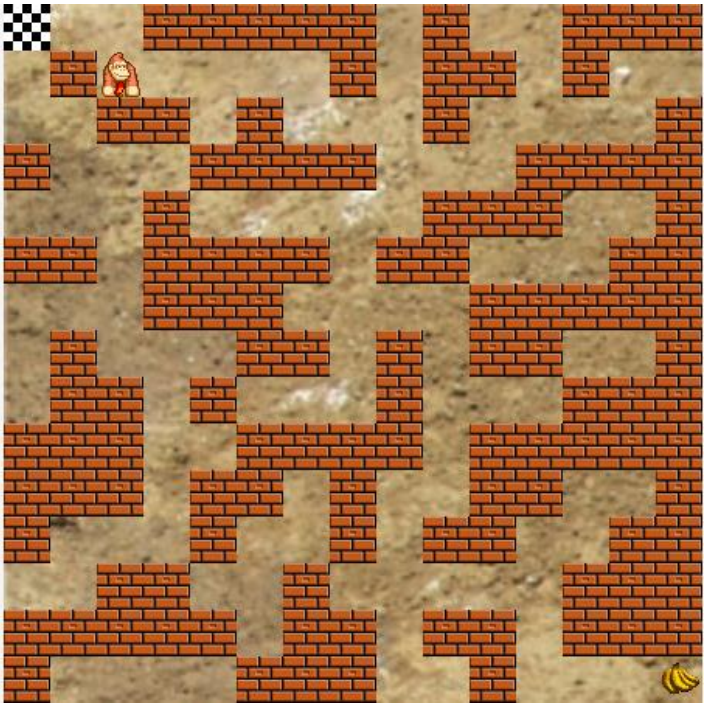
2. Maintenant que vous savez gérer le déplacement vers la droite, faites de même pour la gauche, le haut, et le bas.

# Les matrices

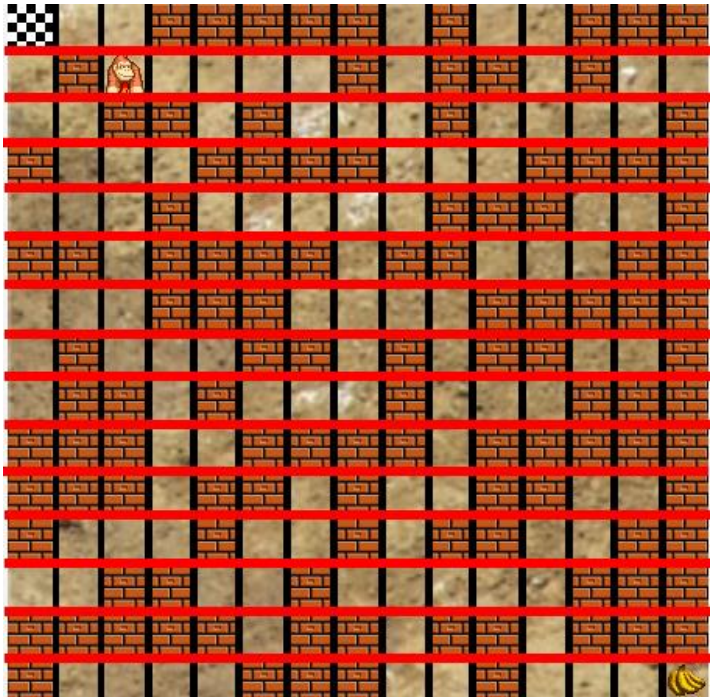
En Python, une manière de représenter les données d'une surface plane divisée en « petit carré » se fait à l'aide d'un objet qu'on appelle « matrice ».

Pour saisir un peu plus précisément l'idée…

Voici un plateau de jeu représentant un labyrinthe :

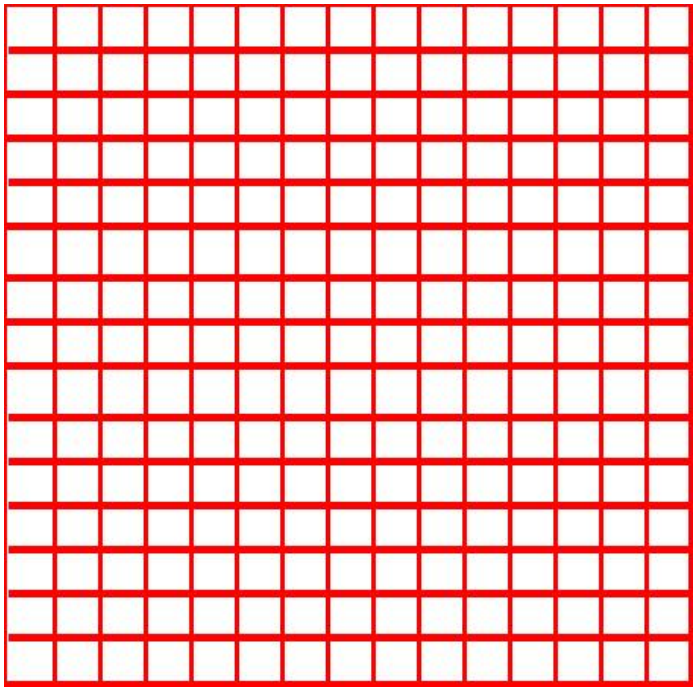


On peut voir que le plateau de jeu peut être divisé en plusieurs petits carrés de même taille, formant une grille de 15 carrés par 15 carrés.



Mais qu'est ce qui se cache derrière l'interface graphique ?

En réalité, il s'agit d'un tableau à deux dimensions, comme suit :



On appelle ça une **matrice.**

Une matrice en Python est représentée par une **liste de listes d'éléments.**

Une liste s'écrit nom_de_la_liste = [ élément , élément , élemen t]
Par exemple, la liste 1 2 3 s'écrit exemple_liste = [1,2,3]

Ainsi, on peut écrire la matrice exemple_matrice = [ exemple_liste , exemple_liste ] qui correspondra à [ [1 , 2 , 3] , [ 1 , 2 , 3 ] ]

Par exemple, une matrice de 3 cases sur 3 cases est une liste contenant 3 listes de 3 éléments.
Matrice = [ [1,1,1], [2,2,2], [3,3,3] ] n'est autre que :

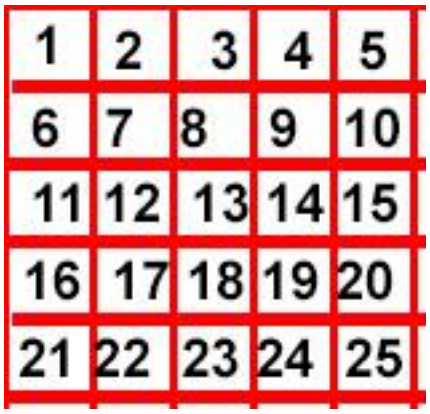
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |

# À faire :

- Ouvrir le fichier ex-matrice.py avec Python IDLE. Celui-ci contiendra une fonction afficher à **ne pas effacer**.
- Lancer le fichier en appuyant sur la touche F5. Quel résultat observe-t-on dans la console ?

- Créer une variable T2 qui sera une matrice de 5 cases sur 5 cases, représentée par l'image suivante, et l'afficher :

T2 =

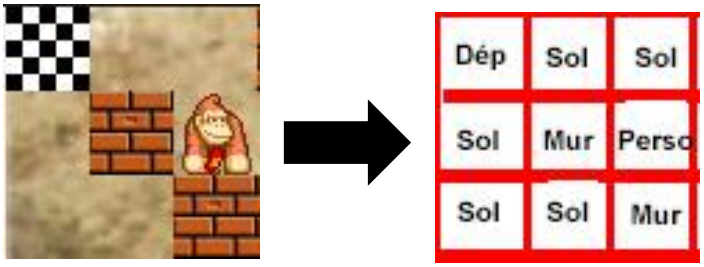



L'accès à un élément d'une matrice se fait de la manière suivante :

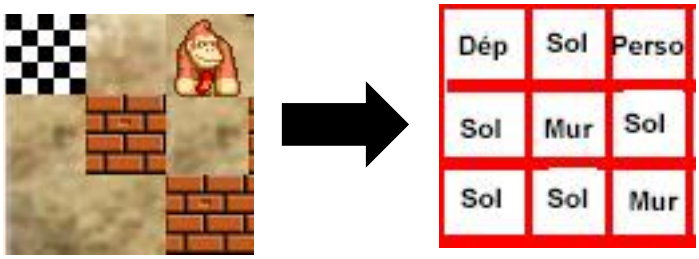
Nom_de_la_matrice [ indice_de_ligne ] [indice_de_colonne ]
Par exemple, sur la matrice T2, si je veux afficher la valeur qui se trouve dans la 2ème ligne, 3ème colonne, il faut que j'effectue la commande :
**print(T2[1,2])**

- Mais pourquoi T2[1,2] et pas T2[2,3] ?
  Essayez **print(T2[0,0])** et expliquez-le.
- Quelle commande pour afficher le chiffre 23 dans la matrice ?
- Pour afficher le chiffre 15 ?
- **Quel est l'indice maximum pour les lignes ? les colonnes ?**

Après avoir compris ces principes de manipulations, on peut adapter cette structure à notre plateau de jeu. Le jeu n'est autre qu'une matrice, où chaque case prend une valeur différente suivant ce qu'elle représente sur l'interface graphique. Petite explication :




Si on déplace le personnage vers le haut, cela donnerait :




Ainsi, lorsque l'on appuie sur une touche pour faire monter le personnage d'une case, on met à jour **simultanément** :

- L'interface graphique
- La matrice

Mais surtout, on se sert de la matrice pour représenter **l'état du jeu**, c'est-à-dire :

- Où se trouve mon personnage
- Quelles sont les cases libres
- Quelles sont les cases que je ne peux pas franchir
- Etc…

# A faire :

- Définir dans le fichier ex-matrice.py la matrice représentant l'état du jeu avant de déplacer le personnage et l'afficher. Dans la matrice, les éléments peuvent être n'importe quel type d'élément (chiffre, texte,…) **CEPENDANT** du texte doit être défini entre guillemets

Exemple : ["a", "b" , "c" ]

# Accéder à un élément d'une liste

Une liste en Python est définie de la sorte : [élément1, élément2, élément3 …]

Ainsi, chaque élément est caractérisé par sa **valeur**, mais aussi par sa **position dans la liste**.

Par exemple, dans la liste [2,4,5], le premier élément est le chiffre 2, et se trouve à la **position 0 (ATTENTION, la première position d'une liste est 0, puis la deuxième position est 1, etc…).**

Ainsi, pour accéder dans la liste L1 à un élément à la position X, il faut utiliser la commande L1[X].

Prenons comme exemple la liste L2 = [1,10,25,10,2]

- Quelle est la commande pour afficher l'élément « 1 » ?
- Quelle est la commande pour afficher l'élément « 25 » ?

# Modifier un élément de la liste

On sait maintenant comment accéder à un élément de la liste à l'aide de sa **position dans la liste**.

Pour modifier un élément de la liste, il suffit de réaffecter la valeur de cet élément comme dans l'affectation d'une variable.

Exemple : L1 = [0,1,2]
pour remplacer le 0 par un 3, il suffit d'utiliser la commande L1[0]=3

Prenons la liste L2 = [0,0,0,0]

- Quelles commandes effectuer pour que L2 = [1,2,0,4] ?

# Afficher une liste

Pour afficher une liste, il suffit d'utiliser la commande **print(nom_de_la_liste)**

Exemple :

L1 = [1,2,3]

Print(L1) donnera l'affichage [1,2,3]

# Longueur d'une liste

Pour connaître la longueur d'une liste, il suffit d'utiliser la fonction **len(nom_de_la_liste).**

Par exemple, on a L1 = [1,2,3,4], si on cherche à afficher la longueur de la liste L1, on exécute la commande print(**len(L1)**)
car :

- len(L1) calcule la longueur de la liste L1
- print(len(L1)) calcule la longueur de la liste L1, puis l'affiche

Par exemple, la liste L2 = [1,2,3,4,5]

- Quel sera le résultat de len(L2) ?

# La boucle for

Une boucle for permet de répéter un certain nombre de fois une opération.

La syntaxe python est la suivante :

```
For indice in range (0,3):
        print(indice)
```

Quelques explications…

- On appelle une variable « indice »
- On lui donne la valeur 0
- On print(indice) qui donnera comme résultat 0
- On donne à « indice » la valeur 1
- On print(indice) qui donnera comme résultat 1
- On donne à « indice » la valeur 2
- On print(indice) qui donnera comme résultat 2
- On sort de la boucle for

Ce qui donne comme résultat final dans la console :

```
0
1
2
```

- Que se passe il si on remplace le range(0,2) par range(0,5) ?

- Que se passe il si on remplace le range(0,2) par range(2,5) ?
- Ecrire une boucle for qui permet cet affichage :
    - 0
      2
      4
      6
- Ecrire une boucle for qui permet cet affichage :
    - 0
      0
      0
      0

# Parcourir une liste

Si on connait la longueur d'une liste, on peut utiliser une boucle for pour parcourir une liste, élément par élément.

Exemple : une liste L1 = [10,20,30,40]
len(L1) = 4
L1[0]=10
L1[1]=20
L1[2]=30
L1[3]=40

Ainsi, si on utilise une boucle for, on peux remplacer le 0,1,2,3 des indices de position dans la liste par une variable automatisée.

Explications :

```
L1 = [10,20,30,40]
for indice in range (0,len(L1)):
        print(L1[indice])
```

Indice va donc varier de 0 à len(L1), c'est-à-dire de 0 à 4
Ensuite, L1[indice] va successivement être L1[0], L1[1], L1[2], L1[3]

Cette boucle for va donc permettre l'affichage successif de L1[0], L1[1], L1[2], L1[3] donc provoquer l'affichage suivant :

10
20
30
40

# A faire

Recopier L2 = [1,2,3,4,5,6]

- Comment afficher un à un les éléments de la liste L2 ?
- Comment seulement afficher le premier, deuxième et troisième élément de L2 avec une boucle for ?
- Comment seulement afficher l'avant dernier et le dernier élément de L2 avec une boucle for ?

Recopier L3 = [0,0,0,0,0]

- Comment faire avec une boucle for pour que L3 devienne [1,2,3,4,5] ?

# Matrice et listes ?

En outre, une matrice n'est autre qu'une liste de listes.

Prenons comme exemple M1 = [ [1,2,3] , [4,5,6], [7,8,9] ]

- Si nous exécutons la commande print(M1[0]), qu'est ce qui sera affiché ?
- Pareil pour print(M1[1]) ?

Ainsi, pour afficher le premier élément de la première liste de M1, il faut effectuer la commande suivante : print(M1[0][0])

- Quelle commande pour afficher le 3eme élément de la 1ere liste ?
- Quelle commande pour afficher le chiffre 5 contenu dans la matrice M1 ?

Mais comment afficher TOUS les éléments d'une matrice ?

Il suffit d'utiliser une **double boucle for.**

On peut très bien faire une première boucle for, puis une deuxième à l'intérieur de celle-ci. A vous de trouver comment afficher tous les éléments de M1 à l'aide d'une double boucle for.

If et Else

On sait maintenant comment créer une liste, la modifier, l'afficher, etc.

Désormais, on cherche à savoir quand exécuter une commande ou non, en fonction d'un résultat. La structure de contrôle « if – else » permet cela.

Exemple :

```
x=4
if x>3 :
        print(« oui ! »)
else :
        print(« non ! »)
```

Ce programme va afficher « oui ».

```
x=2
if x>3 :
        print(« oui ! »)
else :
        print(« non ! »)
```

Ce programme va afficher « non ».

*Pour tester l'égalité entre deux éléments, on utilise l'opérateur ==*
*(3==3 est vrai, 3==2 est faux)*
*Pour tester l'inégalité entre deux éléments, on utilise l'opérateur !=*
*(2 !=3 est vrai, 3 !=3 est faux)*

Testons ce programme :

```
x=3
if (x+2)==5 :
        print(x)
else:
        print(x+1)
```

Quel sera l'affichage de ce programme ?

```
x=4
if (x != 4) :
        print(x)
else
        print("hello")
```

Quel sera l'affichage de ce programme ?

# A faire

Maintenant, récupérez la matrice représentant l'état du jeu que vous deviez faire à la fin de la séance précédente. Si vous ne l'avez pas, refaites en une petite de 5 cases sur 5 cases.

Représentez le personnage par un chiffre 1, les cases libres par un 0, et les murs par un 9.

Créez un programme permettant de déplacer le chiffre 1 dans la matrice avec les touches du clavier, à l'aide de tout ce que vous avez appris jusqu'à présent (la boucle for n'est pas utile pour cette séance).