# LES BASES PYTHON

## Les variables

Une variable est une etiquette qui va nous permettre facilement de stocker et d’accèder à des données dans la mémoire de l’ordinateur. Le nom d’une variable peut s’écrire comme on veut du moment qu’elle commence par une lettre ou un « \_ ».

Mais des régles existent « CONVENTION » et il est dit par exemple que le premier mot d’une variable doit etre en minuscule et que chaque mot suivant à la premier lettre en majuscule.

**Exemple de variables communes :**

* *maPetiteVariable = 0*
* *nom = ‘’boris’’*
* *superVariable39 = 12*

**Exemple de variables dites CONSTANTES : (qui ne changeront pas)**

* *DIMENSION\_DE\_MON\_ECRAN = 12*
* *CODE\_SECRET = ‘’oiseau’’*

**Mais a quoi elles servent ?**

*Imaginons que l’on écrive une histoire de 200 pages. Dedans le personnage principal apparait 613 fois. Et à la fin du livre, on s’apercoit que le nom du personnage est nul... C’est la galère...*

*Maintenant, à la place du nom du personnage, on utilise une variable NOM\_DU\_PERSONNAGE. Du coup, à tout moment on peut changer le nom du personnage en un claquement de doigt.*

**Les types de variables :**

|  |  |
| --- | --- |
| **TYPE : Chaine (STRING - STR)** | |
| # --- Variable de type chaine (STRING)  # --- on peut mettre des lettres, chiffres, symboles ...  # --- /!\ toujours des guillemets avant et après /!\  nom = "CHAIX"  prenom = "Guillaume"  print(nom)  print(prenom)  print(nom, prenom) | |
| Résultat : | CHAIX  Guillaume  CHAIX Guillaume |

Exercices pratiques :

1. Quel est la fonction qui permet d’afficher le contenu d’une variable ?

|  |  |
| --- | --- |
| **TYPE : Entier (INTEGER - INT)** | |
| # --- Variable de type entier (INTEGER)  # --- on ne peut mettre que des nombres sans virgule, positif (12121), ou négatif (-121212)  age = 37  print(age) | |
| Résultat : | 37 |

|  |  |
| --- | --- |
| **TYPE : Décimal (FLOAT)** | |
| # --- Variable de type décimale (FLOAT)  # --- on peut mettre n'importequel nombre avec une virgule, positif (3.45) ou négatif (-2.45)  # --- /!\ les virgules sont remplacées par des points /!\  poids = 74.3  print(poids) | |
| Résultat : | 74.3 |

|  |  |
| --- | --- |
| **TYPE : Booléen (BOOLEAN - BOOL)** | |
| # --- Variable de type booléen (BOOL)  # --- Elle ne peut avoir que deux valeurs, VRAI ou FAUX, True ou False  jeSuisLePlusFort = True  jeSuisMalade = False  ceProgrammeEstPourri = True  print(jeSuisLePlusFort)  print(jeSuisMalade)  print(ceProgrammeEstPourri) | |
| Résultat : | True  False  True |

|  |  |
| --- | --- |
| **TYPE : Tuple (TUPLE)** | |
| # --- Variable de type tuple ()  # --- c'est une variable avec de multiples valeurs  positionSurLaCarte = (12, 13)  positionX, positionY = positionSurLaCarte  print(positionSurLaCarte)  print(positionSurLaCarte[0], positionSurLaCarte[1])  print(positionX, positionY) | |
| Résultat : | (12, 13)  12 13  12 13 |

|  |  |
| --- | --- |
| **TYPE : Liste (LIST)** | |
| # --- Variable de type liste (LIST) []  # --- on peut mettre n'importequel TYPE DE VARIABLE dedans !!!!  # --- on accede à chaque élèment en spécifiant son index dans des crochets [id]  # --- /!\ le premier est toujours 0 et le dernier (MAX -1) /!\  enfants = ["Hugo", "Lola", "Louis"]  bordel = [12.3, "Salut", 12121212]  print(enfants)  print(enfants[0], enfants[2])  print(bordel[0]) | |
| Résultat : | ['Hugo', 'Lola', 'Louis']  Hugo Louis  12.3 |

Exercices pratiques :

1. Quel est le type du premier élèment de la variable « bordel » ?
2. Quel est l’indice de « Lola » ?

|  |  |
| --- | --- |
| **TYPE : Dictionnaire (DICTIONARY)** | |
| # --- Variable de type dictionnaire (DICTIONARY) {}  # --- On associe un TYPE DE VARIABLE à une liste  nourrituresPreferes = {"BOISSON" : ["COCA-COLA", "CHAMPAGNE"], \                         "PLAT CHAUD" : ["COUSCOUS", "RACLETTE", "MCDO"]}  mesLivres = {1 : ["LA MORT DES ZOMBIS"], \               2 : ["DINGUE DE TOI"], \               10 : ["PLUS D'IDEE"]}  print(nourrituresPreferes)  print(nourrituresPreferes["BOISSON"])  print(nourrituresPreferes["BOISSON"][1])  print(mesLivres[10])  print(mesLivres[10][0]) | |
| Résultat : | {'BOISSON': ['COCA-COLA', 'CHAMPAGNE'], 'PLAT CHAUD': ['COUSCOUS', 'RACLETTE', 'MCDO']}  ['COCA-COLA', 'CHAMPAGNE']  CHAMPAGNE  ["PLUS D'IDEE"]  PLUS D'IDEE |

Exercices pratiques :

1. En utilisant la variable « nourrituresPreferes », affiches « CHAMPAGNE » ?
2. Quel est l’indice de « MCDO » ?

## Les opérateurs arithmétiques

L’addition + fonctionne avec les chaines (STRING), entiers (INTEGER) et les décimaux (FLOAT).

La soustraction -, multiplication \*, division \ ne fonctionnent qu’entre types numériques (entiers, décimaux).

|  |  |
| --- | --- |
| **+ ADDITION** | |
| nom = "CHAIX"  prenom = "Guillaume"  age = 37  poids = 74.3  nomComplet = nom + " " + prenom  dans10Ans = age + 10  apresMcDo = poids + 1.2  print("Nom Complet : ", nomComplet)  print("Dans 10 ans, j'aurai : ", dans10Ans)  print("Mon poids après le Mcdo : ", apresMcDo) | |
| Résultat : | Nom Complet : CHAIX Guillaume  Dans 10 ans, j'aurai : 47  Mon poids après le Mcdo : 75.5 |

Par contre, on ne peut pas directement additionner une chaine à un nombre (entier, décimaux). Pour cela, il faut utiliser une fonction de conversion str().

|  |  |
| --- | --- |
| **CHAINE + (INTEGER ou FLOAT)** | |
| nom = "CHAIX"  prenom = "Guillaume"  age = 37  nomComplet = nom + " " + prenom  print(nomComplet + " a " + age + " ans.") | |
| Résultat : | **Une exception s'est produite : TypeError**  can only concatenate str (not "int") to str  File "D:\#ENTREPRISES\PYTHON-SERPY\variables.py", line 10, in <module> print(nomComplet + " a " + age + " ans.") |
| ...  print(nomComplet + " a " + str(age) + " ans.") | |
| Résultat : | CHAIX Guillaume a 37 ans. |

|  |  |
| --- | --- |
| **CALCUL** | |
| calcule2LaMort = ((10 \* 5) - 4) / 2  print("Resultat : ", calcule2LaMort) | |
| Résultat : | Resultat : 23.0 |

## Les conditions (SI)

Les conditions permettent à l’ordinateur de faire tel ou tel choix.

**Attention à l’indentation**

|  |  |
| --- | --- |
| **SI** (condition) **ALORS**  (faire action)  **OU** **SI** (condition) **ALORS**  (faire action)  **OU** **SI** (condition) **ALORS**  (faire action)  **SINON**  (faire action) | **if** (condition) **:**  (faire action)  **elif** (condition) **:**  (faire action)  **elif** (condition) **:**  (faire action)  **else :**  (faire action) |

|  |  |
| --- | --- |
| **PLUS PETIT < , PLUS GRAND > , EGALE ==** | |
| age = 8  if age < 12:      print("Tu es trop petit.")  elif age == 12:      print("Tu a pile l'age qu'il faut.")  else:      print("Tu es trop grand.") | |
| Résultat : | Tu es trop petit. |

Il est possible de tester plusieurs conditions avec des opérateurs logiques.

|  |  |
| --- | --- |
| **SI** (condition) **ET** (condition) **ALORS**  (faire action)  **SINON**  (faire action) | **if** (condition) **and** (condition) **:**  (faire action)  **else :**  (faire action) |

|  |  |
| --- | --- |
| **ET (AND) – avec des entiers** | |
| age = 8  print("Tu regarderas à la TV :")  if age < 3:      print("Snoopy")  elif age > 2 and age < 8:      print("PowerRangers")  elif age > 7 and age < 16:      print("Chair de poules")  else:      print("Walking Dead") | |
| Résultat : | Chair de poules |

|  |  |
| --- | --- |
| **OR (OU) – avec des chaines** | |
| maReponse = input("Tapez le code ????")  codeSecret = "MORIS"  if maReponse == codeSecret or maReponse == "crotte":      print("Accès Autorisé")  else:      print("Accès Refusé") | |
| Résultat :  Résultat :  Résultat : | Tapez le code ????magique  Accès Refusé  Tapez le code ????crotte  Accès Autorisé  Tapez le code ????MORIS  Accès Autorisé |

|  |  |
| --- | --- |
| **NOT (NON) – avec des chaines** | |
| maReponse = input("Tapez le code ????")  codeSecret = "MORIS"  if not (maReponse == codeSecret):      print("Accès Autorisé")  else:      print("Accès Refusé") | |
| Résultat :  Résultat :  Résultat : | Tapez le code ????magique  Accès Autorisé  Tapez le code ????crotte  Accès Refusé  Tapez le code ????MORIS  Accès Refusé |

## Les boucles

Une boucle est une fonction qui permet la repetition. Il en existe plusieurs. Je vais en présenter 2 qui peuvent s’utiliser dans tous les cas, mais qui sont plus efficaces dans certaines situations.

La boucle FOR

Elle permet de parcourir un ensemble du début, jusqu’a la fin.

|  |  |
| --- | --- |
| **BOUCLE FOR** | |
| # --- Boucle FOR  # --- for (variable) in (liste\_de\_choix)  # ---   (variable) => contient la valeur parcourue  # ---   (liste\_de\_choix) => liste de valeurs  for i in [0, 1, 2, 3]:      print (i) | |
| Résultat : | 0  1  2  3 |

La boucle WHILE

Elle permet de se repetter tant que la condition qui lui est associé est VRAI.

|  |  |
| --- | --- |
| **BOUCLE WHILE - ENTIER** | |
| # --- Boucle WHILE  # --- while (condition):  i = 0  while i < 4:      print(i)      i = i + 1 | |
| Résultat : | 0  1  2  3 |

|  |  |
| --- | --- |
| **BOUCLE WHILE - BOOLEEN** | |
| # --- Boucle WHILE  # --- while (condition):  i = 0  malAlaTete = True  while malAlaTete:      print(i)      if i < 3:          i = i + 1      else:          malAlaTete = False | |
| Résultat : | 0  1  2  3 |

## Les fonctions

|  |  |
| --- | --- |
| **FONCTIONS AVEC/SANS PARAMETRES** | |
| def direBonjour():      print("Bonjour")  def dire\_resultat\_addition(a, b):      c = a + b      print("Résultat : ", c)  direBonjour()  dire\_resultat\_addition(5, 6) | |
| Résultat : | Bonjour  Résultat : 11 |

## Les classes

|  |  |
| --- | --- |
| **FONCTIONS AVEC/SANS PARAMETRES** | |
| class homme():      def \_\_init\_\_(self, \_nom, \_age):          self.nom = \_nom          self.age = \_age      def parler(self):          print("Salut, je m'appelle " + self.nom)      def age(self):          return self.age  soldat1 = homme("Guillaume", 37)  soldat2 = homme("Robert", 99)  soldat1.parler()  soldat2.parler()  print("L'age du soldat 1 est ", soldat1.age, " ans.")  print("L'age du soldat 2 est ", soldat2.age, " ans.") | |
| Résultat : | Salut, je m'appelle Guillaume  Salut, je m'appelle Robert  L'age du soldat 1 est 37 ans.  L'age du soldat 2 est 99 ans. |

# PYGAME



Pygame à le rôle d'une interface graphique, pour un langage comme Python !

C’est à dire qu’il permet de créer une fenêtre et d’y afficher des images, de jouer de la musique et du sons, et de réagir quand on utilise le clavier, la souris ou une manette trés facilement.

Il faut savoir que Pygame ne fait pas partie des bibliothèques standards de Python, qu'elle est donc développée indépendamment et qu'il est nécessaire de la télécharger et de l'installer en plus de Python pour pouvoir l'utiliser. Celle-ci est gratuite et libre.

# LES FONDATIONS

## Création de la fenêtre de jeu

Tu vas créer la fenêtre qui contiendra ton jeu.  
Ce qui est important dans cette partie est l’utilisation de Pygame,   
et la boucle principale qui recommencera à l’infini pour animer ton jeu.

1. Crées un nouveau fichier « **jeu.py** ».
2. *Tu vas ajouter les lignes qui permettent de dire à l’ordinateur que tu vas utiliser les fonctions de PyGAME. Ensuite tu vas créer une fenêtre et lui donner le nom de ton jeu.*  
   Recopies le code en entier ci-dessous.
3. Executes ton programme.

|  |  |
| --- | --- |
| **Jeu.py** | |
| import pygame  « import » des bibliothèques PYGAME va te permettre d’utiliser des fonctions de création de jeux vidéos. « pygame.ini » te permet d’activer le moteur PYGAME.  C’est lui qui fera tourner ton jeu.  from pygame.locals import \*  # --- Initialisation  pygame.init()  On crée la fenetre de ton jeu avec une dimension de 800 pixels de largeur et 600 pixels de hauteur.  On met un titre à la fenêtre de ton jeu « SerPy » pour Serpent Python.  fenetre = pygame.display.set\_mode((800, 600))  pygame.display.set\_caption("SerPy") | |
| Résultat : | La fenêtre apparait, mais disparait quelques secondes après ! |

1. *Tu vas rajouter la boucle principale de ton jeu, qui se repettera à l’infini et qui évitera au programme de s’arrêter tout seul.*   
   Complètes ton programme avec le code dans la zone grise.

|  |  |
| --- | --- |
| **Jeu.py** | |
| import pygame  from pygame.locals import \*  # --- Initialisation  pygame.init()  fenetre = pygame.display.set\_mode((800, 600))  pygame.display.set\_caption("SerPy")  # --- Boucle du jeu  Tant qu’on donne pas l’ordre au programme d’arrêter ton jeu, il lancera sans fin, le code à l’intérieur.  boucle\_active = True  while boucle\_active:      pygame.draw.rect(fenetre, (0,0,0), (0, 0, 800, 600))  Dessine un rectangle noir (0,0,0), à la position X : 0, Y : 0, de largeur 800 et hauteur 600.      pygame.display.flip()  Affiche tout ce qui a été dessiné à l’écran. | |
| Résultat : | La fenêtre apparait, reste, mais le programme reste bloqué ! |

1. *Tu vas ajouter les lignes qui permettent à l’ordinateur de sortir de la boucle si on clique sur la croix de la fenêtre ou si tu appuies sur la touche Echappe. Sortir de la boucle mettra fin à ton jeu.*  
   Complètes ton programme avec le code dans la zone grise.

|  |  |
| --- | --- |
| **Jeu.py** | |
| import pygame  from pygame.locals import \*  # --- Initialisation  pygame.init()  fenetre = pygame.display.set\_mode((800, 600))  pygame.display.set\_caption("SerPy")  # --- Boucle du jeu  boucle\_active = True  while boucle\_active:      pygame.draw.rect(fenetre, (0,0,0), (0, 0, 800, 600))      for event in pygame.event.get():  Le programme regarde si la X de la fenêtre a été cliqué ou si la touche ESC a été pressée pour arrêter la boucle. principale !!!          if event.type == QUIT \          or (event.type == KEYDOWN and event.key == K\_ESCAPE):              boucle\_active = False      pygame.display.flip()  # --- Fin du programme  Ferme la fenêtre.  Ferme le programme.  On quitte proprement.  pygame.display.quit()  pygame.quit() | |
| Résultat : | La fenêtre apparait. Le programme attend que tu quittes !  Et quitte proprement. |

## Mise en place des variables communes

Au cours de la création de ton jeu,   
tu seras amené à modifier plusieurs fois les valeurs de ton jeu.   
Imagines avoir écris 100 fois le nom de ton jeu dans ton code.   
Si tu le changes, il faudra le faire de partout.   
Si tu utilises une variable, tu n’auras qu’a changer sa valeur une fois. Pratique !!!

1. Crées un nouveau fichier « **variables.py** »
2. *Tu vas recopier ses différentes variables, elles te servirons tout au long de la conception de ton jeu. Tu en rajouteras par la suite un certain nombre. Celle écrite en lettre capitale sont des CONSTANTES, c’est à dire que leur valeur ne changera pas.*   
   Recopies l’ensemble du code.

*A chaque fois que tu voudras utiliser les variables de ce fichier dans un autre fichier du jeu, tu ajouteras en debut de code la ligne «****import variables as VAR****».*

* Import variables, indiquera au programme que tu veux accéder aux variables de ce fichier.
* as VAR, te permettra de ne taper que VAR.COULEUR\_MUR plutot que VARIABLES.COULEUR\_MUR pour accèder au contenu de la variable.
* Il est important de commenter les variables afin de savoir a quoi elles servent.

|  |
| --- |
| **variables.py** |
| # --- Couleurs  COULEUR\_MUR = (0,128,0)  CONSTANTES (variable qui ne changera pas) de couleurs.  Une couleur est un tuple de (3 valeurs) :  Valeur 1 : rouge  Valeur 2 : vert  Valeur 3 : bleu  Chaque valeur est comprise entre 0 et 255.  0 étant noir et 255 étant la couleur vive.  COULEUR\_FOND = (20,0,20)  COULEUR\_FOND\_ECRAN = (16,16,16)  COULEUR\_BLANC = (255,255,255)  COULEUR\_NOIR = (0,0,0)  COULEUR\_ROUGE = (255,0,0)  COULEUR\_BLEU = (0,0,255)  COULEUR\_ORANGE = (255,128,0)  # --- Constantes du jeu  CONSTANTES de ton jeu.  Le titre du jeu et les dimensions de la fenêtre.  X => Ce sont les colonnes (Largeur)  Y => Ce sont les lignes (Hauteur)  TITRE = "SerPy"  ECRAN\_X = 800  ECRAN\_Y = 600  # --- Variables  boucle\_active = True  La variable qui permet de quitter ton jeu. |

1. A présent, tu vas remplacer toutes les valeurs contenues dans le fichier « **jeu.py** » par les variables adéquates.

* Il ne faut pas oublier d’importer le fichier « **variables** » et d’y ajouter le surnom « **as VAR** ».
* Il ne faut pas oublier d’ajouter « **VAR.** » devant chaque nom de variable.

|  |  |
| --- | --- |
| **Jeu.py** | |
| import pygame  from pygame.locals import \*   * L’avantage des variables est que si on change une valeur, on la change une seule fois. * A partir de maintenant, on pourra changer certains réglages sans devoir relire tout le code. On ira simplement dans le fichier « variables.py »   # On importe nos fichiers  import variables as VAR  # --- Initialisation  pygame.init()  fenetre = pygame.display.set\_mode((VAR.ECRAN\_X, VAR.ECRAN\_Y))  pygame.display.set\_caption(VAR.TITRE)  # --- Boucle du jeu  VAR.boucle\_active = True  while VAR.boucle\_active:      pygame.draw.rect(fenetre, VAR.COULEUR\_FOND\_ECRAN, (0, 0, VAR.ECRAN\_X, VAR.ECRAN\_Y))      for event in pygame.event.get():          if event.type == QUIT or (event.type == KEYDOWN and event.key == K\_ESCAPE):              VAR.boucle\_active = False      pygame.display.flip()  # --- Fin du programme  pygame.display.quit()  pygame.quit() | |
| Résultat : | Rien n’a changé ! Normal, tu as juste remplacé des valeurs par des variables ! |

# LES OBJETS DU JEU

## Création de l’objet Serpent

### Voir le serpent

Il est temps de créer notre serpent.   
Nous allons donc créer un objet serpent qui aura des variables et des fonctions bien à lui.  
Une fois créé notre serpent deviendra ta marionette ...

* 1. *Ta fenêtre représente une grille de 800 pixels de largeur sur 600 de hauteur. Chaque partie de ton serpent pourrait avoir la taille d’un pixel, mais cela serait trop petit à l’écran. C’est pour cela, que tu vas créer une nouvelle variable « tailleCellule ». Elle représentera une cellule de ta grille et du coup un morceau de ton serpent.*Ajoutes la variable « **tailleCellule** » dans le fichier « **variables.py** »

|  |
| --- |
| **variables.py** |
| tailleCellule = 16 |

* 1. Crées un nouveau fichier « **serpents.py** ».
  2. *Maintenant, tu vas créer un nouveau type de variable. Ton type a toi, celui du serpent. Ce type est « objet\_serpent ». Comme tout objet, il contiendra des variables et des fonctions son fonctionnement.*Recopies le code ci-dessous.

|  |
| --- |
| **serpents.py** |
| import pygame  from pygame.locals import \*  On importe les fichiers que tu vas utilisé.  import variables as VAR  class objet\_serpent:  On crée l’objet « objet\_serpent ».  Quand on l’utilisera, la fonction « \_\_init\_\_ » sera appelé et elle fabriquera :   * un corps vide * une couleur rouge * et ajoutera un morceau du serpent sur l’écran à la position (x :15, y :15)         # --- Crée notre serpent      def \_\_init\_\_(self):          self.corps = []          self.couleur = VAR.COULEUR\_ROUGE          # --- Création du corps du serpent          self.corps.append((15,15))        # --- Affiche notre serpent      def afficher(self, \_fenetre):          for bout\_du\_serpent in self.corps:  Récupére la position de chaque morceau du serpent, puis l’affiche sur l’écran.              x, y = bout\_du\_serpent              pygame.draw.rect(\_fenetre, self.couleur, \                  ((x \* VAR.tailleCellule), (y \* VAR.tailleCellule), \                  VAR.tailleCellule, VAR.tailleCellule)) |

* 1. *On va utiliser les fonctions du serpent.*Importes le fichier « **serpents** », tu vas en avoir besoin.
  2. On crée une variable « **serpent** » à partir de « **serpents**.**objet\_serpent** » !
  3. Et on appelle la fonction « **afficher()** » de notre variable « **serpent** ».

|  |  |
| --- | --- |
| **jeu.py** | |
| import pygame  from pygame.locals import \*  # On importe nos fichiers  import variables as VAR  On importe le fichier serpents afin de pouvoir utiliser les fonctions dedans.  import serpents  # --- Initialisation  pygame.init()  fenetre = pygame.display.set\_mode((VAR.ECRAN\_X, VAR.ECRAN\_Y))  pygame.display.set\_caption(VAR.TITRE)  # --- Création de l'objet serpent  On crée une variable de type « objet\_serpent ».  C’est à ce moment la que la fonction \_\_init\_\_() de serpent va être lancé.  serpent = serpents.objet\_serpent()  # --- Boucle du jeu  VAR.boucle\_active = True  while VAR.boucle\_active:      pygame.draw.rect(fenetre, VAR.COULEUR\_FOND\_ECRAN, (0, 0, VAR.ECRAN\_X, VAR.ECRAN\_Y))      # --- On controle les actions du joueurs.      for event in pygame.event.get():          if event.type == QUIT or event.type == KEYDOWN and event.key == K\_ESCAPE:              VAR.boucle\_active = False # Indique de sortir de la boucle.    On appelle la fonction afficher() de notre variable serpent.  On oublie pas d’indiquer en paramètre de la fonction la surface où sera dessiner ton serpent.      serpent.afficher(fenetre)      # --- Actualise la fenêtre      pygame.display.flip()  # --- Fin du programme  pygame.display.quit()  pygame.quit() | |
| Résultat : | Tu dois voir un point rouge aux coordonnées du serpent (15, 15). |

* 1. Lances ton jeu.

### Animer le serpent

*Ton serpent va bouger, mais a quelle vitesse ? On va lui dire de faire une pause entre chaque mouvement. Une petite pause 100ms, qui équivaut à 10 mouvements par seconde, devrait être suffisant. Plus la valeur sera petite, plus ton serpent ira vite.*.

* 1. Dans le fichier « **variables.py** ».
  2. Ajoutes la variable « **vitesse** », la valeur est en seconde.

|  |
| --- |
| **variables.py** |
| vitesse = 0.10  Délais de mouvement commun à tous les serpents. |

* 1. Dans le fichier « **serpents.py** ».
  2. Importes la bibliothèque « **time** ».
  3. Ajoutes les nouvelles variables « **vitesse** » et « **vitesse\_timer** » qui vont te permettre de limiter la vitesse de ton serpent.
  4. Ajoute la variable « **direction** » qui indiquera au programme où va ton serpent.

|  |
| --- |
| **serpents.py : \_\_init\_\_()** |
| import pygame  from pygame.locals import \*  Import des fonctions qui gère le temps.  import time  import variables as VAR  class objet\_serpent:      # --- Crée notre serpent  La variable « vitesse » contient le délais entre chaque mouvement.  La variable « vitesse\_timer() » contient la valeur du chronomètre de l’ordinateur du précédent mouvement du serpent.      def \_\_init\_\_(self):          self.corps = []          self.vitesse = VAR.vitesse          self.vitesse\_timer = time.time()          self.couleur = VAR.COULEUR\_ROUGE  La variable « direction » va permettre de savoir où va le serpent.          self.direction = "" |

* 1. Dans le fichier « **serpents.py** ».
  2. Tu vas créer une nouvelle fonction « se\_deplacer ».
  3. Recopies l’ensemble du code.

*La fonction « se\_deplacer » n’est pas très compliqué. Elle consiste à :*

* *Controle le delais entre chaque mouvement*
* *Récupère la position de la tête*
* *Deplace la tête*
* *Enregistre la position de la tête*

|  |
| --- |
| **serpents.py : se\_deplace()** |
| class objet\_serpent:  # --- Affiche notre serpent      def afficher(self, \_fenetre):    (...)      # --- Deplace notre serpent      def se\_deplace(self):          # --- Temporise le deplacement          if time.time() < self.vitesse\_timer + self.vitesse:  Ca vérifie que le délais entre chaque mouvement de ton serpent est respecté.              return          self.vitesse\_timer = time.time()          # --- Position de la tete après deplacement          xTete, yTete = self.corps[0]  Ca récupère la position X et Y du premier morceau du serpent, la tête !!!          if self.direction == "BAS":              yTete = yTete +1          elif self.direction == "HAUT":              yTete = yTete -1  Le programme modifie les variables de position selon la direction de ton serpent.  Si je vais en haut, je monte d’une ligne ...  Si pas de durection, je sors de la fonction sans rien faire.          elif self.direction == "DROITE":              xTete = xTete +1          elif self.direction == "GAUCHE":              xTete = xTete -1          else:              return None    Réinjecte les positions modifés au morceau (0) du serpent. La tête !          self.corps[0] = (xTete, yTete) |

1. Dans le fichier « **jeu.py** ».
2. Tu vas rajouter la gestion du clavier.
3. Tu vas ensuite appeler la fonction « se\_deplacer » du serpent pour qu’il se déplace.

*A chaque fois que tu appuieras sur une touche la direction de ton serpent prendra une valeur de son orientation. Cette orientation sera traiter ensuite grace à sa fonction « se\_deplacer().*

|  |  |
| --- | --- |
| **jeu.py** | |
| VAR.boucle\_active = True  while VAR.boucle\_active:      pygame.draw.rect(fenetre, VAR.COULEUR\_FOND\_ECRAN, (0, 0, VAR.ECRAN\_X, VAR.ECRAN\_Y))      # --- On controle les actions du joueurs.      for event in pygame.event.get():          if event.type == QUIT or event.type == KEYDOWN and event.key == K\_ESCAPE:              VAR.boucle\_active = False # Indique de sortir de la boucle.            if event.type == KEYDOWN:              if event.key == K\_RIGHT : serpent.direction = "DROITE"  Change la direction de ton serpent, selon la touche appuyée.              if event.key == K\_LEFT  : serpent.direction = "GAUCHE"              if event.key == K\_UP    : serpent.direction = "HAUT"              if event.key == K\_DOWN  : serpent.direction = "BAS"    Déplace ton serpent, grace à la fonction « se\_deplace() ».      serpent.se\_deplace()      serpent.afficher(fenetre)      # --- Actualise la fenêtre      pygame.display.flip() | |
| Résultat : | Utilises les fléches, le serpent bouge ! |

### Ajouter la queue du serpent

1. Dans le fichier « **serpents.py** ».
2. Modifies le code comme ci-dessous.

|  |
| --- |
| **serpents.py : \_\_init\_\_()** |
| # --- Crée notre serpent      def \_\_init\_\_(self):          self.corps = []          self.vitesse = VAR.vitesse          self.vitesse\_timer = time.time()          self.couleur = VAR.COULEUR\_ROUGE          self.direction = ""  Repete 8 fois, la création d’un morceau à la position (15,15). Le programme va créer la tête et la queue sur le même emplacement.          # --- Création du corps du serpent          for i in range(0, 8):              self.corps.append((15,15)) |

1. Ajoutes le code qui va permettre de déplacer l’ensemble de ton serpent.

|  |
| --- |
| **serpents.py : se\_deplace()** |
| # --- Deplace notre serpent      def se\_deplace(self):          # --- Temporise le deplacement          if time.time() < self.vitesse\_timer + self.vitesse:              return          self.vitesse\_timer = time.time()          # --- Deplace le derrière vers l'avant, jusqu'a la tête          for i in range(len(self.corps)-1, 0, -1):  En partant de la fin, chaque morceau du serpent prend la position du morceau juste devant, la boucle ne traite pas la tête. Du coup le corps du serpent avance ... Telle la chenille ...  *La tête (morceau 0) se retrouve sous un bout de la queue, car le morceau 1 prend la place du morceau 0.*  Ce n’est pas grave, puisque plutard, on déplace la tête !              self.corps[i] = self.corps[i-1]          # --- Position de la tete après deplacement          xTete, yTete = self.corps[0]          if self.direction == "BAS":              yTete = yTete +1 |

1. Bloques le serpent pour qu’il ne puisse pas se retourner.

|  |  |
| --- | --- |
| **jeu.py** | |
| if event.type == KEYDOWN:              if event.key == K\_RIGHT and not serpent.direction == "GAUCHE": serpent.direction = "DROITE"              if event.key == K\_LEFT  and not serpent.direction == "DROITE": serpent.direction = "GAUCHE"              if event.key == K\_UP    and not serpent.direction == "BAS":    serpent.direction = "HAUT"              if event.key == K\_DOWN  and not serpent.direction == "HAUT":   serpent.direction = "BAS"        serpent.se\_deplace()      serpent.afficher(fenetre)  Empêche le serpent de se retourner brutalement, il se mordrait la queue !  Du coup le corps du serpent avance ... Telle la chenille ...  *La tête (morceau 0) se retrouve sous un bout de la queue, car le morceau 1 prend la place du morceau 0.*  Ce n’est pas grave, puisque plutard, on déplace la tête ! | |
| Résultat : | Le serpent est plus grand et il ne peut pas se retourner ! |

# Création de l’objet Terrain

## Création de l’air de jeu.

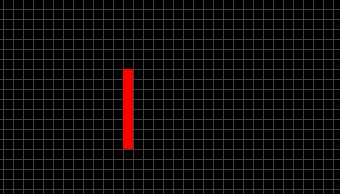
Tu vas créer l’air de jeu, c’est à dire une grille sous le serpent.

* 1. Crées un nouveau fichier « **terrains.py** ».
  2. Recopies le code ci-dessous.

|  |
| --- |
| **terrains.py** |
| import pygame  from pygame.locals import \*  import variables as VAR  dimensionX = 80  dimensionY = 60  # --- Dimension du carré qui représente le terrain, ainsi que les morceaux des serpents  tailleCellule = int(VAR.ECRAN\_X / dimensionX)  # --- Recal  positionX = int((VAR.ECRAN\_X-(dimensionX \* tailleCellule))/2)  positionY = int((VAR.ECRAN\_Y-(dimensionY \* tailleCellule))/2)  def afficher\_fond(\_fenetre):      # --- dessine un rectangle (un peu plus grand) vert sur tout l'espace du jeu      pygame.draw.rect(\_fenetre, VAR.COULEUR\_MUR, \  (positionX-4, positionY-4, (dimensionX \* tailleCellule)+8, (dimensionY \* tailleCellule)+8))      # --- dessine un rectangle noir sur tout l'espace du jeu      pygame.draw.rect(\_fenetre, VAR.COULEUR\_NOIR, (positionX, positionY, \  (dimensionX \* tailleCellule), (dimensionY \* tailleCellule)))  def afficher\_grille(\_fenetre):      for y in range(0, dimensionY):          for x in range (0, dimensionX):              posX, posY = positionX + (x \* tailleCellule), positionY + (y \* tailleCellule)              pygame.draw.rect(\_fenetre, VAR.COULEUR\_FOND\_ECRAN, \  (posX, posY , tailleCellule+1, tailleCellule+1), 1)  def afficher(\_fenetre):      afficher\_fond(\_fenetre)      afficher\_grille(\_fenetre) |

|  |
| --- |
| **variables.py** |
| ...  # --- Variables  boucle\_active = True  vitesse = 0.10  ~~tailleCellule = 16~~ |

|  |
| --- |
| **jeu.py** |
| while VAR.boucle\_active:      pygame.draw.rect(fenetre, VAR.COULEUR\_FOND\_ECRAN, (0, 0, VAR.ECRAN\_X, VAR.ECRAN\_Y))      # --- On controle les actions du joueurs.      for event in pygame.event.get():          if event.type == QUIT or event.type == KEYDOWN and event.key == K\_ESCAPE:              VAR.boucle\_active = False # Indique de sortir de la boucle.            if event.type == KEYDOWN:              if event.key == K\_RIGHT and not serpent.direction == "GAUCHE": serpent.direction = "DROITE"              if event.key == K\_LEFT  and not serpent.direction == "DROITE": serpent.direction = "GAUCHE"              if event.key == K\_UP    and not serpent.direction == "BAS":    serpent.direction = "HAUT"              if event.key == K\_DOWN  and not serpent.direction == "HAUT":   serpent.direction = "BAS"        terrain.afficher(fenetre)      serpent.se\_deplace()      serpent.afficher(fenetre)      # --- Actualise la fenêtre      pygame.display.flip() |



## Controle la sortie du terrain.

Notre serpent

* 1. Crées un nouveau fichier « **serpents.py** ».

*Maintenant, tu vas créer un nouveau type de variable. Ton type a toi, celui du serpent. Ce type est*

|  |
| --- |
| **Serpents.py** |
| def afficher(self, \_fenetre):          ...      def estCeQuiSort(self, xTete, yTete):          if xTete > terrain.dimensionX-1:              xTete = 0          elif xTete < 0:              xTete = terrain.dimensionX-1          elif yTete > terrain.dimensionY-1:              yTete = 0          elif yTete < 0:              yTete = terrain.dimensionY-1          return xTete, yTete      # --- Deplace notre serpent      def se\_deplace(self):  ... |

|  |
| --- |
| **serpents.py : se\_deplace()** |
| # --- Deplace notre serpent      def se\_deplace(self):          ...          elif self.direction == "GAUCHE":              xTete = xTete -1          else:              return None            xTete, yTete = self.estCeQuiSort(xTete, yTete)          self.corps[0] = (xTete, yTete) |

## Création des traces au passage du serpent.

Notre serpent

* 1. Crées un nouveau fichier « **serpents.py** ».

*Maintenant, tu vas créer un nouveau type de variable. Ton type a toi, celui du serpent. Ce type est « objet\_serpent ». Comme tout objet, il contiendra des variables et des fonctions son fonctionnement.*

|  |
| --- |
| **terrains.py** |
| import pygame  from pygame.locals import \*  import variables as VAR  grille = []  # --- Dimension du carré qui représente le terrain, ainsi que les morceaux des serpents  tailleCellule = int(VAR.ECRAN\_X / dimensionX)  # --- Recal  positionX = int((VAR.ECRAN\_X-(dimensionX \* tailleCellule))/2)  positionY = int((VAR.ECRAN\_Y-(dimensionY \* tailleCellule))/2)  def creer():      global grille      # --- Remplie chaque cellule de la grille avec une variable COULEUR composée de 3 valeurs (0,0,0)      grille = [ [(20,0,20) for j in range(dimensionY+1) ] for i in range(dimensionX+1) ]  def afficher\_fond(\_fenetre):  ...  def afficher\_grille(\_fenetre):  ...  def afficher\_traces(\_fenetre):      for y in range(0, dimensionY):          for x in range (0, dimensionX):              couleurCase = grille[x][y]              posX, posY = positionX + (x \* tailleCellule), positionY + (y \* tailleCellule)              pygame.draw.rect(\_fenetre, couleurCase, (posX, posY , tailleCellule, tailleCellule))  def afficher(\_fenetre):      afficher\_fond(\_fenetre)      afficher\_traces(\_fenetre)      afficher\_grille(\_fenetre)  def modifier(\_x, \_y):      global grille        v1,v2,v3 = grille[\_x][\_y]      if v1 < 256 - 15:          v1 = v1 + 15      else:          v1 = 0      grille[\_x][\_y] = (v1, v2, v3) |

|  |
| --- |
| **serpents.py : objet\_serpent()** |
| # --- Affiche notre serpent      def afficher(self, \_fenetre):          ...      def frotteLesFesses(self):          xTete, yTete = self.corps[0]          # --- Accentue la couleur du terrain au passage de la tete du serpent          terrain.modifier(xTete, yTete)      def estCeQuiSort(self, xTete, yTete):          ... |

|  |
| --- |
| **serpents.py : objet\_serpent().se\_deplace()** |
| elif self.direction == "GAUCHE":              xTete = xTete -1          else:              return None            xTete, yTete = self.estCeQuiSort(xTete, yTete)          self.corps[0] = (xTete, yTete)          self.frotteLesFesses() |

|  |
| --- |
| **jeu.py** |
| pygame.display.set\_caption(VAR.TITRE)  # --- Création du terrain  terrain.creer()  # --- Création de l'objet serpent  serpent = serpents.objet\_serpent()  # --- Boucle du jeu  VAR.boucle\_active = True  while VAR.boucle\_active: |

# Création de l’objet Pomme

|  |
| --- |
| **variables.py** |
|  |

|  |
| --- |
| **variables.py** |
|  |

|  |
| --- |
| **variables.py** |
|  |

|  |
| --- |
| **variables.py** |
|  |

# Mécanique

## Gestion de l’appétit du serpent

## Ajout du deuxième joueur

## Gestion des morsures

# Habillage

## Mise en place de l’interface

## Mise en place du menu

# Niveaux

## Mise en place du niveau

## Gestion du niveau suivant

## Ajout d’un message entre les niveaux