



Stages d'été - Python

Syllabus de l'étudiant

10 - 14 Juillet 2017

PIERRE-PAUL DE BREUCK
TANGUY LOSSEAU

TABLE DES MATIÈRES

A quoi s'attendre ?	i
0.1 Plan de la semaine	ii
Ordinateur et programme	iv
0.2 Pourquoi utiliser PYTHON ?	iv
0.2.1 Python est interprété	iv
1 A la découverte du Raspberry Pi	1
1.1 Anatomie	1
1.2 Système d'exploitation (OS)	1
1.3 Allumons le Raspberry !	2
2 Python en tant que calculatrice	3
3 Les types et opérations	4
3.1 A retenir	4
3.2 Exercices	4
4 Votre premier programme	5
4.1 A retenir	5
4.2 Exercices	5
5 Fonctions	6
5.1 A retenir	6
5.1.1 Fonctions :	6
5.1.2 Modules :	6
5.2 Exercices	6
5.3 Fonctions utiles	7
6 Dessinons !	8
6.1 Exercices	8
7 Conditionnel : if/else	10
7.1 A retenir	10
7.2 Exemple	11
7.3 Exercices	11
8 Les boucles	12
8.1 A retenir	12
8.2 Exercices	13

9 Listes	14
9.1 A retenir	14
9.2 Exercices	14
10 Dictionnaires	15
10.1 A retenir	15
10.2 Exercices	15
11 Classes et objets	16
11.1 A retenir	16
11.2 Exercices	16
12 Projet code snake : Pygame	17
12.1 Introduction	17
12.2 Création d'une fenêtre	17
12.3 Interraction avec la fenêtre	18
12.4 Dessiner dans la fenêtre	19
12.5 Full Snaky Python	21
13 Station météo	24

A QUOI S'ATTENDRE ?

Durant ce stage, nous allons aborder les points suivant :

1. Nous allons étudier le langage de programmation **PYTHON**. Ce langage va permettre aux étudiants d'interagir avec le raspberry pi et sera utile pour atteindre les objectifs fixés par le stage. Nous reprendrons les bases - que les élèves sont supposés maîtriser -, puis nous approfondirons la matière progressivement. De nombreux exercices accompagneront l'élève durant cet apprentissage.
2. Nous découvrirons également comment intéragir avec le **raspberry pi**. D'une part nous verrons les bases de linux, le cœur du système d'exploitation du raspberry. D'autre part on verra comment utiliser les pins GPIO permettant de connecter par exemple un senseur de température à notre mini ordinateur.
3. En fin de semaine nous réaliserons une petite *station météo* ou un *jeu vidéo* sur le raspberry pi.



Vous disposez chacun d'un raspberry pi permettant de résoudre les exercices proposés. Cependant n'hésitez pas à dialoguer avec vos voisins concernant votre solution. En informatique, nous appelons cette technique le **pair programming**. Une telle méthode de travail permet à la fois de rendre l'apprentissage plus agréable et plus efficace.

Syllabus

Ce syllabus est destiné à des élèves de 12 à 18 ans assistant aux stages d'été organisé par Technofutur TIC. Ce syllabus est un complément aux slides et aux discussions proposées durant les séances de ce stage. Il doit être complété par l'élève selon ses notes personnelles et les exercices réalisés durant la semaine.

Le syllabus a été écrit sous forme d'activités. Chaque activité peut être vue comme un exercice permettant de découvrir la matière. Cela permet d'apprendre la théorie de façon ludique et pratique.

0.1 Plan de la semaine

Ce plan est donné à titre informatif, il pourra être modifié selon les circonstances et les envies du groupe.

Lundi	Mardi	Mercredi	Jeudi	Vendredi
Introduction au Raspberry, Premier programme	Python <u>Types</u> Fonctions	Python <u>Conditions</u> Boucles	Coder un jeu : PyGame	Station météo

ORDINATEUR ET PROGRAMME

Avant de se lancer dans les activités il est important de comprendre la notion d'ordinateur et programme.

Ordinateur Wikipedia nous fournit la définition suivante : *Un ordinateur est une machine électronique programmable qui fonctionne par la lecture séquentielle d'un ensemble d'instructions, organisées en programmes, qui lui font exécuter des opérations logiques et arithmétiques.* Un ordinateur peut donc être vu comme une machine qui exécute des instructions les unes après les autres. Ces instructions peuvent être des lectures de clavier ou souris ou encore une écriture de fichier. Cependant une question persiste : comment dire à mon ordinateur la suite d'instruction qu'il doit exécuter ? Cela ce fait grâce à un language de programmation.

Language de programmation Un langage qui permet de décrire, selon une **syntaxe** (forme) et une **sémantique** (sens), un **programme** (suite d'instructions) qu'un ordinateur doit exécuter.

Grâce à ce language nous pouvons écrire sous forme de texte ce que doit faire l'ordinateur. C'est ce qu'on appelle le **programme**. Le texte du programme est appelé le **code**.

Durant cette semaine de stage, nous allons utiliser un raspberry pi comme *ordianiteur*. Oui, le raspberry pi n'est rien d'autre qu'un très petit ordinateur. Un seul langage de programmation sera utilisé : PYTHON.

0.2 Pourquoi utiliser Python ?

Il existe une multitude de langages de programmation : JAVA, Ruby, C, C++, Python, Scala,...

Toutefois, PYTHON possède plusieurs atouts pour les débutants :

- **Lisibilité** : il est proche d'un langage humain (anglais).
- **Modules** : Python contient de nombreux modules facilitant la programmation.
- **Simple** : Pas de compilation, typage dynamique, une syntaxe succincte, ...
- **Fortement documenté, gratuit et accessible**
Python est donc idéal pour débuter.

0.2.1 Python est interprété

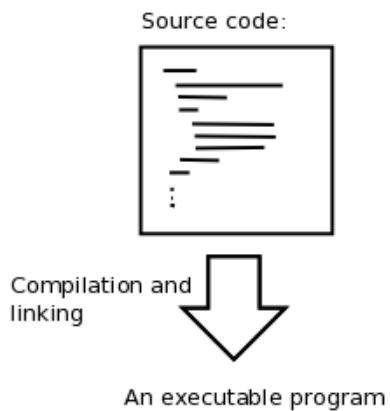
Il existe principalement deux 'familles' de laguages : les languages compilés et interprétés.

Language compilé Le programme, sous forme de texte (language python), est convertit en language machine par un programme appelé 'compilateur'. On passe donc d'un fichier text à un executable qui peut directement être exécuté par l'ordinateur.

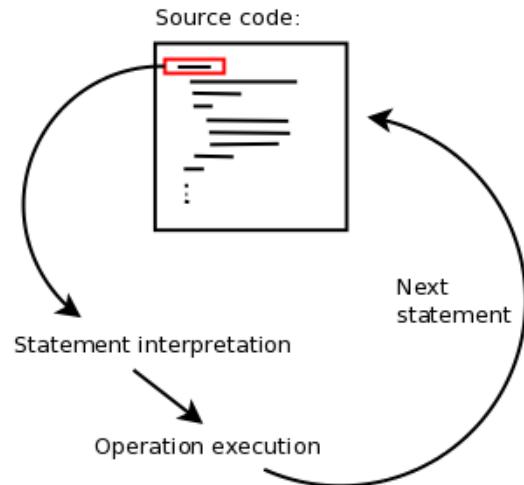
Langage interprété Une fois qu'on veut exécuter notre programme, sous forme de texte, on le passe à un autre programme appelé 'interpréteur'. Le code (texte) est lu ligne par ligne par l'interpréteur et c'est lui qui donne les ordres à l'ordinateur. Le fichier texte n'est donc jamais compilé.

Python est un langage interprété.

Compilation



Interpretation



ACTIVITÉ 1

A LA DÉCOUVERTE DU RASPBERRY PI

1.1 Anatomie

Comme dit précédemment le raspberry pi n'est rien d'autre qu'un mini ordinateur. On connecte donc souvent un écran, une souris et un clavier au raspberry. En plus de cela il existe d'autres connectiques, voir figure 1.1, que nous verrons ensemble.

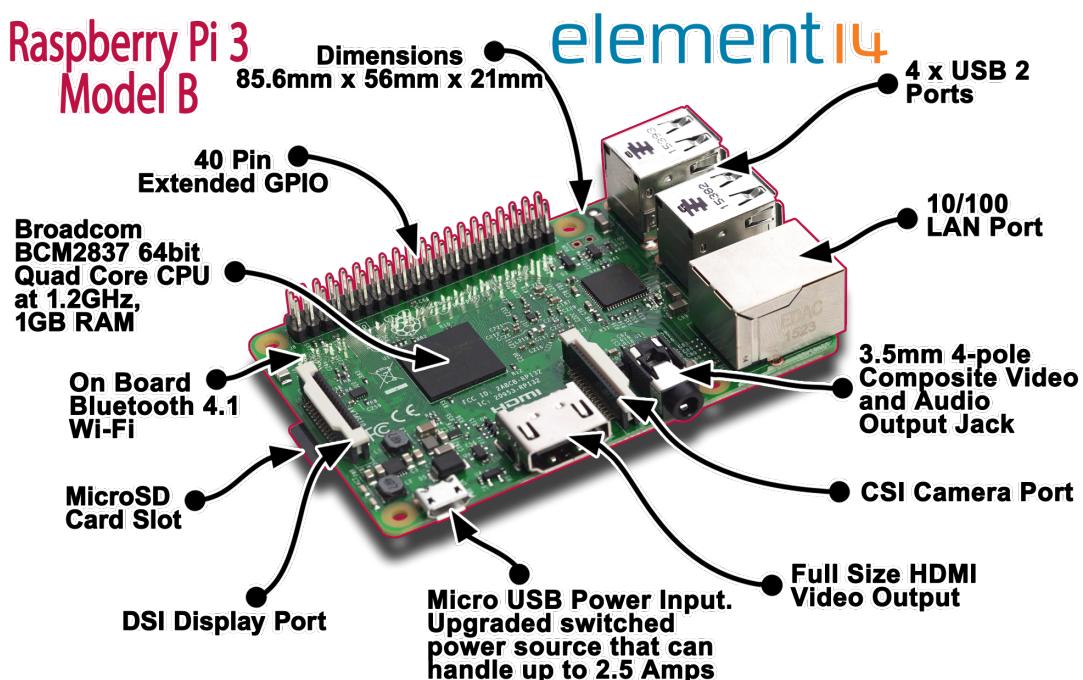


FIGURE 1.1 – Anatomie du Raspberry Pi 3 B - *image tiré de element14*

1.2 Système d'exploitation (OS)

Le raspberry pi, comme tout ordinateur a besoin d'un système d'exploitation. Il est chargé de faire la liaison entre le matériel (hardware), le logiciel (software) et l'utilisateur. Il s'occupe entre autre de la gestion du processeur, mémoire vive ou encore les entrées/sorties.

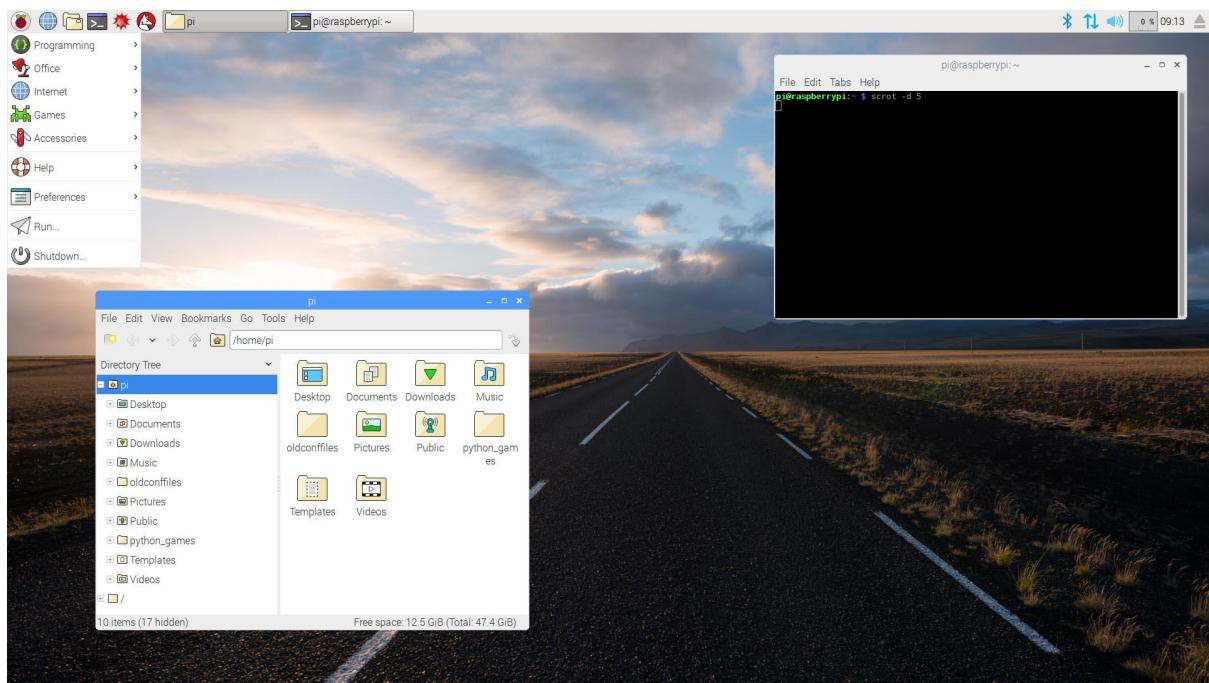
Windows est un exemple de système d'exploitation le plus utilisé sur les ordinateurs personnels. D'autres exemples sont Mac OS ou d'autres variantes basées sur Unix (Ubuntu, Mint,...). Il existe une multitude d'OS installable sur Raspberry, nous utiliserons Raspbian, basé sur Linux. Il est gratuit et peut être facilement installé sur la carte SD du Raspberry. Voir <https://www.raspberrypi.org/downloads/raspbian/>

1.3 Allumons le Raspberry !

1. Insérez la carte microSD avec l'OS préinstallé dans le Raspberry si ce n'est déjà fait.
2. Branchez l'écran au port HDMI, la souris et le clavier aux ports USB.
3. Pour allumer le Raspberry Pi, branchez l'alimentation (Micro USB).
4. Le terminal (ou console) apparaît à l'écran. C'est grâce au terminal qu'on peut communiquer avec l'ordinateur.
5. Si demandé, entrez le nom d'utilisateur et mot de passe suivant :

1 User : pi
2 Password: raspberry

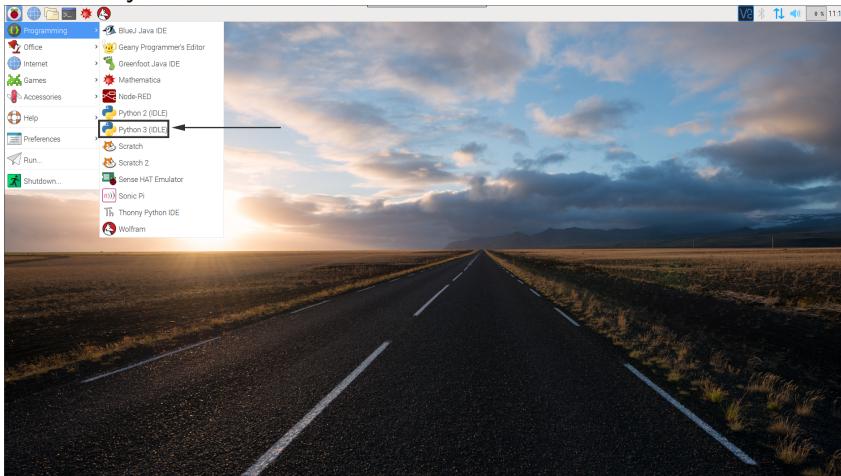
6. Votre raspberry est fonctionnel ! Faites le tour des applications pour vous habituer.



ACTIVITÉ 2

PYTHON EN TANT QUE CALCULATRICE

1. Ouvrez Python 3 IDLE



2. Entrez 3+2. Observez le résultat.

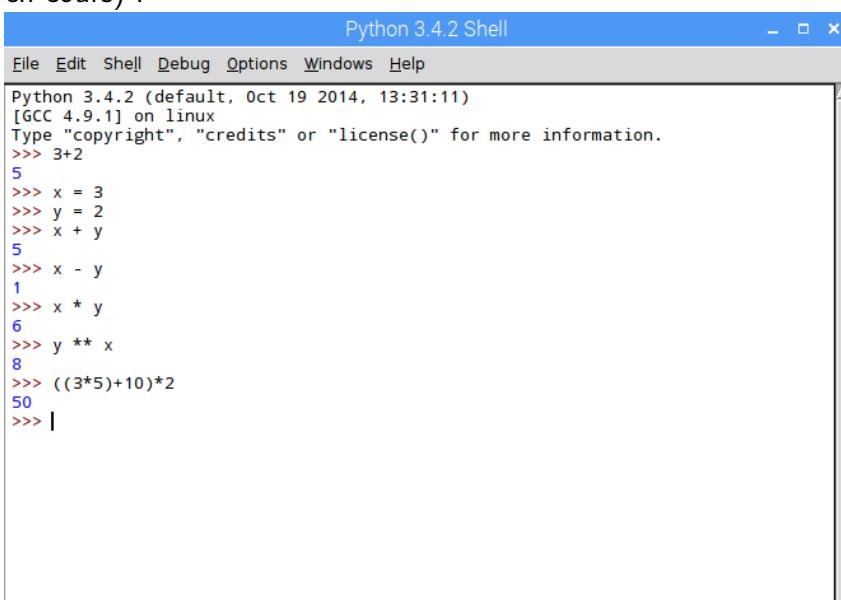
3. Créez une variable x et égalez celle ci à 3.

1 x = 3

4. Créez une variable y initialisée à 2 :

1 y = 2

5. Faites quelques calculs comme illustré à la figure ci-dessous (ou encore ceux montrés en cours) :

A screenshot of the Python 3.4.2 Shell window. The title bar says "Python 3.4.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main window shows the Python interpreter's prompt (">>>>). The user has entered several commands:

```
Python 3.4.2 (default, Oct 19 2014, 13:31:11)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.

>>> 3+2
5
>>> x = 3
>>> y = 2
>>> x + y
5
>>> x - y
1
>>> x * y
6
>>> y ** x
8
>>> ((3*5)+10)*2
50
>>> |
```

The output of the last command, "50", is shown in orange.

ACTIVITÉ 3

LES TYPES ET OPÉRATIONS

1. Suivez le tutoriel sur les types et opérations donné lors du cours.

3.1 A retenir

Une variable commence toujours par une lettre (majuscule ou minuscule).

```
1     #types
2     String1 = "Bonjour!" #String1 est une variable, "Bonjour!" est la valeur.
3     String1 = 'Bonjour' #Autre manière de définir un string
4     x = 3 #entier
5     y = 3.0 #float (pas la même chose!)
6
7     type(2) # fonction rentrant le type, ici c'est un 'int'
8
9     #opérations sur les int et floats
10    i = 2 + 3
11    i = 3 / 2 #i sera égal à 1.5
12    i = 3 // 2 # division entière. i sera égal à 1 (et non 1.5) !
13    i = 7 - 2
14    i = 2 * 10
15    i = 5 ** 2 # exposant, i = 25
16
17    #opérations sur les strings
18    s = "Hello" + " World!" #concaténation
19    s = "bla" * 5 # s sera blablablabla
20    print("Bonjour!") #affiche 'Bonjour!'
```

3.2 Exercices

A vous de jouer! Lancez le mode interactif de Python 3 comme l'activité précédente. Commencez par retapper les exemples ci dessus. Ensuite faites les exercices ci-dessous :

1. Créez une variable avec la valeur "Bon". Créez une autre variable avec la valeur "jour". Avec cela affichez "Bonjour" avec une opération. Ensuite affichez "Bonjour Bonjour" grâce à l'opérateur '*'.
2. Calculez la surface d'un cercle de rayon $r = 2.17\text{m}$.

ACTIVITÉ 4

VOTRE PREMIER PROGRAMME

Il est temps de créer votre premier script !

1. Suivez le tutoriel donné lors du cours.

4.1 A retenir

Dans Python 3 IDLE, créez un nouveau script : File -> New File. Ensuite enregistrez le : File -> Save. Pour exécuter le script : appuyer sur 'F5'.

```
1  in = input('entrez un chiffre') #permet de lire le clavier.  
2  x = int("3") # convertie le string "3" en entier 3.
```

4.2 Exercices

A vous de jouer !

1. Expliquez ce que fait le programme ci dessous :

```
1  print('Bonjour')  
2  x = input('entrez un chiffre: ')  
3  y = int(x)  
4  
5  print(y*2) # que se passe-t-il quand y est remplacé par x?
```

2. Créez un programme affichant "Hello World!"
3. Créez un programme calculant la surface d'un cercle, le rayon sera demandé lors de l'exécution à l'utilisateur.

ACTIVITÉ 5

FONCTIONS

1. Ecoutez bien le tutoriel donné lors du cours.

5.1 A retenir

5.1.1 Fonctions :

```
1 # 1.a definition d'une fonction
2     def nomDeMaFonction(param):
3         """Une description de ma fonction """
4         incr = param + 1 # le corps de la fonction est indenté (utilisez tab)!
5         return incr
6
7 # 1.b appel à la fonction
8     y = nomDeMaFonction(4) # y=5
```

5.1.2 Modules :

```
1 # 2 modules
2     import math
3     monPi = math.pi
4     x = math.cos(2*pi)
5
6 # 2.b variantes des modules
7     from math import *
8     monPi = pi # il est plus nécessaire de faire math.(...)
9     x = cos(2*pi)
```

5.2 Exercices

1. Expliquez ce qu'il s'affiche quand la ligne 9 est interprétée :

```
1     def print_lyrics():
2         print("I'm a lumberjack, and I'm okay.")
3         print("I sleep all night and I work all day.")
4
5     def repeat_lyrics():
6         print_lyrics()
7         print_lyrics()
8
9     repeat_lyrics()
```

2. Ecrivez une fonction 'right_justify' qui prend en paramètre un string 's' et qui affiche le string précédé d'assez d'espaces tel que le dernier caractère s'affiche dans la colonne 70.

Astuce : utilisez la concaténation. Egalement, Python fournit la fonction len() qui retourne la longueur d'un string, par exemple len('monty') retourne 5.

5.3 Fonctions utiles

Nous vous présenterons certaines fonctions très utiles, intégrées dans Python. Notez les dans le cadre ci-dessous :

ACTIVITÉ 6

DESSINONS!

1. Suivez le tutoriel sur le module 'turtle'. Notez les fonctions de base ci dessous :

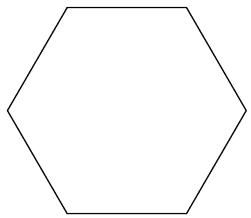
Python permet de répéter certaines actions grâce à des boucles :

```
1   for i in range(4):
2       print('Hello') #tab!
```

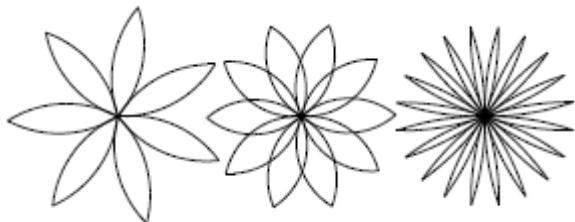
6.1 Exercices

1. Dessinez un carré grâce de coté 100. Faites le d'abord sans boucle ensuite avec boucle.
2. Créez une fonction prenant en argument : une tortue t, un entier n et une longueur l. Elle dessine ensuite un polygone régulier avec n cotés de longueur l grâce à cette tortue.

Voici le résultat pour `polygone(t,6,50)` :



3. Bonus : A partir de la fonction précédente, créez une fonction dessinant une cercle de rayon r, fournit en argument.
4. Amusez-vous en dessinant d'autres formes ! Nous organiserons un petit concours avec les plus beaux dessins :-) Quelques idées :



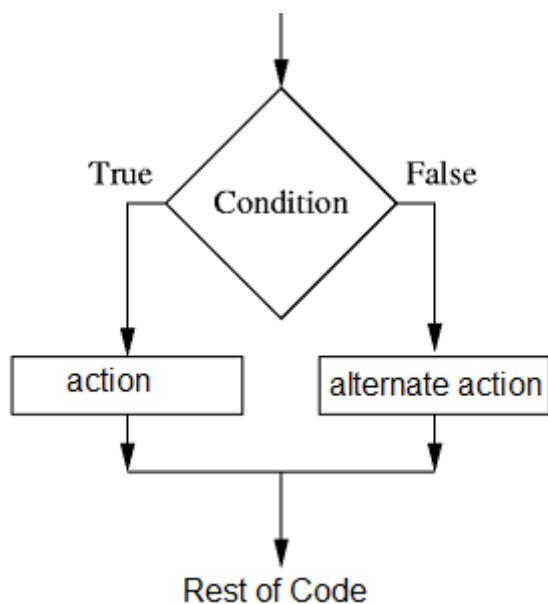
ACTIVITÉ 7

CONDITIONNEL : IF/ELSE

Il est temps d'ajouter un nouveau concept au langage : la structure if/else.

1. Nous vous présenterons les bases en classe.

7.1 A retenir



```
1 #syntaxe
2 if (condition1):          # SI
3     print("la condition1 est vraie")
4 elif (condition2):        # SI AUTRE CHOSE
5     print("la condition2 est vraie")
6 else:                     # SINON
7     print("Condition1 et condition2 sont fausses")
8
9
10 #conditions possibles:
11
12     x == y
13     x != y # x n'est pas égal à y
14     x > y
15     x < y
16     x >= y
17     x <= y
18
19
```

```
20     #opérateurs logiques
21
22         or
23         and
24         not
```

7.2 Exemple

```
1
2 def pos_not1(nombre):
3     """ affiche "vrai" si nombre est positif et non égal à 1.
4     Affiche "faux" sinon."""
5
6     if (nombre > 0) and (nombre != 1):
7         print("vrai")
8     else:
9         print("faux")
10
11 pos_not1(5) # "vrai"
```

7.3 Exercices

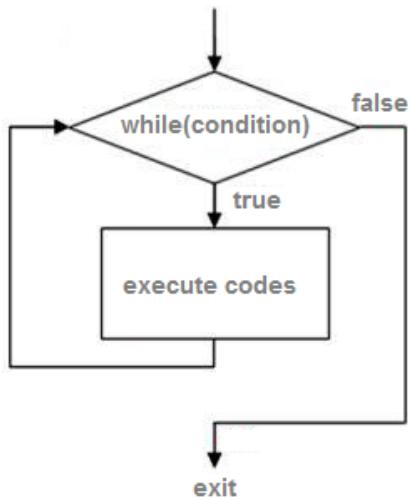
1. Implémentez une fonction affichant "vrai" si l'argument est pair, "faux" sinon. L'opérateur `%` vous sera utile. (`x % y`) retourne le reste après division de `x` par `y`. Par exemple (`13 % 4`) retourne 1.
2. Implémentez une fonction 'max' retournant le maximum entre deux entiers.
3. Implémentez une fonction 'abs' retournant la valeur absolue d'un entier.

ACTIVITÉ 8

LES BOUCLES

1. Nous vous présenterons les bases en classe.

8.1 A retenir



```
1      #syntaxe boucle for
2  for <variable> in <sequence>:
3      <statements>
4  else:
5      <statements>
6
7      #exemples
8  for x in range(1,10): # le 10 n'est pas compris!
9      print(x)
10
11 for character in "Bonjour":
12     print(character)
13
14
15      #syntaxe boucle while
16  while <condition>:
17      <statement>
18
19      #exemple
20  x = 0
21  while x < 10:
22      print(x)
23      x = x + 1
```

8.2 Exercices

1. Enumerez tout les multiples de 3 avec un petit programme.
2. Enumerez toutes les puissances de 2.
3. Implémentez un programme affichant les valeurs (x, y, z) tel que $x^2 + y^2 = z^2$. Faites le d'abord avec une boucle explicite ensuite par récursion.
Bonus : changez l'exposant 2 en une valeur plus grande que 2, par exemple 3. Vous observez qu'aucune solution est trouvée. C'est le dernier théorème de Fermat.
4. Implémentez un programme énumérant les nombres premiers.

ACTIVITÉ 9

LISTES

Nous verrons ensemble une nouvelle structure de données : les listes.

9.1 A retenir

```
1 list1 = [1, 2, 3, 4, 5, 'a', 'b']
2 list2 = ['c', 'd', 'e']
3
4 print(list1[2]) # accéder aux éléments, attention les indices commencent à 0.
5 list1[0] = 10
6 'a' in list1 # True
7 del list1[2]
8
9 for x in list1:
10     print(x)
11
12 list3 = list1 + list2
13 list4 = list1[1:3]
14 list5 = list1[:3]
15
16 #fonctions utiles
17
18 list1.append('c')
19 list1.sort()
20 x = list1.pop(1)
```

9.2 Exercices

1. Soit t une liste. Que fait $t[2:]$? Et $t[:]$?
2. Que se passe-t-il pour des indices négatifs. Par exemple $t[-1]$?
3. Implémentez une fonction retournant le dernier élément d'une liste.
4. Ecrivez une fonction retournant la somme des entiers d'une liste prise en argument.
5. Implémentez une fonction $isSorted(t)$ retournant 'True' (True est de type boolean et non un string!), si t est triée, 'False' sinon. Aide : t est triée si pour tout i , $t[i] < t[i+1]$.

ACTIVITÉ 10

DICTIONNAIRES

Nous verrons ensemble une nouvelle structure de données : les dictionnaires.

10.1 A retenir

```
1 dict1 = {'animal':'chien', 'nom' : 'Booly'}
2 dict2 = {'pomme': 3, 'banane': 6}
3
4 print(dict1['animal']) # accéder aux éléments
5 dict1['race'] = 'carlin' #ajouter un élément
6 'animal' in list1 # True
7
8 for x in dict1:
9     print(x)
10
11
12 #fonctions utiles
13
14 dict1.item() # retourne un liste des (key, value)
15 dict1.keys() # retourne une liste des 'keys'
16 dict1.values() # retourne une liste des 'values'
```

10.2 Exercices

1. Soit dict = 'Couleur' :'Rouge' . Que fait : del dict['Couleur'] ?
2. Implémentez une fonction 'freq' recevant une liste de strings, retournant un dictionnaire contenant la fréquence de chaque string dans la liste.
Exemple *freq(["Bonjour", "Hola", "Bonjour"])* donne {"Bonjour" : 2, "Hola" : 1}.

ACTIVITÉ 11

CLASSES ET OBJETS

11.1 A retenir

```
1 import math
2
3 class Point():
4     def __init__(self, x = 0, y = 0):
5         self.x = x
6         self.y = y
7     def distance(self):
8         return math.sqrt(self.x**2+self.y**2)
9
10 myPoint = Point(2,3)
11 dist = myPoint.distance()
12 print(dist)
```

11.2 Exercices

1. Reprenez l'exemple ci-dessus. Ajoutez un méthode delta(self,dx,dy), ajoutant dx à x et dy à y.
2. Informez vous sur la méthode `__str__()`. Proposez une implémentation possible pour la classe Point.
3. Proposez une implémentation possible d'une classe représentant un rectangle.

ACTIVITÉ 12

PROJET CODE SNAKE : PYGAME



12.1 Introduction

Maintenant que vous connaissez les bases de Python, nous pouvons nous mettre d'accord : la console, c'est bien... mais les graphiques, c'est mieux ! Nous allons voir maintenant comment créer et gérer une interface graphique à l'aide de Pygame. Cette librairie présente plusieurs avantages :

- compatibilité avec les versions récentes de python
- portable sur les OS principaux : windows, mac, distributions linux grand public...
- une grande communauté d'utilisateur : en cas de problème, vous êtes sur de trouver une solution via google.

Pygame permet de gérer des périphériques (clavier, souris), l'audio, et l'affichage vidéo 2D. Par exemple, voici des images de jeu réalisés avec pygame :



12.2 Création d'une fenêtre

Avant de créer une fenêtre, nous devons bien entendu importer d'abord la bibliothèque.

```
1 import pygame
2 from pygame.locals import *
```

Vous n'êtes sûrement pas surpris par "import pygame", fin connasseurs que vous êtes de python. La ligne "from pygame.locals import *" sert à importer les constantes de Pygame, de

sorte à pouvoir y accéder en tapant "constante" au lieu de "pygame.constante" !

Enfin, avant de faire quoi que ce soit avec cette librairie, on doit l'initialiser ainsi :

```
1 pygame.init()
```

Voilà ! Vous êtes paré pour ouvrir votre première fenêtre.

```
1 fenetre = pygame.display.set_mode((800,800))
2 input('Appuyez sur entrée pour fermer la fenêtre.')
```

La fonction "pygame.display.set_mode" initialise et renvoie une fenêtre. Elle prends en argument un tuple de deux int : la largeur/hauteur de la fenêtre créée.

Une fois que la fenêtre est créée, le programme continue. Pour pauser son exécution et vous laisser admirer le résultat, on a rajouté dessous un appel à input : dès que vous appuyez sur entrée, le programme se finit et la fenêtre se ferme.

12.3 Interraction avec la fenêtre

Pour le moment, c'est chouette : on a une fenêtre vide, qui ne fait rien. Mais comment interagir avec elle ? On procède de la suite.

```
1 import pygame
2 from pygame.locals import *
3
4 pygame.init()
5 fenetre = pygame.display.set_mode((300,300))
6
7 en_jeux = True                      #On est en train de jouer
8 while en_jeux:
9     for evenement in pygame.event.get(): #On parcours la liste des nouveaux événements
10         if evenement.type == QUIT:       #Si on a demandé à la fenêtre de se fermer
11             en_jeux = False            #On sort de la boucle
12         if evenement.type == KEYDOWN and evenement.key == K_DOWN:
13             en_jeux = False            #Si on a appuyé la flèche 'bas'
14             #On sort de la boucle
```

Miracle ! La fenêtre se ferme lorsque l'on appuie sur la croix. Mais que c'est-il passé ?

Les événements Pour gérer les interactions avec la fenêtre, pygame définit des "événements", qui correspondent à des actions/demandes de l'utilisateur : fermer la fenêtre, détecter quelle touche du clavier a été appuyée...

Des que l'utilisateur effectue une action, pygame sauvegarde dans une liste l'évènement associé. La fonction " pygame.event.get()" renvoie cette liste, et en crée une nouvelle dans laquelle les nouveaux événements seront sauvegardés, de sorte que deux appels consécutifs à cette fonction ne renvoie pas des événements déjà renvoyés.

Les types d'événements Pygame définit des événements pour la souris (click gauche/droit à telle position...), la fenêtre(fermer, changer de taille...), le clavier (touche appuyée...)... Pour créer notre jeu, nous n'avons besoins de récupérer que les clefs appuyées du clavier : ce sont les events de type "KEYDOWN". On accède à la touche appuyée via "evenement.key". Nous aurons besoins notamment des clefs :

- K_DOWN, flèche du bas
- K_UP, flèche du haut
- K_LEFT, flèche vers la gauche
- K_RIGHT, flèche vers la droite

12.4 Dessiner dans la fenêtre

```
1 import pygame
2 from pygame.locals import *
3
4 pygame.init()
5 fenetre = pygame.display.set_mode((300,300))
6
7 fenetre.fill((255,255,255))
8 pygame.display.flip()
```

L'instruction "fenetre.fill" remplis la fenêtre avec la couleur donnée en argument, au format RGB.

Le format RGB Pour rappel, toute couleur est un mélange des trois couleurs primaires : Rouge(Red), Vert(Green) et Bleu(Blue). Le format RGB est composé des entiers (R, G, B), chacun compris entre 0 et 255, indiquant la concentration en la couleur primaire correspondant de la couleur finale. Ainsi,

- (0,0,0) correspond au noir ;
- (255,255,255) au blanc ;
- (255,0,0) au rouge pur ;
- etc...

Afficher l'image générée Les opérations de dessins ne sont pas directement affichées sur la fenêtre : pour rafraîchir la fenêtre, on utilise : "pygame.display.flip()"

Afficher et faire bouger un rectangle On peut créer un rectangle de la sorte :

```
1 import pygame
2 from pygame.locals import *
3
4 pygame.init()
5 fenetre = pygame.display.set_mode((300,300))
6 fenetre.fill((255,255,255))
7
8 rectangle = pygame.Surface((10,10))    #Creation du rectangle
9 rectangle.fill((100,50,50))           #On colorie le rectangle
10 fenetre.blit(rectangle, (0,0))        #On dessine le rectangle
11 pygame.display.flip()
```

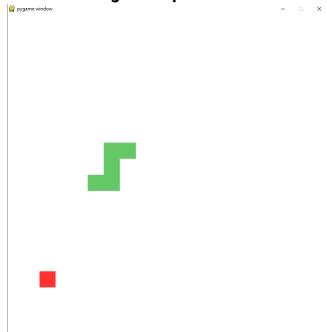
Quelques explications :

- pygame.Surface retourne une surface, dont les dimensions selon les axes x et y sont passées en arguments.
- rectangle.fill((r,g,b)) remplis la surface rectangle avec la couleur donnée en argument.
- fenetre.blit(surface, position) dessine la surface passée en argument à la position donnée.

Conclusion Félicitation ! Vous savez désormais créer une fenêtre, réagir aux inputs d'un joueur, et dessiner des rectangle. Il est temps de vous faire coder votre premier jeu : Python ! Un snake basique.

12.5 Full Snaky Python

Voici le jeu que vous allez coder :



Un snake. Tout petit ! Tout mignon ! Le... Full Snaky Python !!!

Dans cette partie, nous allons vous laisser une plus grande marge de manœuvre : maintenant que vous connaissez le minimum de théorie, vous devriez facilement pouvoir coder ce jeu :)

Les règles du jeu Avant d'écrire ne serait-ce qu'une ligne de code, il est important d'écrire les règles du jeu. Quel est le but du jeu ? Sous quelle condition on a gagné le jeu/perdu ? Quels sont les contrôles de l'utilisateur ? Ecrivez ces règles ci-dessous !

Sur la fenêtre du jeu Pour ce jeu, nous assimilerons le 'Full Snaky Python' à un ensemble de carré de la même dimension que la pomme à manger. Comment faire en sorte que la fenêtre de jeu occupe juste la bonne surface pour afficher au maximum fenetre_taille_x carrés en

abscisse, et fenetre_taille_y carrés en ordonnée, en prenant comme base un carré de coté dim_carre ? A vous de jouer :)

Le Full Snaky Python Dans un premier temps, nous allons créer un joueur basique : rien qu'un rectangle, qui bouge à intervalle régulier, réagis au contrôle de l'utilisateur. Quelques points importants :

- Dans un premier temps, contentez vous de créer un carré de dimension 'dim_carre'
- Pour rendre le code plus lisible, on recommande d'assigner à chaque direction un nombre dans une variable, par exemple : DIRECTION_BAS = 0.
- Pour le moment, la boucle while s'exécute aussi vite que possible... Votre personnage risque donc de bouger à la vitesse de l'éclair. Pour pallier à ce problème, on 'pause' à chaque itération le programme durant delais_affichage millisecondes, de la sorte :

```
1  delais_affichage = 500
2  pygame.time.delay(delais_affichage)
```

La pomme Le but du jeu est de manger autant de pommes que possibles... Ce serait bien d'afficher une pomme, non ? :p

Dans un premier temps, contentez-vous d'afficher une pomme à une location aléatoire, sur la surface de jeu. Vous aurez besoins de :

```
1  import random
2  random_number = random.randint(0,fenetre_taille_x-1)
```

En sachant que 'r = random.randint(min_value,max_value)' génère un nombre aléatoirement tel que $min_value \leq r \leq max_value$.

Ensuite, à chaque fois que le Full Snaky Python mange une pomme, celle ci doit apparaître à une nouvelle location. Pour détecter la collision entre le Full Snaky Python et la pomme, vous aurez besoins de cette fonction :

```
1  #Fonction de detection de collisions
2  def collision (position_0, position_1):
3      if position_0[0] + carre_dim > position_1[0] and position_0[0] < position_1
4          [0] +carre_dim and position_0[1]+carre_dim > position_1[1] and
5              position_0[1]< position_1[1] + carre_dim:
6                  return True
7      else:
8          return False
```

Animer le Full Snaky Python On ne vous prends pas pour des poires : vous savez sûrement que notre snake peut manger des pommes, mais pour des prunes. En effet, il ne grandit pas en mangeant des fruits ! Pour ce faire, nous allons remplacer le tuple snake_pos par un array de tuple ! On dessinera un carré à l'emplacement de chaque tuple. Dans un premier temps,

- remplacez le tuple snake_pos par un array, et adaptez le reste du code pour accepter cet array.
- si le Full Snaky Python mange une pomme, ajouter un tuple vide au serpent avant la fonction de mouvement

— enfin, dessiner une boîte à chaque position indiquée dans l'array

Appliquer les règles du jeu Enfin, vous pouvez vérifier à chaque itération de la boucle while que le joueur respecte les règles du jeu, et n'a pas perdu. Dans un premier temps, vérifiez que le Full Snaky Python est toujours dans la zone de jeu, puis vérifiez qu'il n'est pas en train de manger son propre corps.

Menu et fin de jeu Faites un menu de jeu en console, qui demande une difficulté(en fonction de laquelle vous pouvez modifier la vitesse de déplacement du serpent, la taille de la zone de jeu, etc...).

Dans le cas où la personne a perdu/gagné, écrivez un message d'encouragement(GITGUD) ou de félicitation, et proposez de relancer le jeu.

Touches finales Pour jouer un son avec pygame, vous pouvez procéder de la sorte :

```
1 son = pygame.mixer.Sound("coins.wav") #Préparer le son
2 son.play() #Jouer le son
```

Faites en sorte qu'un son soit joué chaque fois que le python mange une pomme. Ajoutez aussi un son de défaite/victoire.

De même, essayez de mettre un peu de couleur sur notre Full Snaky Python : par exemple, colorer 1 case sur deux avec des couleurs différentes,...

ACTIVITÉ 13

STATION MÉTÉO

Un capteur de température et d'humidité vous est fourni. Nous vous apprendrons à l'utiliser lors d'un **tutoriel interactif** en classe. Vous pouvez noter, si nécessaire, les informations utiles ci-dessous :