Python – Fiche n°1
Installer et utiliser Python et Pygame

Python peut être téléchargé depuis cette adresse :
http://www.python.org/download/
Pygame peut être téléchargé depuis cette adresse :
http://www.pygame.org/download.shtml

En ce qui concerne python, préférez la version 3 à la version 2. Pour pygame, choisissez une version se terminant par le même numéro de version majeure que python, par exemple pygame-1.9.2a0.win32-py3.2.msi pour la version 3 de python.
Si la version la plus récente de pygame ne correspond pas à la version la plus récente de python, vous pouvez retrouver les versions précédentes de python sur la page de téléchargement en cliquant sur « Releases » dans le menu à gauche.

Installation sous Windows

Installer python :
- Double-cliquer sur le fichier python-3.2.0.msi (ou la version téléchargée)
- Choisir « Install for all users » puis cliquer sur « Next » (suivant)
- Garder le répertoire par défaut (C:\Python33) et cliquer sur « Next »
- Garder la selection de paquets par défaut et cliquer sur « Next »
- Une fois l'installation terminée, cliquer sur « Finish »

Installer pygame :
- Double-cliquer sur le fichier pygame-1.9.2a0.win32-py32.msi (ou la version téléchargée)
- Choisir « Install for all users » puis cliquer sur « Next »
- Ne pas changer l'endroit où Python est installé et cliquer sur « Next »
- Une fois l'installation terminée, cliquer sur « Finish »

Installation sous GNU/Linux

Recherchez dans le gestionnaire de paquet de votre distribution quelles versions de python et de pygame sont disponibles.

Si pygame pour python3 est disponible, installer python3 et pygame pour python3. Si pygame n'est disponible que pour python2, installer python2 et pygame pour python2.

Installation sous Mac

Python est installé par défaut sur Mac OS X. Il est possible d'installer une version plus récente depuis le lien de téléchargement fourni ci-dessus.

Depuis la page de téléchargement de pygame, sélectionner la version correspondant à la version de python installée. Par exemple, pygame-1.9.1release-py2.6-macosx10.5.zip pour Python 2.6.

Utiliser python

Lancer IDLE.
Au premier lancement, seule la fenêtre « Python Shell » s'ouvre. C'est une fenêtre dans laquelle on peut exécuter des commandes Python.
Essayez de taper « 2+2 ». Le résultat devrait être proche de ceci :

```
Python 3.3.0 (default, Dec 22 2012, 21:14:17)
[GCC 4.7.2] on linux
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>>
```

Il est pratique de pouvoir exécuter des commandes python et de voir le résultat immédiatement. Cependant, si l'on veut pouvoir ré-exécuter son programme à volonté et le conserver, il faut passer le sauvegarder dans un fichier.

Cliquez sur « File » → « New Window » ou pressez Ctrl+N. Ecrivez 2+2 puis cliquez sur « Run » → « Run module » ou pressez F5. Avant de pouvoir exécuter votre programme, vous devez le sauvegarder. Choisissez un répertoire et un nom pour le sauvegarder.

Vous pouvez voir les lignes suivantes dans la fenêtre « Python Shell » :

```
>>> ================================ RESTART ================================
>>>
>>>
```

La ligne RESTART indique que le programme a été lancé. Mais rien ne s'est affiché ! Lorsque l'on exécute un programme Python, le résultat des commandes ne s'affiche pas, contrairement à lorsque l'on exécute les commandes directement dans le shell python.
Modifiez le fichier pour remplacer la ligne « 2+2 » par « print(2+2) » puis relancez le programme. Le résultat est le suivant :

```
>>> ================================ RESTART ================================
>>>
4
>>>
```

Cette fois, la commande print fait que le résultat est affiché.

Quelques liens en français sur Python :

- http://www.olivierberger.org/python/
- http://wiki.python.org/moin/FrenchLanguage
- http://the3fold.free.fr/index.php?src=contenu/Langages/Python

Python – Fiche n°2
Écrire, lire, variables, conditions et boucles

```
prenom = input("Bonjour, comment t'appelles-tu ? ")
print("Enchanté ", prenom, " je suis content de faire ta connaissance.")
age = input("Quel age as-tu ? ")
age_en_mois = 12*age
print("Tu as donc ", age_en_mois, " mois.")
age_en_mois = 12*int(age)
print("Pardon, je voulais dire ", age_en_mois, " mois.")
```

Que se passe-t-il lorsque le programme est lancé ?
Essaie de répondre « 10 » à la question sur l'age.
Essaie de répondre « 10 ans » à la question sur l'age.
D'où viens la première valeur que Python donne pour l'age en mois ?

```
jour = input("Quel jour somme nous ? ")
if jour == "Dimanche":
      print("Vive le repos !")
# On peut mettre autant de elif que l'on veut
elif jour != "Samedi":
      print("On doit être un jour de semaine alors.")
else:
      print("Serions-nous samedi ? On pourrait aller à la médiathèque …")
age = int(input("Quel age as-tu ? "))
if age >= 18:
      print("Tu es majeur, tu peux voter.")
if age < 7:
      print("Tu n'as pas encore l'age de raison.")
```

À quoi sert la ligne commençant par un # ?
Que se passe-t-il si tu réponds « samedi » ou « SAMEDI » ? Pourquoi le résultat est-il différent de « Samedi » ?
Qu'est-ce qui change si tu remplaces le mot clef « elif » par « if ». Pourquoi ?

```
reponse=""
while reponse != "oui":
    reponse = input("Ai-je raison ? ")
    if reponse == "je refuse de répondre":
        break
```

Que peux-tu répondre pour arrêter le programme ? Il y a deux réponses possibles.

Python – Fiche n°3
Listes, tuples et dictionnaires

```
maListe = [ "a", "toto", 3, 4.5 ]
print(maListe[0])
print(maListe[3])
print(maListe[-1])
print(maListe[-4])
print(maListe[1:3])
print(maListe[:3])
print(maListe[1:])
maListe[1]="b"
print(maListe)
maListe.append("titi")
print(maListe)
print(maListe[::2])
print(maListe[1:5:2])
maListe.insert(2, "nouveau")
print(maListe)
maListe.extend(["une", "autre", "liste"])
print(maListe)
print("toto" in maListe)
print(maListe.index("toto"))
maListe.remove("toto")
print(maListe)
```

Que se passe-t-il lorsque l'on utilise maListe avec un index négatif (maListe[-1], maListe[-4]) ?
À quoi correspond maListe[1:3] ? Et maListe[::2] ou maListe[1:5:2] ?
Quelles sont les différences entre les méthodes append, insert et extend ?
Que fait le mot clef « in » dans "« toto » in maListe " ?

```
maListe = [ "a", "toto", 3, 4.5 ]
for element in maListe:
    print(element)
print(range(0,5))
print(list(range(0,5)))
for i in range(0,5):
    print(i)
```

Range n'est pas une méthode mais un générateur. Dans certains cas, il se comporte comme une liste mais ne nécessitera pas autant de mémoire qu'une liste qui contiendrait autant d'élément.

```
monTuple = ( "a", "toto", 3, 4.5 )
print(monTuple)
monTuple[1]="b"
```

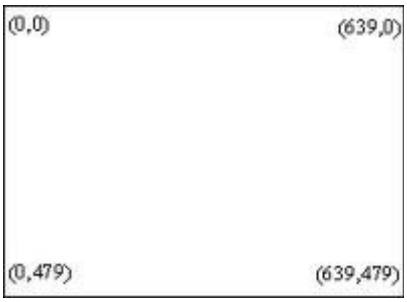
Quelle est la différence entre une liste et un tuple ? Que se passe-t-il lorsque l'on essaie de modifier un tuple ?

Python – Fiche n°4
Pygame – Fenêtre, dessin et évènements

```
import pygame
pygame.init()

size = width, height = 640, 480
speed = [2, 2]
black = 0, 0, 0

screen = pygame.display.set_mode(size)
pygame.display.set_caption("Balle rebondissante")

ball = pygame.image.load("balle.gif")
ballrect = ball.get_rect()

quitter = False
while quitter != True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            quitter = True
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                quitter = True

    ballrect = ballrect.move(speed)
    if ballrect.left < 0 or ballrect.right > width:
        speed[0] = -speed[0]
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]

    screen.fill(black)
    screen.blit(ball, ballrect)
    pygame.display.flip()

pygame.quit()
```

Pour executer le programme, il faut une image s'appelant balle.gif. Ce peut être n'importe quelle image.
Import pygame permet d'importer le module pygame et d'accéder aux fonctions qu'il contient en les prefixant par "pygame.". C'est le cas de "pygame.init()" qui doit être appelé avant d'utiliser pygame.
"pygame.display" est un objet représentant la fenêtre dans laquelle s'affiche ce que l'on va dessiner. "set_mode" et "set_caption" sont deux méthodes agissant sur cet objet. La méthode "set_mode" renvoi un objet représentant la surface dans laquelle on dessine.

Dans cet exemple, la taille de la surface contenue dans la fenêtre est de 640 pixels de large et 480 pixels de haut. On peut accéder aux pixels ou placer des objets en utilisant les coordonnées représentées sur le schema suivant :



Le pixel tout en haut à gauche a les coordonnées (0,0), le pixel tout en haut à gauche a les coordonnées (639, 0), etc … On note souvent x la première coordonnée (position horizontale) et y la seconde coordonnée (position verticale). Contrairement aux coordonnées cartésienne que l'on peut utiliser en mathématiques, y augmente en allant vers le bas.

"pygame.image.load" permet de créer une surface à partir d'une image contenue dans un fichier. Cette surface peut ensuite être dessinée à l'écran à l'endroit où on le désire. La méthode "get_rect" permet de récupérer un tuple contenant les coordonnées et la taille de la surface. Au départ, les valeurs de top et left (haut et gauche) sont à zéro et height et width (hauteur et largeur) contiennent la taille de l'image. Le rectangle possède une méthode "move" qui permet de déplacer le rectangle.

```
rect.move((2,3))
```

revient au même que

```
rect.top = rect.top + 2
rect.left = rect.left + 3
```

"pygame.event.get" permet de récupérer une liste de tous les événements qui se sont produit depuis le dernier appel. Les évènements peuvent être un mouvement ou un clic de souris, une touche pressée ou relachée, etc … L'évènement QUIT correspond à la fermeture par la décoration de la fenêtre (souvent une croix située en haut à droite de la fenêtre) ou par le raccourci clavier correspondant. L'évènement KEYDOWN correspond à une touche pressée. Dans ce cas, l'évènement contient un membre "key" précisant quelle touche a été pressée.

Lorsque la balle touche un bord, on la fait rebondir. C'est à dire :
- si le bord gauche de l'image (ballrect.left) est tout à gauche de l'écran (0) on inverse la vitesse horizontale (speed[0])
- si le bord droit de l'image (ballrect.right) est tout à droite de l'écran (width) on inverse la vitesse horizontale (speed[0])
- si le bord du haut de l'image (ballrect.top) est tout en haut de l'écran (0) on inverse la vitesse verticale (speed[1])
- si le bord du bas de l'image (ballrect.bottom) est tout en bas de l'écran (height) on inverse la vitesse verticale (speed[1])

Jusque là, rien n'a été dessiné. Nous avons juste vérifié si un événement s'était produit et calculé la position de l'image. Pour dessiner, on commence par remplir la surface de noir, afin d'effacer l'image précédente, avec la méthode "screen.fill". La constante black=(0,0,0) définie au début du programme correspond à la couleur noire. Une couleur est un triplet de couleurs, chaque nombre allant de 0 à 255. Le premier nombre correspond au rouge, le second au vert et le troisième au bleu. Avec ces trois couleurs, il est possible de créer tous les autres grâce à la synthèse additive des couleurs (à ne pas confondre à la synthèse soustractive qui est celle utilisée en peinture).

On affiche ensuite l'image à l'endroit désiré en utilisant la méthode "screen.blit" qui copie une surface donnée en premier paramètre à un endroit donné en second paramètre. On appelle ensuite la méthode "pygame.display.flip" qui affiche l'image produite à l'écran. En effet, si le résultat s'affichait au fur et à mesure que l'on trace, il y aurait un effet de clignotement des éléments composant l'image finale.

Comment faire pour qu'au lieu de rebondir toute seule, la balle soit guidé à l'aide du clavier ?

Python – Fiche n°5
Pygame – Encore des évènements, encore du dessin

```
import pygame
import sys

pygame.init()
screen = pygame.display.set_mode((640, 480), 0, 32)

points = []

black = (0, 0, 0)
white = (255, 255, 255)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                sys.exit()
        if event.type == pygame.MOUSEBUTTONDOWN:
            # 1 est le bouton de gauche
            if event.button == 1:
                points.append(event.pos)
        if event.type == pygame.MOUSEMOTION:
            if event.buttons[0]:
                points.append(event.pos)

    screen.fill(white)

    for point in points:
        pygame.draw.circle(screen, black, point, 5)

    pygame.display.update()
```

Trois nouveautés dans cet exemple :

- pygame.MOUSEBUTTONDOWN est le type d'évènement correspondant à un bouton de la souris qui a été pressé. Cet événement à deux membres en plus du type : pos qui est la position du curseur à l'écran au moment où le bouton a été pressé et button qui contient un numéro permettant de savoir quel bouton a été pressé (1 = bouton gauche, 2 = bouton du milieu, 3 = bouton de droite, 4 = molette vers le haut, 5 = molette vers le bas).
- pygame.MOUSEMOTION est le type d'évènement correspondant à un mouvement de la souris. Cet événement à trois membre en plus du type : pos qui est la position du curseur à l'écran au moment où l'évènement s'est produit, rel qui est le déplacement du curseur par rapport au précédent événement du même type (permet d'avoir une idée de la vitesse de la souris) et buttons qui est un tableau de 3 valeurs contenant l'état des boutons de gauche, du milieu et de droite dans cet ordre (0 = bouton relaché, 1 = bouton pressé).
- pygame.draw.circle qui permet de dessiner un cercle (ou plutôt dans notre cas un disque car il est plein). Le premier paramètre est la surface sur laquelle dessiner, dans notre cas l'écran. Le second est la couleur à utiliser. Le troisième est le point servant de centre au cercle. Le quatrième est le rayon du cercle. Un cinquième argument, optionnel, est l'épaisseur du trait. Si il n'est pas présent, le cercle est rempli.

Le sous module pygame.draw contient plusieurs méthodes permettant de dessiner des formes géometriques. La documentation complète ainsi que des exemples d'utilisation peuvent être trouvés ici: http://www.pygame.org/docs/ref/draw.html

Il existe une quinzaine d'évènements. La liste complète et les paramètres associés peuvent être trouvés ici : http://www.pygame.org/docs/ref/event.html

Python – Fiche n°6
Pygame – Monte le son !

```
import pygame, time

pygame.init()

print("Mixer settings : ", pygame.mixer.get_init())
print("Mixer channels : ", pygame.mixer.get_num_channels())

pygame.mixer.music.load('Demo.ogg')
pygame.mixer.music.set_volume(1.0)

pygame.mixer.music.play(1, 0)

while pygame.mixer.music.get_busy():
    print("Playing", pygame.mixer.music.get_pos())
    time.sleep(1)

pygame.mixer.music.stop()

pygame.quit()
```

Cet exemple très simple montre comment jouer un son avec pygame. Cette méthode est utilisée pour jouer une musique en arrière plan car elle permet de controler l'execution du son : jouer en boucle, mettre en pause … L'avantage principal est que la musique n'est pas entièrement chargée en mémoire mais streamée, c'est à dire lue au fur et à mesure. Cela permet d'avoir une musique très longue sans pour autant consommer beaucoup de mémoire.

Il existe une autre manière de jouer des sons qui est plus appropriée pour les effets sonores durant le jeu :

```
import pygame, time

pygame.init()

demo = pygame.mixer.Sound('Demo.ogg')
demo.play()

pygame.quit()
```

Cette méthode est plus simple : on peut créer autant d'objets que nécessaire pour les différents sons du jeu et les jouer très simplement en appelant la méthode play.

Python – Fiche n°7
Fonctions & classes

Un programme devient rapidement grand ! Il est alors difficile de s'y retrouver. Des commentaires peuvent aider à structurer le programme et à le découper en différents morceaux, mais il reste difficile de savoir quel impact aura une modification fait à un endroit. De plus, il est possible que des morceaux du programme se ressemblent beaucoup et fasse la même chose à quelques différence près.

Les fonctions permettent à la fois de réduire la complexité d'un programme et de réutiliser le même code pour effectuer des actions semblables.

```
def soitPoli(nom):
      print('Bonjour',nom,'!')
      reponse = input('Comment vas-tu ? ')
      if reponse.upper() == 'BIEN':
            print('Tant mieux !')
      elif reponse.upper() == 'MAL':
            print('Mince alors !')
      else:
            print('Je vois …')

soitPoli('Colin')
soitPoli('Jean-Pascal')
```

Une fonction peut prendre un ou plusieurs arguments, et peut renvoyer une valeur. Il est possible de renvoyer un type complexe, comme un tuple, ce qui permet donc de renvoyer plusieurs valeurs.

```
def demandeInfos() :
      prenom = input('Quel est votre prenom ? ')
      nom = input('Quel est votre nom ? ')
      age = input('Quel est votre age ? ')
      return(prenom, nom, age)

(p, n, a) = demandeInfos()
print('Bonjour' , p , n , 'vous ne faites pas vos' , a , 'ans !')
```

Un autre moyen de structurer un programme et de réutiliser du code est d'utiliser des classes. Une classe permet la définition d'un nouveau type. Ce type peut contenir des données (des variables) et des méthodes (des fonctions) qui peuvent lui être appliquées.

```
class Point:
    x = 0
    y = 0

    def __init__(self, _x, _y):
        self.x = _x
        self.y = _y

    def move(self, dx, dy):
        self.x += dx
        self.y += dy

    def pair(self):
        return (self.x, self.y)

monPoint = Point(10, 10)
print("Voici mon point: ", monPoint)
print("Voici ses coordonnees: ", monPoint.pair())
monPoint.move(5, -2)
print("Voici mon point apres le move: ", monPoint)
print("Voici ses coordonnees apres le move: ", monPoint.pair())
```