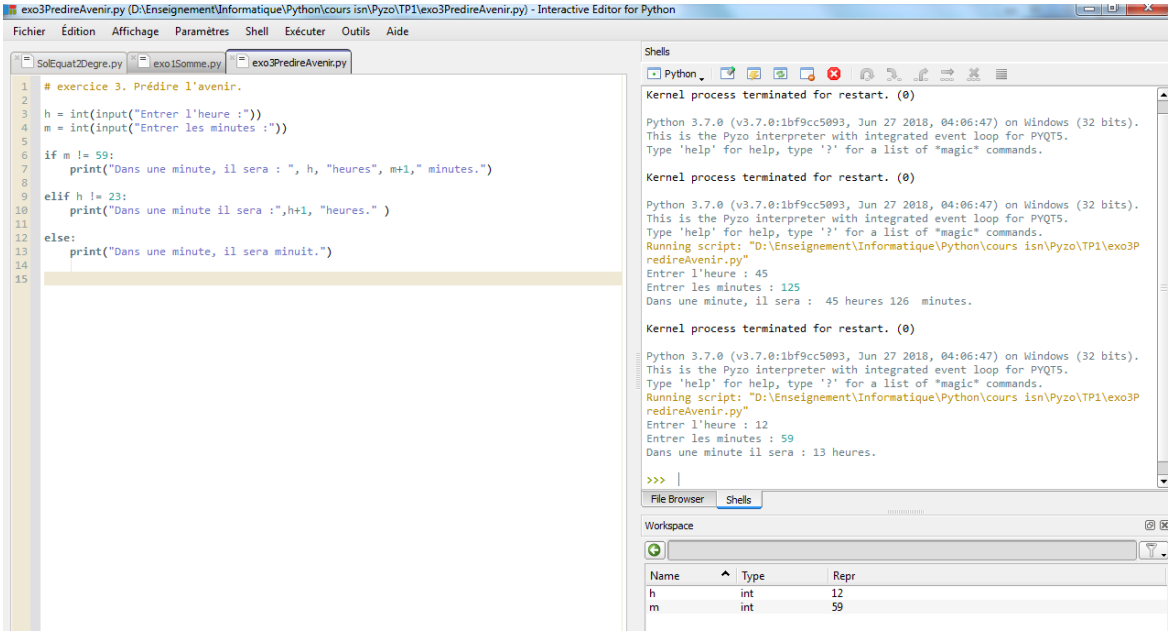
Memo python

(d'après http://python.lycee.free.fr, http://tkinter.fdex.eu/doc/caw.html# et autres)

Vous écrirez vos programmes en langage Python dans le logiciel Pyzo.
Il se présente ainsi : la partie gauche sert à écrire les programmes, la partie droite sert de lien de communication entre le logiciel et vous (test du programme, erreurs signalées…)



On peut ajouter des commentaires dans Python pour expliquer ce qui est fait en écrivant # avant le commentaire.

Interagir dans la console avec le clavier.

input() est une fonction directement accessible (builtin) qui permet d'entrer une information par le clavier. Cette information est saisie en tant que chaîne de caractères.

print(" ce que l'on veut afficher ", à l'écran) : print() est aussi une fonction builtin et permet d'afficher un message ou un résultat à l'écran dans l'interpréteur de Pyzo.
Plusieurs méthodes existent (avec format ou f par exemple).

```
>>> nb = input("Entrer un nombre.")
>>> print(nb)
'25' → ceci est une chaîne de caractères et non un nombre
```

```
>>> a = 9
>>> print("le carré de ", a, " est ", a**2)
le carré de 9 est 81
```

```
>>> prenom = Paul
>>> nom = Dupont
>>> age = 6

>>> print(« je m'appelle{0} {1} et j'ai {2}ans ».format(prenom, nom, age))

>>> print(f "je m'appelle {prenom}{nom} et j'ai {age} ans. ")
```

Des modules que l'on doit importer pour les utiliser.

math	pour les fonctions mathématiques	**random**	pour faire des choix au hasard
numpy	pour faire du calcul scientifique	**matplotlib**	pour le tracé de courbes
tkinter	pour réaliser des dessins	**PIL**	pour le traitement d'images
turtle	pour réaliser des figures animées		

Quelques fonctions utiles

Sans module			
Toutes les fonctions pour les listes et les chaînes de caractère définies plus loin			
min(d, e, f)	Donne le minimum entre d, e et f	**max(d, e, f)**	Donne le maximum entre d, e et f
abs(a)	Calcule la valeur absolue de a	**pow(a,n)**	Donne 'a puissance n' : a^n
Module math			
sqrt(*a*)	√ (racine carré)	**pi**	le nombre pi (avec 15 décimales)
round(*a*)	Arrondi à l'entier	**floor(*a*)**	Partie entière de *a*
degres(a)	Conversion de a en degrés	**radians(a)**	Conversion de a en radians
cos(a)	Cosinus de a (en radians)	**sin(a)**	Sinus de a (en radians)
tan(a)	Tangente de a (en radians)		
Module random			
random()	Renvoie un nombre aléatoire entre 0 et 1	**randint(2,12)**	Renvoie un nombre entier aléatoire entre 2 et 12
choice(liste)	Choisit au hasard une valeur dans une liste donnée		

Appeler un module

import math print(math.cos(math.pi/4))	import math as mt print(mt.cos(mt.pi/4))	from math import pi, cos print(cos(pi/4))

Remarque : il existe aussi la syntaxe from math import * qui importe tout le module math mais ce n'est pas conseillé car il est important de savoir de quel module est extraite telle ou telle fonction donc on n'utilisera pas cette syntaxe.

Les variables

A = 2	Entier : integer	type(A) : int
nombre = 2,45	Décimal : float	type(nombre) : float
" Chameau"	Chaîne de caractères : string	type(" Chameau ") : str
C = [1, 2, 5, 9]	Liste	type(C) : list
True, False	Vrai, faux : Booléen	type(True) : bool

Calculs, opérations

Opérations	Symboles des opérateurs	Exemples
Addition	**+**	2 + 5 donne 7
Soustraction	**-**	8 - 2 donne 6
Multiplication	*	6 * 7 donne 42
Puissance	**	5 ** 3 donne 125
Division	/	7 / 2 donne 3.5
Reste de division entière	%	7 % 3 donne 1
Quotient de division entière Division euclidienne	//	7 // 3 donne 2 donne le quotient euclidien 2

Opérateurs logiques ou expressions logiques

x = = y	Vrai quand x est égal à y	**x != y**	Vrai quand x est différent de y
x > y	Vrai quand x est strictement supérieur à y	**x < y**	Vrai quand x est strictement inférieur à y
x > = y	Vrai quand x est supérieur ou égal à y	**x < = y**	Vrai quand x est inférieur ou égal à y
or	Ou	**and**	Et

Attention : x = y n'est pas une expression mais une affectation : la variable x prend la valeur y.

Listes

Une liste est une collection ordonnée d'objets python. C'est un nouveau type de variable qui peut contenir des nombres, des chaînes de caractères, des variables ou même des listes.

Il y a plusieurs manières de créer une liste

Créer une liste et **lui donner le nom** de Maliste	>>> Maliste = [9, 7, 6, 9] Autre exemple : **>>>** liste1 = [1.56, 7, "tabouret", 3j]

Créer une liste avec la fonction range( ) :
les entiers de 5 (inclus) à 15 (exclu) de 2 en 2.
Afficher avec list().
 	
On peut ne pas préciser l'incrément qui dans ce cas est de 1.
 	
On peut ne pas préciser l'entier de départ qui est automatiquement 0 (et l'incrément reste 1).
	>>> a = **range**(5, 15, 2)
	>>> list(a)
	[5, 7, 9, 11, 13]
	
	>>> B = **range**(3, 9)
	>>> **list**(B)
	[3, 4, 5, 6, 7, 8]
	
	>>> x = **range**(5)
	>>> list(x)
	[0, 1, 2, 3, 4]

Longueur de la liste (nombre d'éléments qui la composent)	>>> **len**(Maliste) 4
Accès à ses éléments On appelle un élément de cette liste d'après son numéro (ou indice) pris dans l'ordre en commençant par 0 pour le premier élément.	>>> Maliste[0] 9
Accès au dernier élément d'une liste	>>> Maliste[– 1] 9
Prendre un élément en comptant à partir de la fin	>>> Liste[-2] 6
Vérifier si un élément est dans la liste	>>> 9 in Maliste True

Prendre une partie de la liste
[1:3] prend les éléments entre la place 1 et la place 2
[1 :] prend les éléments à partir de la place 1
[:2] prend les éléments aux places 0 et 1
	>>> Maliste[1:3]
	[7, 6]
	>>> Maliste[1:]
	[7, 6, 9]
	>>> Maliste[:2]
	[9,7]

Position de la première occurrence d'un élément	>>> Maliste.**index**(7) 1
Nombre d'occurrence d'un élément	>>> Maliste.**count(**9) 2

Ajouter un élément en fin de liste	>>> Maliste = [9, 7, 6, 9]
	>>> Maliste.**append**(2)
	>>> Maliste
	[9, 7, 6, 9, 2]

Ajouter plusieurs éléments en fin de liste	>>> Maliste.**extend**([6, 8])
	>>> Maliste
	[9, 7, 6, 9, 6, 8]

Insérer un élément à une place précise, en position 1, le 5 .	>>> Maliste.**insert**(1,5)
	>>> Maliste
	[9, 5, 7, 6, 9]

Concaténation et multiplication de listes	>>> [1, 2, 3]+[6, 7]
	[1, 2, 3, 6, 7]
	>>> [1,2,3]*2
	[1, 2, 3, 1, 2, 3]

Ranger les éléments d'une liste par ordre croissant	>>> Maliste.**sort**()
	>>> Liste
	[6, 7, 9, 9]

Inverser l'ordre d'une liste	>>> Maliste.**reverse**()
	>>> Maliste
	[9, 9, 7, 6]

Supprimer la première occurrence d'un élément	>>> Maliste.**remove**(9)
	>>> Maliste
	[7, 6, 9]

Retirer un élément en position donnée et le retourner, par défaut, retire le dernier élément	>>> elr = Maliste.**pop**(3)
	>>> Maliste
	[9, 6, 7]
	>>> elr
	9

Chaînes de caractères

Python considère qu'une chaîne de caractères est une liste de caractères, tout ce qui est vrai pour les listes est vrai pour les chaînes de caractères.
Manipulations propres aux chaînes.

Déclarer une chaîne de caractères	>>> Texte = " Bla bli bloublou"

Écrire la chaîne en minuscule	>>> Texte.**lower**()
	>>> Texte
	bla bli bloublou

Écrire la chaîne en majuscule	>>> Texte.**upper**()
	>>> Texte
	BLA BLI BLOUBLOU

Faire une tabulation ou un renvoi à la ligne	**>>>** test2 = u"J'aime **\t** python **\n** et la physique surtout" **>>>** print(test2) J'aime python et la physique surtout
Transforme un nombre en chaîne de caractères	**>>>** liste = **str(**52.12**)** **>>>** liste [4] 1
Transforme une chaîne de caractères en liste : split() ou split(" . ") si la séparation des termes est le point	**>>>**Texte.**split**() **>>>** Texte [‘Bla’ , ‘bli’ , ‘bloublou’.]

Attention à ne pas confondre une chaîne de caractères , "abcd" et des variables a, b, c, d !

>>> abc = "deux mots"
>>> print(abc)
deux mots
	>>> print("abc")
	abc

Tests et boucles

if elif else	Le test commence : si (if) la condition (a<10) est vraie, alors le bloc d’instructions qui suit (ici une seule instruction) est exécuté, sinon si (elif) la condition (a>20) est vraie, alors le bloc d’instructions qui suit est exécuté, sinon (else), c’est-à-dire si les des conditions sont fausses, alors c’est le dernier bloc d’instructions qui est exécuté.	if a<10 : print("Bonjour ") elif a>20 : print(" Bonsoir ") else : print("Bonne nuit")	
for	Lorsque l’on souhaite répéter un nombre **donné** de fois la même instruction ou le même bloc d’instructions, la commande for est la plus appropriée.	for i in range(5) : print(" Blablabla ")	Boucle bornée
while	Le principe de la boucle while, c’est d’exécuter un bloc d’instructions tant que (while) une condition donnée est vraie.	i = 1 while i < 5 : print(i) i = i + 1	Boucle non bornée

Définir une fonction (mathématique ou non) et l’appeler

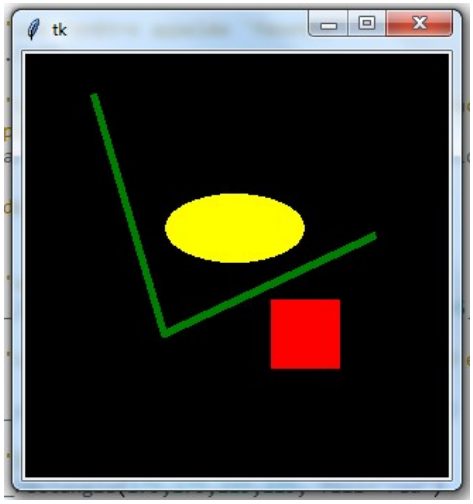
>>> **def** f(x):
return 2 * x + 1
>>> f(0) # on peut maintenant l'utiliser
1
>>> f(5) # et la réutiliser
11
	>>> **def** hi():
	print("Hello from Python!!!")
	>>> hi()
	Hello from Python!!!

>>> **def** hi(name):
print("hello " + name + " from Python!!!")
 	
>>> hi("John")
hello John from Python!!!
	>>> from random import choice
	>>> **def** lettre():
	return choice('abcdefghijklmnopqrstuvwxyz')
	>>> lettre()
	'm'
	>>> lettre()
	'r'

Dessiner dans une fenêtre avec tkinter.

Dans la partie gauche de Pyzo où on écrit le programme.	Le résultat qui s'affiche.

```
# Importation du module tkinter qu'on va nommer 'tk' quand on va
l'utiliser.
import tkinter as tk

# Création d'une fenêtre appelée 'fenetre'.
fenetre = tk.Tk()

# Création d'un canevas appelé 'toile' (avec ses caractéristiques) pour
dessiner et placé dans la fenêtre.
toile = tk.Canvas(fenetre, height=300, bg="black", width=300)

# Affichage du canevas dans la fenêtre.
toile.pack()

# Création d'une ligne.
toile.create_line(50, 30, 100, 200, 250, 130, width=5, fill = "green")

# Création d'un ovale (x1,y1,x2,y2) de diamètres 100 et 50 et de couleur
jaune.
toile.create_oval(100, 100, 200, 150, fill="yellow")

# Création d'1 carré rouge (x1,y1,x2,y2)
toile.create_rectangle(175,175,225,225, fill="red")

#Fermeture fenêtre. Elle reste ouverte tant qu'on ne la ferme pas.
fenetre.mainloop()
```



Sur cette page vous trouverez beaucoup d'informations pour réaliser des formes dans des canvas:
http://tkinter.fdex.eu/doc/caw.html#

Commandes	Effets
Quelques méthodes	**Ce qui s'affiche**
Canvas.**create_line(*x0, y0, x1, y1, ..., xn, yn, option, ...*)**	La ligne est formée de segments qui joignent les points *(x0, y0)*, *(x1, y1)*, … *(xn, yn)*.
Canvas.**create_rectangle(*x0, y0, x1, y1, option, ...*)**	Un rectangle est défini par deux points : *(x0, y0)* pour son coin supérieur gauche et *(x1, y1)* pour son coin inférieur droit.
Canvas.**create_polygon(*x0, y0, x1, y1, ..., option, ...*)**	Un polygone est une ligne fermée. Ainsi, il possède une ligne de contour (formée de segments) et une zone intérieure. Pour le définir, on utilise une série de points [(x0, y0), (x1, y1), … (xn, yn)]. Le premier point et le dernier sont reliés par un segment afin de le fermer.
Canvas.**create_oval(*x0, y0, x1, y1, option, ...*)**	Pour créer l'ellipse (ou le cercle) qui s'inscrit dans le rectangle (ou le carré) *(x0, y0)*, *(x1, y1)* où les premières coordonnées sont celles du coin supérieur gauche et les secondes celles du coin inférieur droit,
Canvas.**create_arc(*x0, y0, x1, y1, option, ...*)**	Le point *(x0, y0)* est le coin supérieur gauche et *(x1, y1)* le coin inférieur droit du rectangle dans lequel s'inscrit l'ellipse. Si le rectangle est un carré, vous obtenez un arc de cercle.
Canvas.**create_text(*x, y, option, ...*)**	Vous pouvez afficher une ou plusieurs lignes de texte sur un canevas, x, y correspondent aux coordonnées du milieu du texte

Quelques options	Traductions
bg = " black "	Couleur de fond, ici en noire
width = 400, height = 400	Taille de la fenêtre, ici 400 pixels sur 400 pixels
width = 5	Épaisseur du trait, ici 5 pixels
fill = " yellow "	Couleur de remplissage de la figure créée, ici en jaune
anchor = " n "	Ancrage de l'objet. Par défaut l'ancrage est " center " par rapport à x,y. Ici il est à " n " c'est-à-dire, nord : le texte apparaît au dessous, le point d'ancrage est donc au nord du texte.
text = " Hello world ! "	Le texte s'affiche. Écrire '\n' pour obtenir un saut de ligne.

Répondre aux événements clavier et souris.

Pages relatives aux événements clavier et souris :
http://tkinter.fdex.eu/doc/event.html
https://zestedesavoir.com/tutoriels/1729/programmation-avec-tkinter/programmation-evenementielle-avec-tkinter/

Événements	Caractéristiques
event.x	Récupère la position en abscisse de la souris
event.y	Récupère la position en ordonne de la souris
'<Motion>'	Mouvements de la souris à l'intérieur de la fenêtre
'<Button-1>'	Un clic sur le bouton gauche de la souris
'<Button-2>'	Un clic sur le bouton central de la souris
'<Button-3>'	Un clic sur le bouton droit de la souris
'<ButtonRelease>'	Le bouton de la souris est relâché
'<KeyPress-c>'	La touche 'c' du clavier est enfoncée
'<Any-KeyPress>'	N'importe quelle touche du clavier est enfoncée
'<KeyRelease>'	La touche clavier est relâchée

Dessiner avec le module Turtle

Pour appeler le module TORTUE	import turtle
effacer le dessin	reset()
aller à l'endroit de coordonnées x ; y	goto(x, y)
avancer d'une distance donnée	forward(distance)
reculer d'une distance donnée	backward(distance)
relever le crayon (pour ne plus dessiner)	up()
abaisser le crayon (pour dessiner)	down()
choisir la couleur du crayon: 'red', 'blue'...	color(couleur)
tourner à gauche d'un angle en degrés	left(angle)
tourner à droite d'un angle en degrés	right(angle)
Toujours terminer le programme par	mainloop()