Mémento python

Variables et opérations

Syntaxe	Rôle
`x=12.5 ou x=y`	Affecte la valeur 12.5 ou la valeur de la variable y à la variable x
`x , y = 1.4 , 3.65`	Affecte simultanément la valeur 1.4 à la variable x et 3.65 à la variable y (équivaut aux instructions x=1.4 et y=3.65)
`a+b`	Somme de a et de b (si a et b sont des chaînes de caractères, le résultat renvoyé est la **concaténation** de a et de b, c'est-à-dire a et b mises « bout à bout »
`a*b`	Produit de a et de b
`a/b`	Quotient de a par b
`a**b`	a puissance b
`a//b`	Quotient de la division euclidienne de a par b
`a%b`	Reste de la division euclidienne de a par b
`int('5')`	Convertit le texte « 5 » en le nombre entier 5
`float('3.14')`	Convertit le texte « 3.14 » en le nombre flottant 3.14
`str(x)`	Convertit la valeur de x en chaîne de caractère (peut être un nombre, une liste, un tableau, etc.)
`round(a)`	Arrondit à l'entier le plus proche, par exemple round(5.61)=6
`int(a)`	Tronque le nombre flottant en entier, par exemple int(5.61)=5
`round(a,b)`	Valeur approchée de a avec une précision de b chiffres après la virgule
`min(a,b)`	Plus petite des deux valeurs a et b
`max(a,b)`	Plus grande des deux valeurs a et b

Demande et affichage (entrées/sorties)

Syntaxe	Rôle
`print("voici un message")`	Affiche le message dans la console
`print("message",var1)`	Affiche « message » et la valeur de var1 séparés par des espaces.
`print("message",end="")`	Affiche « message » et ne passe pas à la ligne.
`texte= input("Réponse? ")`	Affiche un message, attend la saisie de l'utilisateur, affecte la réponse à la variable texte sous le type str.
`n=int(input("Nombre? "))`	Affiche un message, attend la saisie de l'utilisateur, convertit cette saisie en nombre entier et affecte la réponse à la variable n.
`d=float(input("Nombre? "))`	Affiche un message, attend la saisie de l'utilisateur, convertit cette saisie en nombre flottant et affecte la réponse à la variable n.

Fonctions

Syntaxe	Rôle
`def nomfonction(p1,p2) :` `  ....`	Définit une fonction dont le nom est *nomfonction* et qui prend en paramètres p1 et p2.

`def nomfonction(p1,p2) :`
`  ....`
`  return res`
	Définit une fonction qui prend en paramètres p1 et p2.
	Quand elle est appelée, la fonction *renvoie* res

`def nomfonction(p1,p2) :`
`  ....`
`  return res1, res2`
	Quand elle est appelée, la fonction *renvoie* les valeurs *res1*, *res2 :*
	a, b = mafonction(c,d)

Boucles

Syntaxe	Rôle
`for var in range(n):` `    ....`	Répète n fois les instructions. La variable *var* vaut de 0 à n-1.
`for i in range(deb,fin,pas):` `    ....`	Répète les instructions avec la variable i qui vaut de deb inclus à fin exclu avec un pas de pas.
`while condition :` `    ....`	Répète des instructions tant que la condition est vraie (True).
`while True :` `    ....`	Répète indéfiniment les instructions jusqu'à tomber sur l'instruction break
`for element in liste :` `    ....`	Répète les instructions avec la variable element qui vaut tour à tour chaque élément de la liste *liste*.
`for caractere in texte :` `    ....`	Répète les instructions avec la variable caractere qui vaut, tour à tour, chaque caractère du texte *texte*.
`break`	Interrompt la boucle et passe l'instruction suivante.

Instructions conditionnelles

Syntaxe	Rôle
`if condition:` `    ....`	Si la condition est vraie (True) , les instructions sont exécutées

`if condition1:`
`    ....(bloc instructions1)`
`elif condition2:`
`    ....(bloc instructions2)`
`elif condition3:`
`    ....(bloc instructions3)`
`else:`
`    ....(bloc instructions4)`
	Si la condtion1 est vraie, le bloc d'instructions 1 est exécuté.
	Sinon, si la condition 2 est vraie, le bloc d'instructions 2 est exécuté.
	Sinon, si la condition 3 est vraie, le bloc d'instructions 3 est exécuté.
	Sinon (dans tous les autres cas, sans conditions), le bloc d'instructions 4 est exécuté.
	Notez qu'un seul bloc d'instructions peut être exécuté.

`a==b`	Vrai si a et b sont égaux, faux sinon Attention ! « = » est réservé à l'affectation de variables.
`a!=b`	Vrai si a et b sont différents, faux sinon
`a<b ; a>b ; a<=b ; a>=b`	Vrai selon que a est supérieur ou inférieur, strictement ou égal, à b.
`condition1 and condition2`	Vrai si condition1 ET condition2 sont vraies. Faux sinon
`condition1 or condition2`	Vrai si condition1 OU condition2 sont vraies. Faux sinon

Bibliothèques math et rand

Syntaxe	Rôle
`import math`	Importe les fonctions de la bibliothèque math.
`math.pi`	Valeur approchée de π
`math.sqrt(a)`	Racine carrée de a
`mat.sin(a)` `math.cos(a)`	Sinus de a Cosinus de a
`import random`	Importe les fonctions de la bibliothèque random (aléatoire).
`random.random()`	Nombre décimal aléatoire strictement compris entre 0 et 1
`random.randint(a,b)`	Nombre décimal aléatoire compris entre a et b inclus
`random.choice(liste)`	Un élément aléatoire de la liste donnée en paramètre
`random.shuffle(liste)`	Réordonne aléatoirement les éléments d'une liste

Listes

Syntaxe	Rôle
`listef=['iris',lys,'roses' ]`	Crée une liste d'éléments numérotés. 
`len(listef)`	« Length », renvoie la longueur de la liste (nombre d'éléments)

`listef[ 2 ]`
`listef[ -1 ]`
`listef[ 0 : 2 ]`
	Elément de la liste en place i en partant du début
	Elément en place i en partant de la fin
	Eléments de i inclus à j exclu

`listef[ i ] = 'lys'`	Remplace le ième élément par la chaine de caractères « lys »
`listef.append( 'bleuets' )` `listef=listef+['guy','lilas']`	Ajoute des éléments en fin de la liste
`listef.insert(i, 'orchidées' )`	Insert un élément à la place i (décale les éléments suivant)
`listef.pop(i)`	Supprime le ième élément (décale les éléments suivant)
`'iris' in listef` `'iris' not in listef`	Vérifie si un élément appartient à la liste, renvoie True ou False Vérifie si un élément n'appartient à la liste
`listef.sort()`	Ordonne la liste
`Listen=[5,3.5,6,4]`	Construit une liste numérique

`min(listen)`
`max(listen)`
`sum(listen)`
	Renvoie le min, le max, la somme des éléments d'une liste numérique

`import statistics` `statistics.mean(listen)`	Calcule la moyenne des éléments d'une liste numérique.
`listen=list(range(2,12,2))`	Crée la liste de nombre correspondante
`textef='-'.join(['a','b','c'])`	Crée un texte à partir d'une liste (ici « a-b-c »)
`listef='a b c'.split(' ')`	Crée une liste à partir d'un texte (ici [a,b,c])

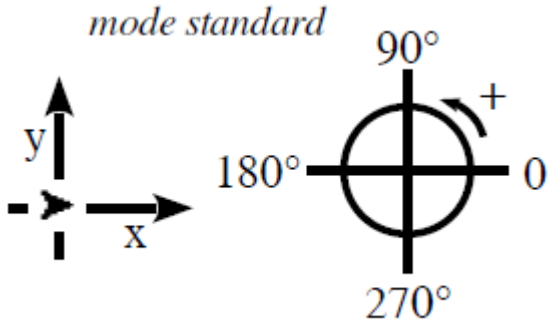
Fichiers

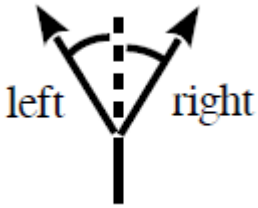
Syntaxe	Rôle
`fichier= open("fichier.txt")`	Crée une variable fichier de type *file*
`for ligne in fichier :` `  ligne.rstrip()`	On peut « boucler » sur un fichier, on le lit alors ligne par ligne (avec des caractères de fin de ligne)

Textes

Syntaxe	Rôle
`f"Je m'appelle {nom}"`	*f-string* : entre accolades, on met les noms des variables qui seront remplacées par leur valeur imprimable.
`txt.lower() / txt.upper()`	Met un texte en minuscule/majuscule

Bibliothèque turtle



Syntaxe	Rôle
`from turtle import *`	Importe les fonctions de la bibliothèque turtle sans avoir besoin de faire précéder les fonctions de turtle. .
`forward(longueur)`	Avance d'une longueur longueur (en pixels)
`backward(longueur)`	Recule d'une longueur longueur (en pixels)
`left(a),right(a)`	Tourne à gauche, tourne à droit, d'un angle *a* en degré, en fonction de la position courante : 
`setheading(direction)`	S'oriente dans une direction (0 = droite, 90 = haut, −90 = bas, 180 = gauche)
`goto(x,y)`	Se déplace jusqu'au point (x, y)
`setx(newx);sety(newy)`	Change la valeur de l'abscisse ; de l'ordonnée
`down() ; up()`	Abaisse/relève le stylo
`width(epaisseur)`	Change l'épaisseur du trait
`color(couleur)`	Change la couleur : "red", "green", "blue", "orange", "purple",. . .
`position()`	Renvoie la position (x, y) de la tortue
`heading()`	Renvoie la direction angle vers laquelle pointe la tortue
`cirle(ray)`	Dessine un cercle de rayon *ray*
`circle(ray,angle)`	Dessine un arc de cercle de rayon *ray* et de degré *angle* (par exemple pour un angle de 180, on aura un demi-cercle)

`fillcolor(couleur)`
`begin_fill()`
`...`
`end_fill()`
	Dessine les instructions … en remplissant les figures dans la couleur demandée.

`exitonclick()`	termine le programme dès que l'on clique (nécessaire sur certains IDE)

