MODULE turtle

<https://docs.python.org/fr/3/library/turtle.html>

```
from turtle import *
```

`showturtle()` Affiche ou cache la tortue
`hideturtle()`

Bouger et dessiner

`forward(x)`, `fd(x)` Avance de x pixels
`backward(x)`, `bk(x)` Recule de x pixels
`right(d)`, `rt(d)` Oriente à droite la tortue de d degrés
`left(d)`, `lt(d)` Oriente à gauche la tortue de d degrés
`goto(x, y)` Déplace la tortue aux coordonnées (x, y)
`circle(r, *a)` Trace un cercle de rayon r. Facultatif : a définit l'angle de l'arc de cercle
`dot(d, *color)` Dessine un point de diamètre d. Facultatif : color définit la couleur
`speed(s)` Définit la vitesse de la tortue
`seth(angle)` Oriente la tortue à angle degré

Réglages du stylo

`pendown()`, `down()` Abaisse le stylo (dessine)
`penup()`, `up()` Lève le stylo (ne dessine pas)
`width(w)` Définit la largeur w du stylo
`color("nom_couleur")` Définit la couleur du stylo

Remplissage

`fillcolor(nom_couleur)` Définit la couleur de remplissage
`begin_fill()` Va remplir la forme dessinée entre `begin_fill()` et `end_fill()`
`end_fill()`

Divers

`numinput(titre, question)` Demande un nombre à l'utilisateur
`onscreenclick(function)` Appelle `function` à chaque clic utilisateur

TYPES, VARIABLES et CONVERSIONS

Types **int** - entier **float** - décimal **str** – texte **bool** - booléen

Déclaration de variable

`T = "Salut"` Assigne à la variable T la valeur texte "Salut"
`E = 22` Assigne à la variable E la valeur entière 22

Conversion de type

`str(N)` transforme un entier ou un décimal N en un texte
`int(T)` transforme un texte T en un entier
`float(T)` transforme un texte T en un décimal

ENTRÉES et SORTIES : demander une valeur à l'utilisateur et afficher à l'écran

Entrée au clavier

`Un_texte = input("question")` Pose question à l'utilisateur et stocke la réponse dans une variable de type **str**
`Un_Entier = int(input("question"))` Pose question à l'utilisateur et stocke la réponse dans une variable de type **int**

Sortie à l'écran

`print("Texte", variable)` Imprime à l'écran `Texte` suivi du contenu de `variable`

TESTS et CONDITIONS

Test simple

```
if condition1:
    bloc1
elif condition2: #optionnel
    bloc2
else: #optionnel
    bloc3
```

Test multiple

```
if condition1 and condition2:
    bloc1
```

Condition

Question comparative à laquelle on ne peut répondre que par OUI ou NON => **True / False**

Opérateurs de comparaison

< plus petit que == égal à
<= plus petit ou égal à != différent de
> plus grand que
>= plus grand ou égal à

```
if condition1 or condition2:
    bloc1
```

OPÉRATEURS et FONCTIONS MATHÉMATIQUES

<code>x ** y</code>	renvoie x puissance y	<code>from math import *</code>	
<code>x // y</code>	renvoie la partie entière de la division de x par y		
<code>x % y</code>	renvoie le reste de la division de x par y	<code>pi</code>	retourne la valeur de PI
<code>round(x)</code>	arrondit x vers l'entier le plus proche	<code>sqrt(x)</code>	retourne la racine carrée de x
<code>max(x,y)</code>	retourne la plus grande valeur entre x et y		
<code>min(x,y)</code>	retourne la plus petite valeur entre x et y	<code>cos(x), sin(x), tan(x)</code>	
<code>abs(x)</code>	retourne la valeur absolue de x	<code>acos(x), asin(x), atan(x)</code>	

BOUCLES

Boucle WHILE (tant que)

`while condition:`
 bloc

Le bloc d'instruction est exécuté **tant que la condition est vraie**.
ATTENTION : la variable utilisée dans condition doit être modifiée dans bloc sinon la boucle sera infinie

Boucle FOR

`for variable in ensemble_de_valeurs:`
 bloc

Variable va prendre toutes les valeurs de *ensemble_de_valeurs* et le *bloc* sera exécuté autant de fois qu'il existe de valeurs différentes

Boucle FOR avec des nombres

`for variable in range(nombre):` *Variable* varie de 0 à *nombre-1*
`for variable in range(début, fin):` *Variable* varie de *début* à *fin-1*
`for variable in range(début, fin, pas):` *Variable* varie de *début* à *fin-1* par saut de *pas*

FONCTIONS

Définir une fonction / séquence d'instructions

`def ma_fonction(paramètre1, paramètre2,...):`
 bloc
 return *variable*

Les paramètres sont facultatifs.
Le retour est facultatif

Appel d'une fonction

Sans valeur de retour: `ma_fonction(p1, p2, ...)`
Avec une valeur de retour: `variable = ma_fonction(p1, p2, ...)`

LISTES / TABLEAUX

Déclaration

`Liste = []` Crée une liste vide nommée **Liste**
`Liste = ["A","B","C","D"]` Crée une liste à 4 éléments de type **str** nommée **Liste**
`Liste = [2,4,6,8,10]` Crée une liste à 5 éléments de type **int** nommée **Liste**

Accéder aux éléments de la liste

`Liste[index]` Renvoie l'élément situé à la position *index* (qui commence à 0) de **Liste**
`Liste[-1]` Renvoie le dernier élément de **Liste**
`Liste[index1 : index2]` Renvoie les éléments de la position *index1* à *index2* de **Liste**

Manipuler une liste

`len(Liste)` Renvoie le nombre d'élément (la longueur) de **Liste**
`Liste.append(element)` Ajoute **element** à la fin de **Liste**
`Liste.remove(element)` Supprime la 1^{ère} occurrence de **element** de **Liste**
`Liste[pos] = element` Met **element** à la position *pos* dans **Liste** et remplace l'élément existant
`element in Liste` Cherche si **element** est présent dans **Liste** et renvoie True/False

Parcourir une liste

`for elem in Liste:` **elem** prend tour à tour la valeur de chaque élément de la liste

MODULE random

`from random import *`

`choice(Liste)` Choisi un élément aléatoirement dans **Liste**
`sample(Liste, n)` Choisi *n* éléments aléatoirement dans **Liste**
`shuffle(Liste)` Mélange les éléments de **Liste**
`randrange(borne1, borne2)` Choisi un entier aléatoirement compris entre *borne1* et *borne2*