

MEMENTO PYTHON

FAKHREDDINE GHOMMID – I.R.E.M. DE DIJON

AFFECTATION (OU ASSIGNATION) :

```
>>> n = 7                # donner à n la valeur 7
>>> msg = "Bonjour"    # affecter la valeur "Bonjour" à msg
```

Sous Python, il n'est pas nécessaire de définir le type des variables avant de pouvoir les utiliser. Il suffit d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée avec le type qui correspond au mieux à la valeur fournie.

OPERATEURS USUELS SUR LES NOMBRES :

+, -, /, *, **(puissance), % (a%b donne le reste de la division euclidienne de a par b)

En ajoutant au début du script : `from math import *`, les fonctions mathématiques suivantes sont chargées :

`acos(x)`, `asin(x)`, `atan(x)`, `cos(x)`, `exp(x)`, `fabs(x)` (valeur absolue), `floor(x)` (partie entière), `log(x)` (logarithme népérien), `log10(x)` (logarithme décimal), `sin(x)`, `sqrt(x)` (racine carrée), `tan(x)`

Le module définit aussi deux constantes mathématiques: `pi` et `e`.

Le module `random` (`from random import*`) propose toute une série de fonctions permettant de générer des nombres aléatoires qui suivent différentes distributions mathématiques. En particulier, pour la répartition uniforme, `random()` retourne un réel de l'intervalle $[0,1[$ et `randrange(n,p)` un entier compris entre n et p-1.

SELECTION OU EXECUTION CONDITIONNELLE :

| | | |
|---|--|--|
| <pre>>>> if (a > 10): ... print "trop grand" ... </pre> | <pre>>>> if (a > 10): ... print "trop grand" ... else: ... print "convenable" ... </pre> | <pre>>>> if a > 0 : ... print "a est positif" ... elif a < 0 : ... print "a est négatif" ... else: ... print "a est nul" ... </pre> |
|---|--|--|

La condition évaluée après l'instruction if peut contenir les *opérateurs de comparaison* suivants :

| | |
|------------------------|-------------------------------------|
| <code>x == y</code> | # x est égal à y |
| <code>x != y</code> | # x est différent de y |
| <code>x > y</code> | # x est plus grand que y |
| <code>x < y</code> | # x est plus petit que y |
| <code>x >= y</code> | # x est plus grand que, ou égal à y |
| <code>x <= y</code> | # x est plus petit que, ou égal à y |

DEFINITION D'UNE FONCTION MATHÉMATIQUE :

```
>>> def f(x) :
...     return x**2+x
```

ou par exemple :

```
>>> def morc(x):
...     if x>2:
...         return 2*x-1
...     else:
...         return x**2
```

INSTRUCTIONS REPETITIVES :

L'instruction while (tant que)

```
>>> a = 0
>>> while (a < 7):
...     a = a + 1
...     print a
```

L'instruction for

```
>>> for i in range(6):
...     print i,i**2
ou
```

```
>>> jours=["lundi","mardi","mercredi","jeudi","vendredi","samedi","dimanche"]
>>> for i in jours:
...     print i
```

GESTION DES ENTREES ET SORTIES :

```
print 'Veuillez entrer un nombre positif quelconque : ',
nn = input()
print 'Le carré de', nn, 'vaut', nn**2
```

ou encore :

```
prenom = input('Entrez votre prénom (entre guillemets) : ')
print 'Bonjour,', prenom
```

Ou bien pour saisir une liste :

```
print 'Veuillez entrer trois nombres séparés par des virgules : '
nn = list(input())
```

POUR LES GRAPHIQUES :

Courbe représentative d'une fonction :

```
from pylab import *
x = arange(-2.0, 2.0, 0.01)
y = sin(2*pi*x)
plot(x, y, linewidth=1.0)
xlabel('abscisses')
ylabel('ordonnées')
title('Petite représentation graphique')
grid(True)
show()
```

LE MODULE « TURTLE »

```
>>> from turtle import *
>>> forward(120)
>>> left(90)
>>> color('red')
>>> forward(80)
```

Exemple :

```
>>> reset()
>>> a = 0
>>> while a <12:
    a = a +1
    forward(150)
    left(150)
```

Les principales fonctions dans le module *turtle* sont les suivantes :

| | |
|---------------------------------|--|
| <code>reset()</code> | On efface tout et on recommence |
| <code>goto(x, y)</code> | Aller à l'endroit de coordonnées x, y |
| <code>forward(distance)</code> | Avancer d'une distance donnée |
| <code>backward(distance)</code> | Reculer |
| <code>up()</code> | Relever le crayon (pour pouvoir avancer sans dessiner) |
| <code>down()</code> | Abaisser le crayon(pour recommencer à dessiner) |
| <code>color(couleur)</code> | <couleur> peut être une chaîne prédéfinie ('red', 'blue', 'green', etc.) |
| <code>left(angle)</code> | Tourner à gauche d'un angle donné (exprimé en degrés) |
| <code>right(angle)</code> | Tourner à droite |
| <code>width(épaisseur)</code> | Choisir l'épaisseur du tracé |
| <code>fill(1)</code> | Remplir un contour fermé à l'aide de la couleur sélectionnée |
| <code>write(texte)</code> | <texte> doit être une chaîne de caractères délimitée avec des " ou des ' |

Références :

1. Apprendre à programmer avec Python par Gérard Swinnen
2. <http://www.pythonxy.com/>
3. <http://matplotlib.sourceforge.net/>

QUELQUES PROGRAMMES ECRITS EN LANGAGE PYTHON

1. Un jeu

Un nombre entier compris entre 1 et 100 est tiré « au hasard ». Le joueur doit le retrouver sachant que l'ordinateur lui répond que le nombre que ce dernier propose est trop grand, trop petit ou le bon.

```
from random import *
secret=randrange(1,101)
rep=0
print("J'ai choisi un nombre compris entre 1 et 100. Quel est-il ?")
while rep!=secret:
    rep=input("Votre proposition ?")
    if rep>secret:
        print("Trop grand")
    elif rep<secret:
        print("Trop petit")
    else :
        print("Gagné")
```

Comment modifier ce programme pour qu'il affiche le nombre de coups nécessaires à l'obtention du nombre cherché ? Et le pourcentage d'essais trop grands ?

2. La boîte à outils des calculs de coordonnées.

```
from math import *
xa=input("abscisse du point A")
ya=input("ordonnée du pont A")
xb=input("abscisse du point B")
yb=input("ordonnée du point B")
xi=(xa+xb)/2
yi=(ya+yb)/2
print "Le milieu du segment [AB] a pour coordonnées : (",xi,",",yi,")"
print "Le vecteur AB a pour coordonnées : (",xb-xa,",",yb-ya,")"
print "La distance AB vaut : ",sqrt((xb-xa)**2+(yb-ya)**2)
if xb!=xa :
    m=(yb-ya)/(xb-xa)
    p=ya-m*xa
    print "L'équation réduite de la droite (AB) est : y=",m,"x+",p
else:
    print "L'équation réduite de la droite (AB) est : x=",xa
```

Réaliser un programme de ce type afin de savoir si deux droites (AB) et (CD) sont parallèles, confondues ou sécantes.

3. Autour des fonctions

Voir documents d'accompagnement

4. Python simule-t-il correctement un lancer de dé ?

```
from random import*
# Python simule-t-il correctement un lancer de dé ?
n=input("Indiquer le nombre de lancers :")
liste=[0,0,0,0,0,0,0]
i=0
while i<n:
    i=i+1
    r=randrange(6)
    liste[r]=liste[r]+1
print liste
```

5. La méthode de Monte-Carlo pour une estimation de Pi.

```
from random import*
n=input("Indiquer le nombre de tirages aléatoires")
s=0.0
i=0
while i<n:
    i=i+1
    x=random()
    y=random()
    if x**2+y**2<=1:
        s=s+1
print "La proportion des impacts dans le disque est de :",4*s/n
```