Python v1.0 11/04/15

Python est un langage puissant et flexible que vous pouvez utiliser pour du développement web, pour réaliser des interfaces homme-machine (IHM), créer des jeux et bien plus encore. Python est :

• **Haut-niveau**, c'est-à-dire qu'écrire et lire en Python est vraiment simple, cela ressemble beaucoup à de l'anglais.

• **Interprété**, c'est-à-dire que vous n'avez pas besoin de compilateur pour écrire et exécuter du Python !

• **orienté objet**, c'est-à-dire que le langage Python permet à ses utilisateurs de manipuler des structures de données appelées **objets** afin de modéliser et exécuter des programmes.

• **Fun to use**. Python a été nommé d'après *Monty Python's Flying Circus*, les exemples de code et les tutoriels utilisent souvent des références à la série et utilisent l'humour pour rendre l'apprentissage du langage plus intéressant.

Déclaration de variable:
```
mon_int = 7
mon_float = 1.23
mon_bool = True
mon_string = 'looool'
```

Afficher une variable :
```
print mon_int
print mon_string
```

Syntaxe d'un définition :
```
def spam():
  oeufs = 12
  return oeufs
print spam()
```

Commentaire:
```
#Je suis un commentaire
ou
"""Je suis un commentaire
sur plusieurs lignes"""
```

Les opérateurs arithmétiques :
+ / - *
La puissance (**)
Le modulo (%).

Afficher la valeur de total à 2 chiffres après la virgule :
```
print("%.2f" % total)
```

Afficher un caractère d'un string :
```
cinquieme_lettre = "MONTY"[4]
print cinquieme_lettre
```

Affiche la longueur d'une variable:
```
print len(perroquet)
ou
print len('looool')
```

Modification de string :
```
print perroquet.lower() //Met en minuscule
print perroquet.upper () //Met en majuscule
print str(pi)  //Transforme une variable en string
```

Concaténation:
```
print "La valeur de pi est environ " + str(3.14)
```

Formatter des Strings avec %:
```
g = "Golf"
h = "Hotel"
print "%s, %s" % (g, h)
```

Affiche le message à l'utilisateur et stocke la réponse dans la variable :
```
nom = raw_input("Quel est votre nom ?")
var = int(raw_input("texte")) //La même en le parsant en int
```

DateTime actuel :
```
from datetime import datetime
now = datetime.now()
print now
et
print now.year //Afficher l'annee
print now.month //Afficher le mois
print now.day //Afficher le jour
et
print '%s/%s/%s' %
print now.hour
print now.minute
print now.second
```

Opérateurs logiques :
True and True is True
True and False is False
False and True is False
False and False is False
True or True is True
True or False is True
False or True is True
False or False is False
Not True is False
Not False is True

Etape d'évalutation :
not est évalué en premier.
and est évalué en deuxième.
or est évalué en dernier.

Syntaxe if/elseif/else :
```
if 8 < 9:
   print "Je suis affiche !"
elif 8 > 9:
   print "Je ne suis pas affiche !."
else:
   print "Je ne suis pas affiche non plus !"
```

Syntaxe fonctions :
```
def premiere_saisie():
   if 1 < 2:
      return "Succes #1"
premiere_saisie() //Appelle la fonction
```

Présence d'un ou plusieurs caractères:
ma_variable . isalpha() *//Vérifie si chiffre*
if premiere in ["a","e","i","o","u","y"]:
//vérifie si la lettre est une voyelle

Import générique :
import math *//Importe le module*
print math.sqrt(25) *//Utilisation du module*

Import de fonction :
from module import fonction
EX: from math import sqrt
from math import * *//Importation de toutes les fonctions*

Affecter le contenu d'un module dans une liste :
tout = dir(math) *//Affecte le contenu de math à une liste*

Recherche de valeur :
min(1,156,123,48,7) *//Affiche la plus petite val*
max(461,41,54,47) *// Affiche la plus grande val*
abs(-15) *// Affiche la valeur absolue*

Affiche le type d'un élément :
print type(42) *//Affiche : <type 'int'>*

Les listes :
```
animaux_zoo = ["pangolin", "casoar", "paresseux",
"emeux"];
```
//Définition de la liste et de ses éléments
```
ma_liste = range(51)
```
//Créer une liste de 0 à 50 inclus
```
animaux_zoo[0]
```
//Accéder à un élément de la liste
```
animaux_zoo[2] = "emeu"
```
//Accéder à un élément de la liste
```
animaux_zoo[2:3]
```
//Accéder à un groupe d'éléments [à partir de la cellule n°, jusqu'à la cellule n° (non inclut)]
```
nomdelalist.append("element")
```
//Ajouterà un élément à la liste
```
ma_liste[:2]
```
// Sélectionnez les deux premiers éléments
```
ma_liste[3:]
```
// Sélectionnez le quatrième élément jusqu'au dernier
```
animaux.index("canard")
```
// Connaitre l'index de canard dans la liste animaux
```
animaux.insert(1, "chien")
```
// Insérer chien à l'index 1 dans la liste animaux
```
x.append(element)
```
//Ajouter un élément à la liste
n.pop*(index) supprimera l'élément à l'index de la liste et vous le retournera*
```
n.remove(1)
# Supprime 1 de la liste,
# et non pas l'élément à l'index 1
```
del(n[1]) *est comme .pop dans la mesure où il va supprimer l'élément à l'index donné, mais ne le retournera pas*
La fonction range() a trois versions différentes :
range(stop)
range(debut, stop)
range(debut, stop, pas)

Les listes à clés (Dictionnaires) :
```
d = {
   "Name": "Guido",
   "Age": 56,
   "BDFL": True
}
```
//Définition de la liste et de ses éléments
```
for key in mon_dict:
   print key, mon_dict[key]
```
//key = clé / mon_dict[key] = valeur
```
print residents['Puffin']
```
// Affiche l'élément correspondant à la clé
```
menu['Poulet'] = 14.50
```
//Ajout une nouvelle paire valur/clé
```
del dict_nom[nom_de_la_cle]
```

EX: del zoo_animaux['Paresseux']
// Effacer un élément
dict_nom[key] = nouvelle_valeur
EX: zoo_animaux['Pingouin'] = 'Pas Arctique'
// Une nouvelle valeur peut être associée à une clé
nomlist.remove("nomelement")
// Supprimer un élément
for element in webster:
    print webster[element]
//Lire tous les éléments
print d.items()
//Affiche [('BDFL', True), ('Age', 56), ('Name', 'Guido')]
La fonction keys() *retourne un tableau contenant les clés du dictionnaire*
La fonction values() *retourne un tableau contenant les valeurs du dictionnaire*.

Liste multiples :
for a, b in zip(liste_a, liste_b):
// ZIP permet de comparer autant de listes que voulu

Liste en compréhension :
pairs_a_50 = [i for i in range(51) if i % 2 == 0]
// Créer une liste [0, 2, 4 … 48, 50]
*pairs_carre = [x**2 for x in range(1,12) if x % 2 == 0]*
// Créer une liste [4, 16, 36, 64, 100]
nom_liste[debut:fin:pas]
// Syntaxe Slicing (tranche)
backwards = ma_liste[::-1]
// Ecrire une liste à l'envers

Syntaxe boucle FOR :
for variable in nom_liste:
    // Tapez votre code ici !
OU COMME CELA
for element in [1, 3, 21]:
    print element

list_of_lists = [[1,2,3], [4,5,6]]
for lst in list_of_lists:
    for item in lst:
        print item
    // Parcourir une liste dans une liste
print c,
// La virgule permet d'écrire les prochains char sur la même ligne

Syntaxe WHILE :
while compte < 5:
    print "Je suis un while et compte vaut",
compte
    compte += 1

Vérifier si un nombre x est entier :
if x - round(x,0) == 0:

Calcul factoriel:
while int(x) != int(0):
        x -= 1
        test *= x

Calcul nombre premier :
for n in range(2,x,1):
        if x%n==0:

Programmation fonctionnelle / fonctions anonymes :
ma_liste = range(16)
print filter(lambda x: x % 3 == 0, ma_liste)
//Affiche [0, 3, 6, 9, 12, 15]
// lambda x: x % 3 == 0
// est équivalent à :
// def par_trois(x):
*   return x % 3 == 0*
fonctions lambda :
SYNTAXE: lambda variable: variable expression
EX:
langages = ["HTML", "JavaScript", "Python", "Ruby"]
print filter(lambda x: x=="Python", langages)
//['Python']

Opérations bit à bit :
print 5 >> 4 # Décalage à droite
print 5 << 1 # Décalage à gauche
print 8 & 5   # ET binaire
print 9 | 4   # OU binaire
print 12 ^ 42 # OU EXCLUSIF binaire
print ~88   # NON binaire

Base 2 (Binaire) :
print 0b1,   #1
print 0b10,   #2
print 0b11,   #3
print 0b100, #4
print 0b101, #5
print 0b110, #6
print 0b111   #7
print "******"
print 0b1 + 0b11 #4
print 0b11 * 0b11 #9

print bin(5)
//Converti en binaire 0b101
print int("0b11001001", 2)
//Converti un binaire en decimal
rint bin(0b1110 & 0b101)
//Opération (ET)
print bin(0b1110 | 0b101)
//Opération (OU)

```
print bin(0b1110 ^ 0b101)
//Opération (XOR) ou OU EXCLUSIF
print ~1
//Opération (NOT) fait +1 et rend le nombre négatif
def verifier_bit4(entree):
    mask = 0b1000
    desire = entree & mask
    if desire > 0:
        return "on"
    else :
        return "off"
//Vérification si un bit est actif
```

Syntaxe CLASSE :

```
class Fruit(object):
    """Une classe qui fait différents fruits
savoureux."""
    def __init__(self, nom, couleur, saveur, toxique):
        self.nom = nom
        self.couleur = couleur
        self.saveur = saveur
        self.toxique = toxique
```

Syntaxe Classe Dérivée (ou SOUS CLASSE) :

```
class ClasseDerivee(ClasseDeBase):
```

Syntaxe METHODE dans une classe :

```
def __init__(self, cote):
    self.cote = cote
```

Syntaxe METHODE representation:

```
def __repr__(self):
        return "(%d, %d, %d)" % (self.x, self.y, self.z)
```

Lecture/Ecriture de fichier :

```
f = open("output.txt", mode)
//FONCTION OPEN()
mode écriture seule ("w")
mode lecture seule ("r")
mode lecture et écriture ("r+")
mode append ("a", qui ajoute n'importe quelle
nouvelle donnée que vous écrivez dans un fichier, à
la fin de ce fichier).
//Les différents modes
mon_fichier.write("Donnée à écrire")
mon_fichier.close()
//Ecriture puis fermeture du fichier (NE PAS
OUBLIER!)
print mon_fichier.read()
//Lire à partir du fichier
with open("fichier", "mode") as variable:
//Lire ou écrire dans le fichier puis REFERMER le
FICHIER

if mon_fichier.closed == False:
    mon_fichier.close()
//Fermer le fichier si il ne l'ai pas
```