**PARTIE 1 : Les premiers mots clé**

GLOSSAIRE

page 1/2

print() : affiche la variable demandée ou le message proposé entre ‘ ‘ .

int() : Reconnait comme un entier ou donne la troncature .

float() : Reconnait comme un réel ou convertit en réel si c’est faisable

str() : Reconnait comme une chaine de caractères ou convertit en chaîne de caractères si c’est faisable

input() : attend la saisie d’une donnée par l’utilisateur, instruction à n’utiliser qu’en phase d’apprentissage

Cette fonction peut être combinée afin de demander un entier : int(input()) ou un flottant ; float(input())

**PARTIE 2 : Boucles et instructions conditionnelles**

|  |
| --- |
| CONDITIONS |
| égalité | == |  |
| différent | != |  |
| comparaison | < ;<= ;> ;>= |  |
| et | and |  |
| ou | or |  |
| INSTRUCTIONS CONDITIONNELLES |
| si …….alors …..sinon | if condition : instruction |  |
| if condition : instruction 1else : instruction 2 |  |
| if condition 1 : instruction 1elif condition 2 : instruction 2elif condition 3 : instruction 3else : instruction 4 |  |
| BOUCLE NON BORNÉE |
| Tant que | while condition : instruction |  |
| BOUCLE BORNÉE |
| pour ….allant de … à … | for i in range(n) : instructionrange (n) : répète n fois, n prenant ses valeurs entre 0 et n - 1 |  |
| for i in range(a,b) : instructionrange(a,b) où a et b sont des entiers : répète de i = a à i = b - 1 |  |

Les boucles ne se finissent pas avec un end, c’est l’indentation qui permet de délimiter les boucles.

Pour une version « papier » des scripts on utilisera le symbole pour signifier une indentation.

Délimiteurs particuliers : les messages de textes sont placés entre «   »

 les commentaires sont placés après un #

**PARTIE 3 : Les Fonctions**

GLOSSAIRE

page 2/2

Syntaxe :

def {nom de la fonction} ({liste de paramètres})

 # cette liste peut être vide

 instructions

return {résultat(s)} # peut renvoyer plusieurs données

EXEMPLE 1 : Fonction à 2 paramètres : Calculer le volume d’un cylindre connaissant les dimensions



Calculer le volume du cylindre de hauteur 5 et de rayon 2,3

Affichage :

EXEMPLE 2 : Fonction avec un paramètre : Somme des n premiers entiers



Tester pour n = 50.

Tester pour les entiers de 50 à 100

Affichage :

Une autre technique pour définir les fonctions : **lambda** : cf exercices 29 + document d’accompagnement lycée.

**POINT INFO : Les bibliothèques :**

Pour l’instant nous utiliserons principalement les bibliothèques **math** et **random**.

Pour avoir la liste des fonctions disponibles dans une bibliothèque on peut utiliser : **import math** puis **dir(math**) :



Pour avoir une aide spécifique sur une fonctionnalité, dès que la bibliothèque est importée, on peut utiliser : help().

La bibliothèque **random** :





On prendra l’habitude de n’importer que les fonctionnalités nécessaires au script en construction : comme dans l’exemple 1 où seul le nombre $π$ a été importé :

Lorsque le script fait appel à plusieurs bibliothèques ou lorsqu’il y a un risque de doublon entre une fonction définie par l’utilisateur et une fonction prédéfinie on utilise un alias, par exemple la fonction **randint** du module **random** peut être définie en faisant : import **random** as **rd** puis **rd.randint**.

**Travail dans la console :**

Exemples pour s’approprier les différentes instructions : opérations de base et types de variables.

|  |  |  |
| --- | --- | --- |
| À saisir | Résultats | Explications |
| >>> 2 + 3 |  |  |
| >>> ‘’a’’+’’b’’ |  |  |
| >>> 2 \* 3 |  |  |
| >>> 2.1 \* 5 |  |  |
| >>> ‘’a’’\*3 |  |  |
| >>> 7 / 3 |  |  |
| >>> 7 // 3 |  |  |
| >>> 15 // 4 |  |
| >>> 2 \*\* 3 |  |  |
| >>> 2 \* 3 + 5 |  |  |
| >>> 52 % 15 |  |  |
| >>> 23 % 7 |  |
| >>> a = 2>>> a + 1>>> "a+1" |  |  |
| >>> a,b = 2,3>>> a + b |  |  |
| >>> 1 > 2 |  |  |
| >>> from math import pi>>> pi < 4 |  |  |
| >>> round(2/3,5) |  |  |

def somme\_carrés(n):

 S = 0

 for i in range(n):

 S = S + i\*\*2

 return S

C’est dans la console que l’on peut afficher les

résultats de fonctions définies dans l’éditeur :

def fonction(x) :

 return x\*\*2+5x-9

|  |  |  |
| --- | --- | --- |
| À saisir | Résultats | Explications |
| >>> fonction(5) |  |  |
| >>> fonction(0) |  |  |
| >>> somme\_carrés(12) |  |  |
| >>> somme\_carrés(25) – somme\_carrés(12) |  |  |

**Typage :**

Python est un langage à typage dynamique fort, sans précision de l’utilisateur, Python type tout seul.

|  |  |  |
| --- | --- | --- |
| À saisir | Résultats | Explications |
| >>> type(‘‘abc’’) |  |  |
| >>> type(12) |  |  |
| >>> type(2.0) |  |  |
| >>> type(2\*\*2+3\*\*2==5\*\*2) |  |  |

En revanche il est possible de faire du **transtypage**, à savoir imposer un type à Python, comme par exemple traiter un entier comme une chaîne de caractère.

|  |  |  |
| --- | --- | --- |
| À saisir | Résultats | Explications |
| >>> a = str(123) |  |  |
| >>> b = str(45) |  |  |
| >>> a + b |  |  |
| >>> int(a) + int(b) |  |  |
| >>> int(a+b) |  |  |