UNIVERSITE JOSEPH KI-ZERBO/IFOAD

INTRODUCTOIN A PYTHON

KYELEM YACOUBA

Doctorant en Informatique

Université Joseph Ki-Zerbo

Laboratoire de Mathématiques et d'Informatique (LAMI)

Contact : kyelemy@yahoo.fr/71025937

Table des matières

CHAPITRE 1 : LES CONCEPTS DE BASE DU LANGAGE PYTHON

Installation des outils pour la programmation en python

- Installer python https://www.python.org/downloads/windows/
- Installer Visual Studio Code : https://visual-studio-code.fr.uptodown.com/windows/telecharger
  Ou
- Installer un IDE python (Anaconda) https://www.anaconda.com/download/

Présentation du langage Python

Historique

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

Guido van Rossum

La dernière version de Python est la version 3.11 publiée en octobre 2022.

La Python Software Foundation est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs.

Les caractéristiques de Python

Langage Open Source

Licence Open Source CNRI, compatible GPL, mais sans la restriction copyle□. Donc Python est libre et gratuit même pour les usages commerciaux.

Travail interactif

- ✓ Nombreux interpréteurs interactifs disponibles (notamment IPython)
- ✓ Importantes documentations en ligne
- ✓ Développement rapide et incrémentiel
- ✓ Tests et débogage outillés
- ✓ Analyse interactive de données

Langage interprété rapide

- ✓ Interprétation du bytecode compilé

- ✓ De nombreux modules sont disponibles à partir de bibliothèques optimisées (souvent écrites en C ou C++)

Simplicité du langage :

- ✓ Syntaxe claire et cohérente — Indentation significative

Gestion automatique de la mémoire (garbage collector)

- ✓ Typage dynamique fort : pas de déclaration

Orienté objet

- ✓ Modèle objet puissant mais pas obligatoire
- ✓ Structuration multifichier aisée des applications : facilite les modifications et les extensions
- ✓ Les classes, les fonctions et les méthodes sont des objets dits de première classe. Ces objets sont traités comme tous les autres (on peut les affecter, les passer en paramètre)

Ouverture au monde

- ✓ Interfaçable avec C/C++/FORTRAN
- ✓ Langage de script de plusieurs applications importantes
- ✓ Excellente portabilité

Disponibilité de bibliothèques

- ✓ Plusieurs milliers de packages sont disponibles dans tous les domaines

Points forts de Python :

- ✓ langage de très haut niveau ;
- ✓ sa lisibilité ;
- ✓ « langage algorithmique exécutable »

Les points faibles de Python

Comme toutes technologies, le langage Python possède des limites. En effet, malgré tous les avantages que l'on perçoit, il présente néanmoins certains points faibles. Nous allons alors voir quelques lacunes qu'il détient.

Sa simplicité

Présentée précédemment comme un point fort de Python, sa simplicité peut également devenir très vite un inconvénient. Le fait qu'il soit très facile à apprendre et que sa syntaxe est allégée au maximum rend par conséquent les développeurs Python moins aptes à l'utilisation d'autres langages qu'ils jugent trop compliqués. En plus, l'énorme bibliothèque destinée à rendre son utilisation moins fastidieuse les pousse à être dépendants à cette forme de facilité. Le passage vers une autre technologie est en conséquence assez difficile pour eux.

Sa vitesse

Python est, comme nous l'avons vu dans sa définition, un langage interprété. De ce fait, il interprète les codes ligne par ligne, ce qui engendre une certaine lenteur à l'exécution du programme. Sa nature dynamique pose également quelques soucis de vitesse.

Donc, pour une application qui a besoin d'une certaine performance à ce niveau, ce n'est pas forcément le langage à privilégier, même si ce défaut ne se fait pas ressentir au premier abord. En plus, quelques bibliothèques comme NumPy ou Pandas, qui sont des bibliothèques relatives aux traitements mathématiques et statistiques, peuvent aider à pallier ces inconvénients lors des opérations nécessitant de grandes vitesses.

Sa consommation de mémoire

À cause de son typage dynamique et sa flexibilité à ce niveau, Python est un langage très gourmand en mémoire. Ainsi, pour ceux qui souhaitent développer une application qui optimise au maximum la mémoire, choisir ce langage n'est pas une bonne idée.

Les erreurs

Ce typage peut également causer certaines erreurs d'exécution aux programmes Python. En effet, le fait que l'on n'attribue pas le type des variables au préalable et que ces mêmes variables peuvent contenir des données de types différents à divers moments du programme pose un problème à ce niveau. De nombreux tests doivent donc être effectués constamment afin d'éviter au maximum ce genre d'éventualité. Le délai gagné à l'écriture du programme est vite rattrapé par celui passé aux tests.

La difficulté en application mobile

Les lacunes qu'il possède au niveau de la vitesse ainsi que sa forte consommation de mémoire le rendent moins adapté pour le développement d'applications mobiles. En effet, à cause de ces problèmes, ces dernières peuvent être vite affaiblies. D'ailleurs, peu de personnes s'aventurent dans ce type de projet avec le langage Python. On le retrouve surtout dans les développements côté serveur.

L'accès à la base de données

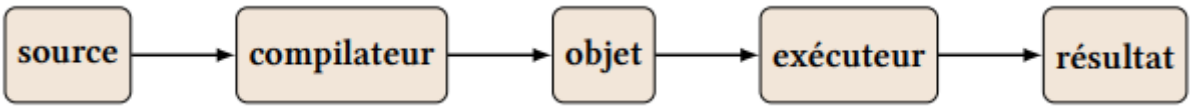
La couche d'accès à la base de données de Python s'avère plus archaïque lorsqu'on la compare avec les autres technologies populaires à l'instar de JDBC ou ODBC. Donc, lorsqu'il s'agit d'une application qui nécessite une interaction fluide et complexe avec les systèmes de gestion de bases de données, utiliser Python n'est pas vraiment conseillé.

Langage compilé et langage interprété

Langage compilé

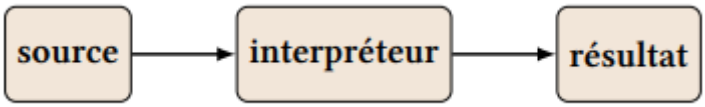
La compilation est la traduction du code source en langage objet. Elle comprend au moins quatre phases (trois phases d'analyse lexicale, syntaxique et sémantique et une phase de

production de code objet). Pour générer le langage machine il faut encore une phase particulière : l'édition de liens. La compilation est contraignante mais offre au final une grande vitesse d'exécution.



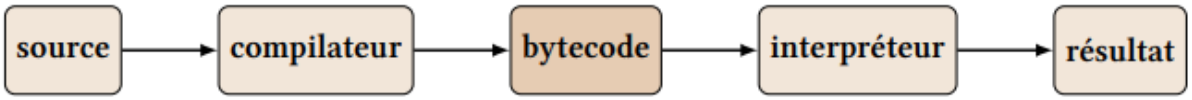
Langage interprète

Dans la technique de l'interprétation chaque ligne du code source analysé est traduite au fur et à mesure en instructions directement exécutées. Aucun programme objet n'est généré. Cette technique est très souple mais les codes générés sont peu performants : l'interpréteur doit être utilisé à chaque nouvelle exécution.



Exécution du code Python

- ✓ **Technique mixte** : l'interprétation du bytecode compilé. Bon compromis entre la facilité de développement et la rapidité d'exécution ;
- ✓ **le bytecode** (forme intermédiaire) est portable sur tout ordinateur muni de la machine virtuelle Python.



Pour exécuter un programme, Python charge le fichier source .py en mémoire vive, en fait l'analyse (lexicale, syntaxique et sémantique), produit le bytecode et enfin l'exécute. Afin de ne pas refaire inutilement toute la phase d'analyse et de production, Python sauvegarde le bytecode produit (dans un fichier .pyo ou .pyc) et recharge simplement le fichier bytecode s'il est plus récent que le fichier source dont il est issu. En pratique, il n'est pas nécessaire de compiler explicitement un module, Python gère ce mécanisme de façon transparente.

Les libraires python

- ✓ **Pandas**

Pandas est une bibliothèque créée pour aider les développeurs à travailler intuitivement avec des données « étiquetées » et « relationnelles ». Elle est basée sur deux structures de données principales : « Série » (unidimensionnelle, comme une liste Python) et « Dataframe » (bidimensionnelle, comme un tableau à plusieurs colonnes). Pandas permet de convertir des structures de données en objets DataFrame, de gérer les données manquantes et

d'ajouter/supprimer des colonnes de DataFrame, d'imputer les fichiers manquants et de tracer les données avec un histogramme ou une boîte à moustache. C'est un outil indispensable pour la manipulation et la visualisation des données.

- ✓ **Numpy**

NumPy (pour Numerical Python) est un outil parfait pour le calcul scientifique et la réalisation d'opérations de base et avancées avec des tableaux.
La bibliothèque offre de nombreuses fonctionnalités pratiques permettant d'effectuer des opérations sur des tableaux (n-arrays) et des matrices en Python. Elle permet de traiter des tableaux qui stockent des valeurs du même type de données et facilite l'exécution d'opérations mathématiques sur les tableaux (et leur vectorisation). En fait, la vectorisation des opérations mathématiques sur le type de tableau NumPy augmente les performances et accélère le temps d'exécution.

- ✓ **Scikit Learn**

Il s'agit d'une norme industrielle pour les projets de science des données basés en Python. Scikits est un groupe de paquets de SciPy qui ont été créés pour des fonctionnalités spécifiques par exemple, le traitement d'images.

- ✓ **Matplotlib**

Elle est une bibliothèque scientifique de données standard qui aide à générer des visualisations de données telles que des diagrammes et des graphiques bidimensionnels (histogrammes, diagrammes de dispersion, graphiques de coordonnées non cartésiennes).

- ✓ **Seaborn**

Seaborn est basé sur Matplotlib et sert d'outil de Machine Learning Python utile pour la visualisation de modèles statistiques – cartes thermiques et autres types de visualisations qui résument les données et dépeignent les distributions globales.

- ✓ **Keras :**

Une excellente bibliothèque pour la construction de réseaux de neurones et la modélisation

- ✓ Etc

Les mots clés python

Python compte 33 mots-clés :

```
and       del       from      None      True
as        elif      global    nonlocal  try
assert    else      if        not       while
break     except    import    or        with
class     False     in        pass      yield
continue  finally   is        raise
def       for       lambda    return
```

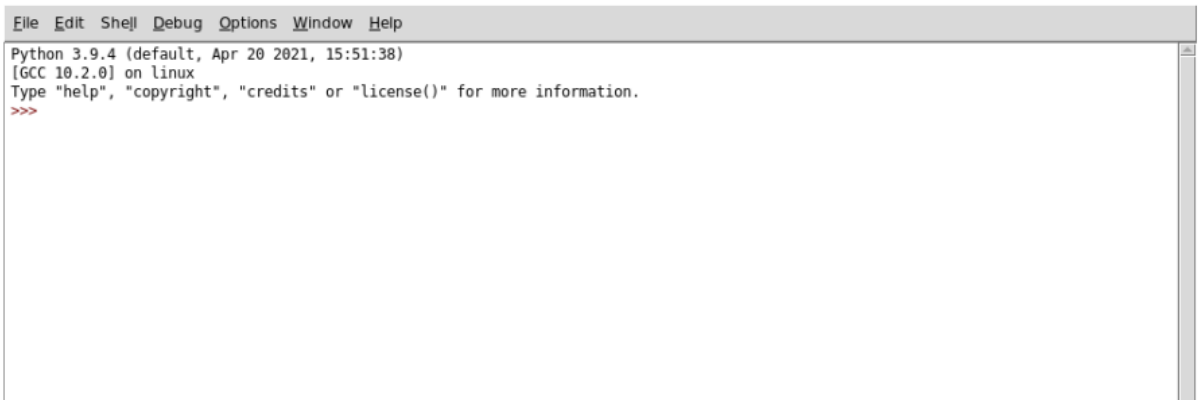
Environnement de développement

Un grand nombre d'IDE sont disponibles pour Python. On peut citer :

- ✓ **IDLE** : l'IDE par défaut de Python.
- ✓ **Spyder** : l'IDE permettant le développement d'applications Python proposant de fonctionnalités similaires a Pycharm.
- ✓ **Pycharm** : une alternative moins libre que Spyder mais avec de nombreuses fonctionnalités comme le support pour les outils de suivi de version ou la complétion automatique.
- ✓ **VSCode** : un IDE gratuit, non limité à Python, aux fonctionnalités très riches grâce à son système d'extensions.

Lancement de Python (IDLE)

Vous pouvez le trouver dans le menu Démarrer, le dossier Applications, ou autres selon votre système d'exploitation. IDLE c'est l'environnement de développement fourni par défaut avec Python (Interactive Development Environment). Il va nous permettre d'exécuter facilement du code Python quelque soit votre système. La fenêtre qui s'ouvre devrait ressembler à cela :



Calculer avec Python

Ainsi, comme nous l'avons vu précédemment, nous pouvons exécuter l'interpréteur interactif à l'aide du programme IDLE. Vous vous retrouvez maintenant face à ce que l'on appelle une invite de commande. Cela se reconnaît par les >>> en début de ligne, qu'on appelle le prompt. L'interpréteur attend que vous lui demandiez quelque chose.

Exécuter les instructions suivantes :

Addition

>>> 5 + 3

On peut lui demander d'additionner plusieurs nombres

>>> 1 + 2 + 3

Les espaces sont facultatif mais permet une bonne lisibilité

>>> 1+2+3

Soustraction

>>> 8 – 5 ; >>> 1 – 10 ; >>> 1 - -10

Multiplication

>>> 4 * 5 ; >>> 6 * 7 * -1

la multiplication est prioritaire sur l'addition

>>> 1 + 2 * 3

on peut utiliser des parenthèses pour prioriser certaines opérations

>>> (1 + 2) * 3 ; >>> 1 – 2 – 3 ; >>> 1 – (2 – 3)

Division

>>> 20 / 3 ; >>> 20 // 3 ; >>> 20.5 / 3 ; >>> 20,5 / 3

L'opérateur puissance utilise le symbole **. Pour obtenir le reste d'une division entière on utilise le symbole modulo %

>>> 2 ** 3 ; >>> 5 % 4

Commentaires

Dans un script, tout ce qui suit le caractère # est ignoré par Python jusqu'à la fin de la ligne et est considéré comme un commentaire. Les commentaires doivent expliquer votre code dans un langage humain.

```
# Votre premier commentaire en  Python.
print("Hello world!")

# D'autres commandes plus utiles pourraient suivre.
```

Déclaration des variables

Variable

Lorsque nous entrons une expression dans l'interpréteur interactif, sa valeur est calculée puis affichée dans le terminal. Mais après cela elle est perdue. Pourtant il pourrait nous être utile de conserver un résultat, afin de le réutiliser par la suite dans d'autres calculs. En effet, ce sont les variables qui vont nous permettre de stocker nos résultats de calculs. Une

variable, c'est juste un nom (identificateur)que l'on associe à une valeur (donnée), afin d'indiquer à Python de la conserver en mémoire (de ne pas l'effacer) mais aussi de pouvoir la retrouver (grâce à son nom). On peut voir la variable comme une simple étiquette qui sera collée sur notre valeur pour indiquer comment elle se nomme.

Exemple >>> resultat=4 + 5
On voit que l'interpréteur ne nous affiche rien cette fois-ci, parce que le résultat a été stocké dans résultat.

Affectation

Affectation simple

On affecte une variable par une valeur en utilisant le signe = (qui n'a rien à voir avec l'égalité en math !). Dans une affectation, le membre de gauche reçoit le membre de droite ce qui nécessite d'évaluer la valeur correspondant au membre de droite avant de l'affecter au membre de gauche.

Exemple >>> resultat=4 + 5, >>> resultat=resultat+ 1

Affectation multiples

Même valeur pour plusieurs variables

>>> x=y=8

Affectation parallèle

>>>a, b = 2.5 , 3.2

Affectation sur une ligne

>>>a=2 ; b=5 ; d=a+7

Remarque : Un nom de variable ne peut pas être composé de n'importe quels caractères. Il ne doit contenir que des lettres (minuscules ou majuscules), des chiffres et des underscores (caractère _). L'autre règle est que le nom ne peut pas commencer par un chiffre.

Types de données

Python est un language à typage dynamique, ce qui signifie qu'il n'est pas nécessaire de déclarer les variables avant de pouvoir leur affecter une valeur. La valeur que l'on affecte possède un type qui dépend de la nature des données (nombre entier, nombre à virgule, chaîne de caractères, etc). Le type du contenu d'une variable peut donc changer si on change sa valeur. Pour connaître le type d'une donnée ou le type de la valeur d'une variable, il suffit d'utiliser la fonction type()

Le type int (entier)

Ce type est utilisé pour stocker un entier, en anglais integer. Pour cette raison, on appelle ce type int.

Exemple >>> x=4

Le type float (flottant)

Ce type est utilisé pour stocker des nombres à virgule flottante, désignés en anglais par l'expression floating point numbers. Pour cette raison, on appelle ce type : float. En français, on parle de flottant.

Exemple >>> x=4.5

Le type str (chaîne de caractères)

Sous Python, une donnée de type str est une suite quelconque de caractères délimitée soit par des apostrophes (simple quotes), soit par des guillemets (double quotes). str est l'abréviation de string, qui veut dire chaîne en français.

Exemple >>> a='salut' ; >>> b= ''salut''

Le type bool (booléen)

Le type bool est utilisé pour les booléens. Un booléen prend les valeurs True ou False.

Remarque : la fonction not est un opérateur logique qui renvoie l'opposé de la valeur booléenne transmise. Pour True, on obtient False. Réciproquement, il renvoie False quand on lui transmet True.

```
a = True
>>> type(a)
bool
b = not(a)
>>> b
False
```

Le type list (liste)

Sous Python, on peut définir une liste comme une collection d'éléments séparés par des virgules, l'ensemble étant enfermé dans des crochets.

```
Exemple jour = ["lundi", "mardi", "mercredi", 1800, 20.357, "jeudi", "vendredi"]
>>> type(jour)
list
```

Le type complex (complexe)

Python posséde par défaut un type pour manipuler les nombres complexes. La partie imaginaire est indiquée grâce à la lettre « j » ou « J ». La lettre mathématique utilisée habituellement, le « i », n'est pas utilisée en Python car la variable *i* est souvent utilisé dans les boucles.

Exemple

```
a = 2 + 3j
>>> type(a)
complex
>>> a
(2+3j)
```

```
b = 1 + j
Traceback (most recent call last):
   File "<pyshell#21>", line 1, in <module>
     b = 1 + j
NameError: name 'j' is not defined
```

```
b = 1 + 1j
>>> a * b
(-1+5j)
```

```
1j**2
(-1+0j)
```

Les opérateurs et expressions

Opérateurs

- ✓ Un opérateur peut être unaire ou binaire.
- ✓ Un opérateur fonctionne avec des opérandes.
- ✓ Un opérateur est lié à un type de données.

Opérateurs arithmétiques

Lorsque les opérandes sont d'un type numérique :

- ✓ x + y # Addition
- ✓ x - y # Soustraction
- ✓ x * y # Multiplication
- ✓ x / y # Division
- ✓ x // y # Division entière

- ✓ x % y # Modulo (reste)
- ✓ -x # Opposé - opérateur unaire
- ✓ abs(x) # Valeur absolue - opérateur unaire
- ✓ x ** y # Puissance

Opérateurs comparatifs

Pour former des expressions booléennes :

x < y # Inférieur

x <= y # Inférieur ou égal

x > y # Supérieur

x >= y # Supérieur ou égal

x == y # Égal (attention !)

x != y # Différent

x <> y # Différent

x is y # Identité

x is not y # Non identité

Opérateurs logiques

Pour réunir des expressions booléennes :

x and y # Intersection

x or y # Union

not y # Négation

Opérateurs sur les séquences

Pour manipuler les séquences :

x in s # Inclusion - Vrai si x est dans s
x not in s # Non inclusion - Vrai si x n'est pas dans s
s1 + s2 # Concaténation
s[i] # Accès - Renvoie l'élément d'indice i
s[i:j] # Accès - Renvoie les éléments d'indice i à j non inclus
s[-1] # Accès - Renvoie le dernier élément
s[-2] # Accès - Renvoie l'avant dernier élément
len(s) # Longueur - Renvoie le nombre d'éléments
min(s) # Minimum - Renvoie le plus petit élément
max(s) # Maximum - Renvoie le plus grand élément

Expression

Une expression :

- est un ensemble de variables, de littéraux, d'appels de fonction et d'opérateurs,
- peut être très simple ou infiniment compliquée,
- peut être évaluée, elle a une valeur.

Exemple

```
>>>2                          # Une expression très simple
>>>nom + " " + prenom    # Une autre...
>>>(1 + math.sqrt(5))/2   # un peu moins simple
```

Lorsqu'une expression est complexe, il est préférable de la décomposer pour améliorer la lisibilité du code.

Les fonctions prédéfinis

Python est livré avec un ensemble de fonctions intégrées pour effectuer les tâches les plus élémentaires :

Les fonctions sur les types numériques

La fonction arrondi, round en Python : elle retourne l'entier le plus proche d'un nombre donné.

Par exemple :

round(5.1) donne 5 ; round(5.5) donne 6 ; round(5.7) donne 6

La fonction racine-carrée est sqrt en Python : cette fonction retourne la racine-carrée d'un nombre donné. Le résultat de la racine est toujours de type réel.

Pour pouvoir utiliser la fonction sqrt en Python, vous devez l'importer de la fonction math : ***from math import sqrt***, ou ***from math import ****. Import * signifie importer toutes les fonctions de la bibliothèque math.

Exemple :

sqrt(2) retourne 1.4142135623730951 ; sqrt(4) donne 2.0

La fonction qui permet de donner la valeur absolue d'un nombre en algorithme et en Python est abs.

Exemple : abs(4) donne 4 ; abs(-10) donne 10 ; abs(-2.5) donne 2.5

La fonction aléa, randint en Python retourne une valeur entière aléatoire. La syntaxe en Python est ***: randint(borne_inferieur, borne_superieure)***. L'importation from random import randint est indispensable.

Exemple :

randint(1, 6) peut donner 2 ; randint(1, 6) peut donner 5 ; randint(1, 6) peut donner 4 ; randint(10, 100) peut donner 70

Les fonctions de conversions

La fonction ent, trunc en Python retourne la partie entière d'un nombre donné. Cette fonction exige l'importation de la bibliothèque math. Cette fonction donne le même résultat que la fonction int en Python.

Exemple :

trunc(63.32) = int(63.32) = 63 ; trunc(80.87) = int(80.87) = 80 ;
trunc(87.20) = int(87.20) = 87 ; trunc(64.37) = int(64.37) = 64

Les fonctions sur les caractères

Un caractère peut être une lettre ("P", "y", 't', 'h'), un chiffre ("1", "5", '4') ou un symbole (un caractère spécial : "@", "#", ".",). Les caractères sont ordonnés selon leurs codes ASCII. Un caractère doit être limité par deux simples quottes ', ou des double quottes " ; "a" est égale à 'a'. Il faut savoir que le langage de programmation Python n'a pas de type spécifique pour les caractères, il utilise le type chaine (str). Python considère un caractère comme étant une chaine composée d'un seul caractère.

Les fonctions que je peux appliquer sur les caractères sont : L'ordre d'un caractère dans la table ASCII, noté en algorithme et en Python ***ord***.

Par exemple : *ord("a") donne 97 ; ord("b") donne 98 ; ord('A') donne 65*

La fonction ***chr***, retourne le caractère dont le code ASCII est passé comme paramètre.

Exemple : ***chr(65) vaut "A" ; chr(98) vaut "b***"

Les fonctions sur les chaines de caractère

Une chaine de caractères contient zéro ou plusieurs caractères. Une chaine avec zéro caractère est appelée la chaine vide. Elle a une longueur nulle et elle contient la chaine "".

Parmis les fonctions que nous pouvons appliquer sur les chaines en python :

long(ch) en algorithme, len(ch) en Python

Le rôle de la fonction len est de calculer le nombre de caractères dans la chaine ch.

Exemple : *len("python") donne 6. len("a b.") donne 4.*

La fonction pos retourne la première position de la chaîne ch1 dans la chaîne ch2. Si la chaine n'est pas trouvée alors la fonction pos retourne -1. Il ne faut pas oublier que Python commence à compter les caractères à partir de 0 ; en Python le premier caractère est numéro zéro. *pos(ch1, ch2) en algorithme, ch2.find(ch1) en Python*

Exemple : *"bonjour".find("jour") donne 3. ; "bonbon".find("bon") donne 0. ; "python.pm".find("www") donne -1.*

La fonction str transforme une valeur numérique en une chaine. Exemple

str(22) donne "22". ; str(1.50) donne "1.5" ;

valeur(ch) en algorithme, int(ch) ou float(ch) en Python. Ces deux fonctions permettent de convertir une chaine en une valeur numérique, si c'est possible. Exemple : *int('2030') donne 2030 ; float("1.550") donne 1.55 ; int("Python") génère une erreur : invalid literal for int()* ;

estnum(ch) en algorithme, ch.isdigit() ou ch.isdecimal() en Python. Cette fonction vérifie si une chaine est numérique : composée uniquement de chiffres. isdigit retourne Vrai (True) si la chaine est numérique, Faux (True) dans le cas contraire. La fonction isdigit est très utile. Transformer une chaine en une valeur numérique peut générer une erreur, donc il faut tester que tous les caractères sont numérique avant utiliser la fonction int(). Exemple :

"20".isdigit() retourne True ; "30".isdecimal() retourne True ; "22,33".isdigit() retourne False ; "Python".isdigit() retourne False ; "20.30".isdigit() retourne False

upper permet de convertir une chaine en majuscule. Exemple :

"Python".upper() donne 'PYTHON' ; 'ABC 123 xyz'.upper() donne 'ABC 123 XYZ'

La fonction ch[d:f] en Python. Cette fonction retourne une partie de la chaîne ch en commençant de la position de début d jusqu'à la position finale f, sans prendre en considération le caractère f.

La fonction ch[:d] + ch[f:] en Python. Ce traitement est utiliser pour supprimer des caractères à partir de la position de début d jusqu'à la position f (f exclue) de la chaîne ch.

Les fonctions d'entré sortie

La fonction **print()** sert à afficher les valeurs des variables d'un programme.

Exemple

print ("il fait chaud")
>>> il fait chaud

La fonction **input** sert à **à saisir au clavier** une valeur d'une variable. La fonction input() s'écrit **en minuscules et toujours suivie de parenthèses** qui contient l'information à saisir. Attention par défaut, la fonction input() saisit **une chaîne de caractères.**

Exemple a=input ("quel est votre nom ")

Les références

https://python.developpez.com/cours/apprendre-python3/?page=page_4#L4

https://courspython.com/introduction-python.html

Cours Matthieu Boileau & Vincent Legoll Apprendre Python pour les sciences CNRS - Université de Strasbourg.

Cours de Python Introduction à la programmation Python pour la biologie Patrick Fuchs et Pierre Poulain https://python.sdv.univ-paris-diderot.fr/ .