UNIVERSITÉ PARIS 8 VINCENNES-SAINT-DENIS

Méthodologie de la Programmation

Palus Jean-Pascal

Licence IV - L1 - Semestre I

Bureau A184 - jpp@up8.edu

Séance II

Fonctions

Fonctions

- Python supporte les expressions avec des fonctions ressemblant aux fonctions mathématiques.
  - Une fonction contenu dans une expression est nommé un *appel de fonction*.
- Les *appel de fonction* sont de la forme :
  - **nom_de_la_fonction**(argument_1, argument_2, ...)
  - Les arguments peuvent être tous types d'expressions.
  - Quand une fonction attend plusieurs arguments, ceux-ci sont séparés par des **virgules**.

Fonctions built-in

- Certaines fonctions existent déjà dans le langage.

Fonctions built-in

- Certaines fonctions existent déjà dans le langage.
- Certaines fonctions mathématiques :
  - **round**(2.35)
  - **max**(a+3, 24)

Fonctions built-in

- Certaines fonctions existent déjà dans le langage.
- Certaines fonctions mathématiques :
  - **round**(2.35)
  - **max**(a+3, 24)
- Les fonctions de cast :
  - **int**(), **float**(), **bool**(), **str**()

Fonctions built-in

- Certaines fonctions existent déjà dans le langage.
- Certaines fonctions mathématiques :
  - **round**(2.35)
  - **max**(a+3, 24)
- Les fonctions de cast :
  - **int**(), **float**(), **bool**(), **str**()
- Quelques autres :
  - **type**(), **exit**(), **print**(), ...
  - La liste complète se trouve dans le doc Python : https://docs.python.org/3/library/functions.html

Fonctions built-in

- La plupart des fonctions sont des **expressions**.
  - On peut les utiliser pour assigner des **valeurs** à des **variables**.
  - Exemple : x = **round**(2.27)

Fonctions built-in

- La plupart des fonctions sont des **expressions**.
  - On peut les utiliser pour assigner des **valeurs** à des **variables**.
  - Exemple : x = **round**(2.27)
- Mais certaines fonctions sont des commandes.
  - Elle ne sont généralement utiles que dans l'interpréteur Python.
  - **help**(), **quit**()

Fonctions built-in

- Il y a assez peu de fonctions directement implémentées dans le langage.
- Il en manque donc que l'on pourrait s'attendre à trouver.
  - **cos**(), **sqrt**(), ...

Fonctions built-in

- Il y a assez peu de fonctions directement implémentées dans le langage.
- Il en manque donc que l'on pourrait s'attendre à trouver.
  - **cos**(), **sqrt**(), ...
- Beaucoup d'autres fonctions sont disponibles via des **modules**.

Modules

- Les modules sont des bibliothèques de **fonctions** et de **variables**.
- Pour accéder à un module on utilise le mot-clef ***import***.
  - **import** module
- On accède aux fonction du module de la manière suivante :
  - module.**fonction**(arguments)

Modules couramment utilisés

- io
  - input / outputs
  - **open**(), **close**()
- random
  - Générer des nombres random.
  - **seed**(), **randrange**(), **randint**(), **random**(), **shuffle**()
- string
  - Manipulation de chaînes de caractères
  - **format**()

Modules couramment utilisés

- sys
  - Fonctions système
  - **argv**(), **getsizeof**(), **path**(), **stdin**(), **stdout**(), **stderr**()
- os
  - D'autres fonctions système
  - Cross-platform
  - **chdir**(), **getcwd**(), **fchmod**(), **kill**()
  - **EX_OK**, **EX_OSERR**,

Créer ses propres modules

- Pour importer un module depuis la console de l'intépreteur python il faut que celui-ci se trouve dans le même dossier que le module que l'ont veut importer.
- Mettre une docstring permet d'ajouter du contenu à ce qui sera affiché par la fonction **help**().

Différences modules / scripts

?

Différences modules / scripts

- Scripts :
  - Se comportent comme des applications
  - Se lancent via la commande **python3**.
- Modules :
  - Fournissent des fonctions et des variables aux scripts.
  - Ne sont (généralement) pas exécutables.
- Ce sont tous des fichiers source, la différence est dans leur utilisation.

Mot-clef *from*

- L'importation de module peut également se faire de la manière suivante :
  - **from** module **import** fonction

  ! Attention aux noms des fonctions. Pas de surcharge possible en Python.

Mot-clef *from*

- L'importation de module peut également se faire de la manière suivante :
  - **from** module **import** fonction
- On peut aussi importer tout le module de cette manière :
  - **from** module **import ***

- Ne pas oublier que lorsqu'on importe un module, la totalité de ce qui est exécutable est exécuté.

Mot-clef **as**

- On peut renomer localement (alias) un module importé avec le mot-clef **as**.

Rendre un script intéractif

- Grâce à la fonction **input**()

! La fonction **input**() renvoie un **str**.

Importer des fonctions

- Importer les fonctions que vous avez écrites se fait de la même manière que pour les modules build-in.

Rappels

- Les variables hors de toute fonction sont globales.
- On peut accéder à la valeur d'une variable globale depuis n'importe où.
- Mais pour la modifier il faut utiliser le mot-clef **global**.

Scope des modules

- Les modules n'ont accès qu'aux variables globales de leur propre fichier.
- Le script a accès aux variables globales des modules qu'il a importé.

scope main

- De nombreux langages de programmation ont une fonction spéciale qui est exétutée automatiquement en premier lorsque le programme est lancé.
- Lorqu'un programme python est exécuté, l'interpréteur chercher un mot-clef particulier à exécuter en premier :
  - **__main__**

argv

```c
// main.c

#include <stdio.h>

int
main(int argc, char *argv[]) {

    printf("Nombre d'arguments: %d\n", argc);

    for (int i = 0; i < argc; i++) {
        printf("Argument %6d: %s\n", i, argv[i]);
    }

    return 0;
}
```

argv

- **sys.argv**
- Contient la liste des arguments du programme.

argc

?

argc

- **len**(sys.**argv**)-1)

Shebang

- Il est possible de rendre un script Python directement exécutable, sans avoir à le passer en argument de la commande **python3**.
- Pour cela il faut faire deux choses :
  - Le fichier doit avoir les droits d'exécution.
  - Indiquer quel interpréteur est chargé de son exécution.
- Un shebang commence par **#!**, puis est suivi du chemin vers l'interpréteur :
  - **#!/usr/bin/python3** par exemple

Python Package Index

- Puisque tout le monde peut écrire ses propres modules il en existe une grande quantité.
- Beaucoup peuvent être installée depuis PyPI
  - https://pypi.org/
- L'outil **pip3** est fourni avec Python est permet d'installer des paquets depuis PyPI.
  - https://packaging.python.org/tutorials/installing-packages/