Formation au langage de programmation Python

Initiation à Python

Partie II
interpréteurs – types prédéfinis – exercices

Formateur : IBRAHIM M. S.
ibrahim.ms@gmail.com

du 17/10 au 20/10 2016

Plan

Testez dans un terminal la version de Python installée

```
# python
>>> print " La configuration installée est opérationnelle."
```

Messages d'erreur possibles

1 : Python non trouvé : il faut alors mettre à jour la variable système $PATH

2 : Code non interprété (parenthéses) – c'est normal ! la machine est utilisable

3 : pas de message d'erreur : Python 2.7.xy installée — mettre à jour

La version installée ou prioritaire est

```
# python

>>> from sys import version as v ;print(v)
3.4.5 |Anaconda 2.3.0 (x86_64)| (default, Jul  2 2016, 17 :47 :57)
[GCC 4.2.1 Compatible Apple LLVM 4.2 (clang-425.0.28)]
```

- Python 3.x.y installée – c'est bon ! pas de problème, vous pouvez continuer

Python en mode interactif : interpréteurs

entre autres : python, iPython, bPython, notebook, idle, terminal, Jupiter

Invite de commande

- on lance python
- on saisi les commandes
- on récupére le résultat
- on dispose d'un historique

Indentation

- organise le programme
- aide de l'éditeur de texte
- facilite (re)-lecture de code
- éviter les tabulations

Indentation

- délimiteur de fin de bloc
- constitutif du langage
- structure le programme
- nombre d'espaces fixe

Séparateurs

- ; instructions sur 1 ligne
- , constructeur de séquences
- : syntaxe pour for if while
- ⟶ structure du programme

Types numériques prédéfinis

Entiers longs : int

- pas de limite de taille
- x**y puissance
- // quotient % reste
- divmod(a,b) -> (q,r)

Complexes : complex

- a = complex(x,y)
- a = x + y * 1j
- a.real, a.imag , abs(a)
- a.conjugate()

Flottants : float

- approximation des réels
- conversions implicites
- a/b $\neq$ a//b
- a**b = $e^{b \cdot ln\ a}$

Booléens : bool

- True / False
- b == a , a != b , not c
- 1 < 4 , z is s
- x in R , a is not c

Le type string : chaîne de caractères

Chaînes de caractères : string

- 'caracteres' ou "caracteres"
- mot = "ceci n'est pas un mot"
- nb = len(mot)
- B = 'non, '
- mot is B
- B+mot (concaténation)
- mot[0], mot[-1], mot[7 :11]
- mot[9 :-2], mot[-2 :3], mot[:-1]
- mot[-9 :-2], mot[4 :2]
- for i in mot : print(i)
- pas de caractère simple (char)
- Unicode pris en charge

Opérations possibles

- mot.upper()
- mot.lower()
- mot.isdigit()
- mot.isalpha()
- mot.strip(" ,'- ?!")
- mot.count("e")
- mot.split(" ")
- mot.format()
- mot.startwith("a")
- mot.endswith("a")
- mot.join('ajout de mots')
- ...

Typage – conversion – évaluation

Typage dynamique faible

- l'interpréteur déduit
- type le plus approprié
- le met à jour si besoin
- pas toujours le meilleur

Connaitre le type

- type(variable)

Conversion

- float() – int() – str()
- x = input("x=(entier)")

Opérations disponibles

- dir(variable)
- dir(type)

eval

- eval("string")

exec

- exec("string")

Un type spécial

- None

Mots-clés et mots réservés du langage

Attention !

- seuls les mots clés ne peuvent être redéfinis — à éviter tout de même

Mots clés : eux seuls sont essentiels

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

Fonctions

help() dir() print() input()
raw_input() len() range() ord()
locals() globals() str() int()

Modules : importés avant tout import

anydbm array atexit bisect calendar cmath codecs collections
commands ConfigParser copy ctypes datetime decimal dummy_thread
dummy_threading exceptions encodings.aliases formatter heapq
gettext locale linecache marshall math mmap operator os pickle
Queue re shelve shutil signal stat string StringIO struct subprocess sys
textwrap tempfile thread threading time timeit traceback unicodedata
xml.sax warnings whichdb _winreg

Commentaires

- # commentaire court
- """ sur plusieurs lignes """
- """ docstrings """
- import proj ; proj.__doc__

Conteneurs : types composés prédéfinis

Tuple

- élèments de types quelconques
- entre `( )` — séparateur `,`
- singleton (a,) et vide ( )
- immuable c-à-d figé
- accés par index

Ensemble : set

- sans répétition,
- non ordonnés
- entre `{ }` — séparateur `,`
- singleton {a} et vide { }
- fonction set()

Liste

- élèments de types quelconques
- entre `[ ]` — séparateur `,`
- singleton [a] et vide [ ]
- ajout insertion suppression
- accés par index

Dictionnaire

- tableau associatif de type
  key $\rightarrow$ value
- $\{k_1 : v_1, k_2 : v_2, k_3 : v_3, ...\}$
- accés par la clé
- unicité des clés

Principales opérations sur les conteneurs

Tuple

- t[i :j :k] – len(t) – in
- tuple() – conversion
- t[0] – lecture possible
- t[0] = 2 – erreur en écriture

Ensemble : set

- | : ∪ — & : ∩ — < : ⊂
- ^ : sym-diff — ∪ — - : diff
- set() — conversion
- .add(x) – .remove(x) – len()

Liste

- t[i :j :k] – del[i :j :k]
- len(l) – sum(l) – in
- list() – conversion
- l[0] – lecture possible
- l[0] = 2 – écriture possible

Dictionnaire

- d[key] = value — len()
- in – max() : sur les clés
- dict() — conversion
- .keys() – .values() – .items()
- iter_items|keys|values()

Exercices

Exécuter et interpréter les résultats

```
liste=['P','6','—','U', 'P', 'M', 'C', '␣', 'U', 'n', 'i', 'v', 'e', 'r', 's', 'i', 't', 'e', '␣', 'P', 'a', 'r', 'i', 's', '␣', 'V','I']
list(liste)
tuple(liste)
str(liste)
dict((x,0) for x in liste)
set(liste)
len(liste)


chaine='P6—UPMC␣Universite␣Paris␣VI'
list(chaine)
tuple(chaine)
str(chaine)
dict((x,0) for x in chaine)
set(chaine)
len(chaine)


ensemble={'P','6','—','U', 'P', 'M', 'C', '␣', 'U', 'n', 'i', 'v', 'e', 'r', 's', 'i', 't', 'e', '␣', 'P', 'a', 'r', 'i', 's', '␣', 'V','I'}
list(ensemble)
tuple(ensemble)
str(ensemble)
dict((x,0) for x in ensemble)
set(ensemble)
len(ensemble)

nuplet=('P','6','—','U', 'P', 'M', 'C', '␣', 'U', 'n', 'i', 'v', 'e', 'r', 's', 'i', 't', 'e', '␣', 'P', 'a', 'r', 'i', 's', '␣', 'V','I')
list(nuplet)
tuple(nuplet)
str(nuplet)
dict((x,0) for x in nuplet)
set(nuplet)
len(nuplet)
```

Exercices 1/2

Exercice 01 – maxint

- maxint = ...

Exercice 01 – solution

- il n'y en a pas

Exercice 02

- 1+2+...9999999 = ...

Exercice 02 – solution

- sum(range(9999999+1))

Exercice 03

- 7+14+..+2199113 =
- 7+14+..+2199113 =

Exercice 03 – solution

- sum(range(7,2199113+1,7))
- sum([i for i in range(1, 2199113+1) if i%7==0])

Exercices 2/2

KasparovKarpov = 1. e4 e5 2. Nf3 Nc6 ... ... 41. Nf7"

- Kasparov : e4 Nf3 ... ... Nf7
- Karpov : e5 Nc6 ... ... Rb8
- liste inversée des déplacements : Nf7 Rb8 ... ... e5 e4

Proportion de triplets (a,b,c) $\in \{a, .., z, A, ..Z\}$

- strictement distincts, puis strictement croissants

gnu = "GNU GENERAL PUBLIC LIC ... lgpl.html>."

- nombre de caractères, nombre de symboles différents
- dictionnaire des fréquences des symboles
- liste des fréquences décroissantes des symboles

KasparovKarpov = 1. e4 e5 2. Nf3 Nc6 ... ... 41. Nf7"

- Kasparov : Kasparov = KasparovKarpov.split(' ')[1 ::3]
- Karpov : Karpov = KasparovKarpov.split(' ')[2 ::3]
- coups = KasparovKarpov.split(' ')
  ([coups[i] for i in range(len(coups)) if i % 3 != 0])[::-1]

Proportion de triplets (a,b,c) $\in \{a, .., z, A, ..Z\}$

- [(x, y, z) for x in range(13) for y in range(13) for z in range(13) if len(set((x,y,z)))==3]
- [(x, y, z) for x in range(52) for y in range(52) for z in range() if x>y>z]

gnu = "GNU GENERAL PUBLIC LIC ... lgpl.html>."

- len(gnu), len(set(gnu))
- fr = dict((c,gnu.count(c)) for c in set(gnu))
  li_fr = list((x,fr[x]) for x in fr.keys())
- li_fr.sort(reverse=True)
  li_fr.sort(key=lambda b :b[1] , reverse=True)