CHAPITRE 3
LES TYPES DE BASE – PARTIE 1

Programmation en Python

Types de données

- En Python, il est possible de manipuler différents types de données :
  - nombres entiers
  - nombres réels (à virgules)
  - nombres complexes
  - booléens (le vrai et le faux)
  - chaînes de caractères
  - listes
  - tuples (listes de taille est non modifiables)
  - dictionnaires

Les entiers

- Les nombres entiers, appelé « integers » en anglais, désignent les nombres sans virgules, positifs ou négatifs.
- En Python, les nombres entiers sont des objets de type `int` (abréviation de integer).

```
>>> x = 3
>>> type(x)
<class 'int'>
>>>
```

Les entiers

- Comme on a déjà vu informellement au chapitre 1, il est possible d'effectuer des opérations mathématiques sur les entiers en utilisant les opérateurs arithmétiques usuels :
  - addition : opérateur « + »
  - soustraction : opérateur « - »
  - multiplication : opérateur « * »
  - puissance : opérateur « ** »
  - division entière : « / » et « // » en Python 2 et 3, resp.
  - modulo : opérateur « % »

Les entiers


```
>>> 7+3
10
>>> 7-3
4
>>> 7*3
21
>>> 7**3 # puissance
343
>>> 7//3 # division entière (résultat sans les décimales)
2
>>> 7%3
1
```

Les entiers

- En Python 2, lorsque nous utilisons la barre de division classique « / », nous effectuons une division entière. En python 3, nous effectuons une division classique et donc obtenons un nombre à virgule, de type float et non de type int (cf. slides suivants).

```
>>> 7/3 # Python 2, division entière (sans décimales)
2
>>>
>>> 7/3 # Python 3, division normale (avec décimales)
2.3333333333333335
>>>
```

Les entiers

- Puisque la mémoire de l'ordinateur est finie, nous ne pouvons évidemment pas représenter l'infinité des nombres entiers en Python, il y a un maximum et un minimum.

```
>>> from os import sys
>>> # importation de la fonction sys de la librairie os
>>> sys.maxsize
9223372036854775807
>>>
```

Les réels ou flottants

- Les nombres réels ou flottants, appelé « floating numbers » en programmation, désignent les nombres à virgules, positifs ou négatifs.
- En Python, les nombres réels sont des objets de type `float`.

```
>>> x = 2.54
>>> type(x)
<class 'float'>
>>> y = 3.0
>>> type(y)
<class 'float'>
```

Les réels ou flottants

- On effectue des opérations mathématiques sur les réels en utilisant les opérateurs arithmétiques usuels :
  - addition : opérateur « + »
  - soustraction : opérateur « - »
  - multiplication : opérateur « * »
  - division : opérateurs « / » ou « // » en Python 2 et 3
  - puissance : opérateur « ** »
  - modulo : opérateur « % »

Les réels ou flottants


```
>>> 7.2 + 3.4
10.6
>>> 7.232 - 3
4.232
>>> 7 * 3.0
21.0
>>> 7.0 ** 3
343.0
>>> 7 / 3.23
2.1671826625387
>>> 7.0 % 3
1.0
```

Programme (faire démo)


```
from math import * # importation de la librairie math

print("Soit l'équation du second degré a*x^2 + b*x + c:")

a = input("Entrer le coefficient a: ")
b = input("Entrer le coefficient b: ")
c = input("Entrer le coefficient c: ")

a = float(a)
b = float(b)
c = float(c)
```

Programme (faire démo)


```
delta = b**2 - 4*a*c

if delta < 0:
        print("Pas de solution...")
elif delta == 0:
        print("Une seule solution:")
        print(-b / (2*a))
else:
        print("Deux solutions:")
        print( (-b + sqrt(delta)) / (2*a) )
        print( (-b - sqrt(delta)) / (2*a) )
```

Les complexes

- Les nombres complexes désignent les nombres de la forme a + i*b, avec a et b réels et .
- En Python, les nombres complexes sont des objets de type `complex`.

```
>>> x = 1.5 + 0.5j
>>> type(x)
<class 'complex'>
>>> x.real    # partie réelle de x
1.5
>>> x.imag    # partie imaginaire de x
0.5
```

Les complexes


```
>>> x = 1.5 + 0.5j
>>> y = -3 + 7j
>>> x + y
(-1.5+7.5j)
>>> x - y
(4.5-6.5j)
>>> x * y
(-8+9j)
>>> a, b = 2, 3
>>> c = a + b*1j # création du nombre complexe a + ib
>>> c
(-2+3j)
```

Programme (faire démo)


```
from math import * # importation de la librairie math

print("Soit l'équation du second degré a*x^2 + b*x + c:")

a = input("Entrer le coefficient a: ")
b = input("Entrer le coefficient b: ")
c = input("Entrer le coefficient c: ")
a = float(a)
b = float(b)
c = float(c)

delta = b**2 - 4*a*c
```

Programme (faire démo)


```
if delta < 0:
        print("Les solutions sont complexes:")
        aux = 0 + (sqrt(-delta))*1j # nb complexe sqrt(delta)
        print( (-b + aux) / (2*a) )
        print( (-b - aux) / (2*a) )
elif delta == 0:
        print("Une seule solution:")
        print( -b / (2*a) )
else:
        print("Deux solutions:")
        print( (-b + sqrt(delta)) / (2*a) )
        print( (-b - sqrt(delta)) / (2*a) )
```

Les booléens

- En logique, les valeurs booléennes sont le « vrai » et le « faux ».
- En Python, le vrai et le faux sont désignés par `True` et `False`.
- Les booléens sont des objets de type `bool`.

```
>>> x = True
>>> y = False
>>> type(x)
<class 'bool'>
>>> type(y)
<class 'bool'>
```

Les booléens

- En Python, il existe cinq opérations possibles sur les booléens :
  - le test d'égalité : opérateur binaire « == »
  - le test de non égalité : opérateur binaire « != »
  - négation : opérateur unaire « not »
  - conjonction (« et » logique) : opérateur binaire « and »
  - disjonction (« ou » logique) : opérateur binaire « or »

Les booléens

- La négation, i.e. l'opérateur unaire « not », inverse la valeur de vérité de la variable booléenne.
- La table de vérité de cet opérateur est :

Variable	**not Variable**
True (1)	False (0)
False (0)	True (1)

Les booléens


```
>>> not True
False
>>> not False
True
>>> 3 <= 4
True
>>> not(3 <= 4)
False
>>>
```

Les booléens

- La conjonction, i.e. l'opérateur binaire « and », renvoie la valeur « vrai » ssi les deux variables booléennes sont vraies, et revoie la valeur « faux » sinon.
- La table de vérité de cet opérateur est :

Var 1	Var 2	Var 1 and Var 2
True (1)	True (1)	True (1)
True (1)	False (0)	False (0)
False (0)	True (1)	False (0)
False (0)	False (0)	False (0)

Les booléens


```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
>>> (3 <= 4) and (5 > 2)
True
>>> (3 <= 5) and (6 == 2)
False
```

Les booléens

- La disjonction, i.e. l'opérateur binaire « or », renvoie la valeur « faux » ssi les deux variables booléennes sont fausses, et revoie la valeur « vraie » sinon.
- La table de vérité de cet opérateur est :

Var 1	**Var 2**	**Var 1 or Var 2**
True (1)	True (1)	True (1)
True (1)	False (0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

Les booléens


```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
>>> not (3 <= 4) or (5 > 2)
True
>>> not (5 <= 2) or (6 == 2)
True
```

Programme (faire démo)


```
name = input("Name? ")
password = input("Password? ")
if name == "Dan" and password == "123456":
    print("Welcome...")
else:
    print("Access denied.")
```