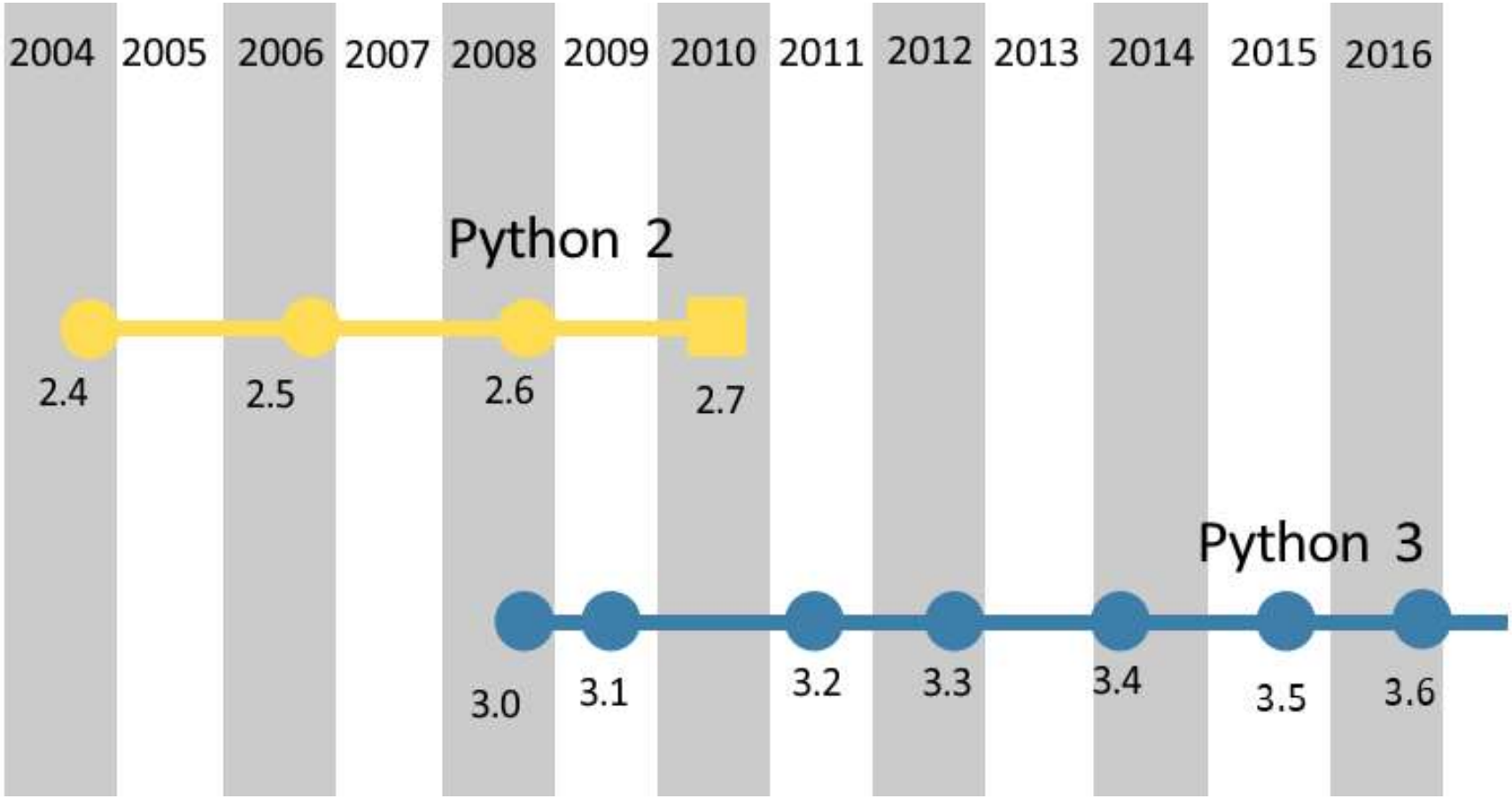
Le langage Python

Introduction

- Python
  - Langage de programmation orienté objet
  - Supporte les types dynamiques
  - Première version publique: 0.9 en 1991

- Implémentations:
  - Cpython (c), Jython (JVM) , IronPython (dotnet), pypy (Rpython)

Python 2.x vs. Python 3.x



Notion de bloc d’instruction

- Un **bloc d'instructions** est une suite d'instructions qui ont le même retrait gauche. Les blocs d'instructions sont crées par les instructions de contrôles comme *if*, *while* et *for*, ainsi que par les instructions permettant de déclarer des **fonctions**.

```
# Ce bloc d'instruction est colle contre le bord gauche du fichier
print "Je suis dans le premier bloc"
print "Je suis toujours dans le premier bloc"a=3
b=3
if (a == 12) :
    print "Je suis dans le second bloc"
    print "Je suis encore dans le second bloc"
if (b == 13 ) :

        print "Je suis dans un troisieme bloc"
        print "et ici aussi"
print "Je reviens dans le second bloc"
print "Je suis revenue dans le premier bloc"
```

Variables

Les variables sont créées automatiquement lors de leurs première utilisation.

Affectations

simples

x=7

ici x = 7

multiples

x=y=7

ici x et y = 7

parallèles

x,y = 7,8

ici x=7 et y=8

Types

- Typage dynamique
  - La fonction type() permet de connaître le type d'une variable

Liste des types

int: Nombre entier

Long: Nombre entier de taille arbitraire

*Float:*Nombre à virgule flottante

Complex: Nombre complexe

*Str:*Chaîne de caractère

*Unicode:*Chaîne de caractère unicode

*tuple:*Liste de longueur fixe

List:Liste te de longueur variable

*dict:*dictionnaire

File

Bool

NoneType

NotImplementedType

Function

Module

Tableau des types Python 3.x[1]

nom	description	équivalent 2.x
int	nombre entier de longueur arbitraire	long
byte	chaîne de caractères ASCII	str
str	chaîne de caractères Unicode	unicode

Les chaînes de caractères

```
# -*- coding: utf-8 -*-
#chaines de caractères

print "une chaîne"

print 'une autre chaîne'

print """encore

une chaîne de caractères
sur
plusieurs lignes"""
```

```
>>> s = '012345'
>>> s[3]
'3'
>>> s[1:4]
'123'
>>> s[2:]
'2345'
>>> s[:4]
'0123'
>>> s[-2]
'4'
```

- Format

```
>>> “Un, %d, trois" % 2
Un, 2, trois '
>>> "%d, deux, %s" % (1,3)
'1, deux, 3'
>>> "%s deux %s" % (1, ‘trois ')
'1 deux trois '
```

Opérations

```
>>> a=‘Hello’
>>> a[1:3]
'el'
>>> a[1:]
'ello'
>>> a[-4:-2]
'el'
>>> a[:-3]
'He'
>>> a[-3:]
'llo’
```

H	e	l	l	o
0	1	2	3	4
-5	-4	-3	-2	-1

Opérateurs de comparaison

```
>>> 5.0 == 5
True
>>>  6 != 2*3
False
>> not(-2 >= 1)
True
>>> (-2 >= 1) or (-2 <= -1)
True
>>> (-2 >= 1) and (-2 <= -1)
False
```

La conversion des types

Il existe plusieurs fonctions qui permettent de forcer le type d'une variable en un autre type.

- int() : permet de modifier une variable en entier. Provoque une erreur si cela n'est pas possible.
- long() : transforme une valeur en long.
- La fonction *str()* permet de transformer la plupart des variables d'un autre type en chaînes de caractère.
- float() : permet la transformation en flottant.
- repr() : similaire à str. Voir la partie sur les objets
- eval() : évalue le contenu de son argument comme si c'était du code Python.

IF

```
if a > 10 :
    print ("a est plus grand que dix")
elif a == 10:
    print ("a est égal à dix")
else:
    print ("a est plus petit que dix")
```

while

```
i=0
while i<5:
   i=i+1
   print i,
```

Fonctions

```
def function_name(input1, input2, …):
  command1
  command2
  …
```

listes

- Une liste ordonnées d'éléments qui peuvent être de différents types, mêmes opérations que pour les chaînes de caractères

```
>>> x = [1,'hello', (3 + 2j)]
>>> x
[1, 'hello', (3+2j)]
>>> x[2]
(3+2j)
>>> x[0:2]
[1, 'hello']
```

* modification

```
>>> x = [1,2,3]
>>> y = x
>>> x[1] = 15
>>> x
[1, 15, 3]
>>> y
[1, 15, 3]
>>> x.append(12)
>>> y
[1, 15, 3, 12]
```