Python Tutorial

Nicolas Pécheux
Alexandre Allauzen

LIMSI/CNRS — Univ. Paris Sud

January 11, 2017

Why Python ?

- It's simple
- It's easy to read
- It's fun
- It's free
- Works on Windows, Linux, Max, ...

Best way to learn it ⇒ program something for fun with it

An interpreted programming language

- Interpreted language: no compilation needed, code is directly executed by an interpretor

```
$ python my_source_code.py
```

- Interactive mode

```
$ python
>>> a = 2
>>> print a
2
>>> a = a + 2
```

Part I

First steps with Python

Variables

Definition:

```
a = 1
c = "the cat"
c = 3.5
```

Usage:

```
print c
a = a + 1
```

- One instruction → one line (no ; at the end)
- Variable → name for a value
- No type declaration: dynamically dertermined by the interpreter during execution

Garbage Collection

```python
a = 3
# ...
a = "hello"
```

- When no more references point to a value, the corresponding memory is automatically released
- You don't have to handle memory allocation

Errors using variables

- Using uninitialized variables

```
>>> planet = "pluton"
>>> print plant
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name "plant" is not defined
```

- Type errors

```
>>> p = "coucou"
>>> q = 3
>>> print p + q
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: cannot concatenate "str" and "int" objects
```

Numbers

```
a = 1  # Integer
b = 3.5  # Float
c = (a - 5) / (a + 1.0)
d = c ** 2  # Same as c * c
```

Warning:

```
>>> 3 / 4
0
```

Math module

A module adds new functions and types.

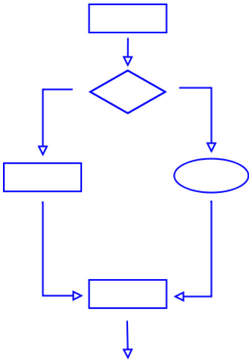
For common math functions, you should use the *math* module

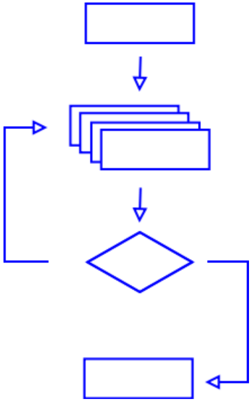
```
import math
x = math.sin(1 + math.e)
y = math.cos(2 * math.pi)

from math import sqrt
norm = sqrt(x ** 2 + y ** 2)
```

Part II

Control flow

Control flow





Conditionals

```python
num = 0.5
if num < 0:
    print "negative"
elif num > 0:
    print "positive"
else:
    print "null"
```

- A block is delimited by
  - A colon ':'
  - Indentation
- Indentation should be always and only 4 spaces
- Note that indentation is important in Python

Loops

```
num_moons = 3
while num_moons > 0:
    print num_moons
    num_moons = num_moons - 1
print "end"
```

- Note that indentation is very important in Python

Blocs interweaving

Print all odd numbers between 0 and 9:

```python
num = 0
while num <= 10:
    if (num % 2) == 1:
        print num
    num += 1
```

- Note that indentation is really important in Python

Part III

Functions

Functions

```
>>> def welcome(name, end=";"):
        print "Welcome", name, end

>>> def get_my_name():
        return "Nicolas"

>>> welcome(get_my_name())
Welcome Nicolas ;

>>> welcome("Alexandre", end=":-)")
Welcome Alexandre :-)
```

- Always try to write many small clear functions
- Note that indentation is really, really important in Python

Part IV

Collections

Strings

```
>>> a = "hello"
>>> b = "everybody"
>>> a.capitalize() + " " + b
"Hello everybody"
```

- string = variable + associated methods (string = object)
- `my_string.my_method()` creates a new string from `my_string`

Need help ?

```
help(a.capitalize)
dir(a)  # list all available methods
```

Split method !

```
>>> a = " hello all   my friends   "
>>> a.split()
["hello", "all", "my", "friend"]

>>> b = "17,3,7,19"
>>> a.split(",")
["17", "3", "7", "19"]
```

- Split a string into a list of substrings
- Delimiter → parameter, whitespace$\underline{\text{s}}$ by default

Formating

Build a string to be formated:

```python
a = "Hello {0}. Pi is {1:.4f}. I am {2:.2%} sure."
```

- Elements {...} are remplaced by provided values
- Whole formating language (number of floating points, ...)
- Instanciation:

```python
import math
print a.format("Nicolas", math.pi, 0.98)
```

- Will output

```
Hello Nicolas. Pi is 3.1416. I am 98.00% sure.
```

Lists

- In Python we have lists (for both lists and arrays)
- Ordered collection of elements
- Elements may have different types

Syntax:

- Create a new list

```
>>> lst = []  # empty list
>>> lst = [1, "hello", [4], 2.1]
```

- Range function

```
>>> range(5)
[0, 1, 2, 3, 4]
>>> range(6, 0, -2)
[6, 4, 2]
```

List operations I

- Append new elements

```
>>> lst = [1, 4, 2, 5]
>>> lst.append(0)
>>> print lst
[1, 4, 2, 5, 0]
```

- Length of a list

```
>>> len(lst)
5
```

- Common operation on numeric lists

```
>>> sorted(lst)
[0, 1, 2, 3, 5]
>>> sum(lst), min(lst), max(lst)
(11, 0, 5)
```

- Iterate over values

List operations II

```
>>> for el in lst:
        print el,
1 4 2 5 0
```

- Test if list contains an element

```
>>> 3 in lst:
False
```

Warning Operation in $\mathcal{O}(n)$

- Indexing

```
>>> lst[0]
1
>>> lst[1:4]
[4, 2, 5]
```

Your turn

Exercice

Write two function that compute the mean and the variance of a list

$$m = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad s = \frac{1}{n}\sum_{i=1}^{n}(x_i - m)^2$$

```
>>> lst = [1, 5, 8, 7, 2]
>>> mean(lst)
4.6
>>> var(lst)
7.44
```

Solution

```python
def mean(lst):
    m = 0.0  # Why not m = 0 ?
    for el in lst:
        m += el
    return m / len(lst)

# Shorthand
def mean(lst):
    return 1.0 * sum(lst) / len(lst)

def var(lst):
    s = 0.0
    m = mean(lst)  # Why now and not in the loop ?
    for el in lst:
        s += (el - m) ** 2
    return s / len(lst)
```

Sets

- Unordered collection of (hashable) elements

```
>>> s = set()  # empty set
>>> s.add(3); s.add(2); s.add(1)
>>> s
set([1, 2, 3])
```

- Similar operation as for lists

```
>>> 2 in s  # Efficient: O(1)
True
>>> for el in s:  # Order is execution dependent
        print el
1 3 2
```

Dictionaries I

- Unordered set of pairs (key, value)
- One unique value is associated with a key
- Keys and value can be (almost) anything

```
>>> empty_dict = {}

>>> birthdays = {"Newton": 1642, "Darwin": 1809}
>>> birthdays[1782] = 1782
>>> print birthdays
{"Newton": 1642, 1782: 1782, "Darwin": 1809}

>>> birthdays["Darwin"] = 2014  # Associations
>>> print birthdays["Darwin"]   # are unique
2014
```

Dictionaries II

- Check if a key is present

```
>>> "Newton" in birthdays  # Efficient
True
>>> "Einstein" in birthdays.keys()  # Inefficient
False
```

- Iterate through a dictionary

```
>>> for key in birthdays:
...     print ("{0} : {1}"
                ".format(key, birthdays[key]))
Darwin : 2014
1792 : 1792
Newton : 1642
```

Exercice

- We have a list the authors for each book in our library
- Write a function that count how many books they wrote

Input:

```
authors = ["Sartre", "Camus", "Bourdieu",
           "Sartre", "Sartre"]
```

Outpu

```
Camus wrote 1 book
Bourdieu wrote 1 book
Sartre wrote 3 books
```

Solution

```python
def count_books(authors):

    counts = dict()
    for aut in authors:
        if aut in counts:
            counts[aut] = counts[aut] + 1
        else:
            counts[aut] = 1

    # Output
    for aut in counts:
        cnt = counts[aut]
        if cnt == 1:
            print "{0} wrote 1 book".format(aut)
        else:
            print "{0} wrote {1} books".format(aut, cnt)

```

Part V

Files

Read a file

- Routine operation: read a file line by line

```python
import codecs

count = 0
with codecs.open("file_name.txt", "rt",
                 encoding="utf8") as my_file:
    for line in my_file:
    line = line.strip()  # remove '\n'
    # some processing
    count += 1
print count
```

Exercices

1. Write a program that counts the number of words in a file.
2. Write a program that counts the number of *different* words in a file.

3. Write a program that for each line in a file prints, the longest word in that line.
4. Write a program that output all the characters used in a file, sorted by frequency counts.

Word count: solution

```python
import codecs
import sys

filename = sys.argv[1]

n_words = 0
with codecs.open(filename, "rt",
                 encoding="utf-8") as text:
    for line in text:
        n_words += len(line.strip().split())

print ("There are {0} words in file {1}"
       "".format(n_words, filename))
```

Different words count: solution

```python
import codecs
import sys

filename = sys.argv[1]

voc = set()
with codecs.open(filename, "rt",
                 encoding="utf-8") as text:
    for line in text:
        for word in line.strip().split():
            voc.add(word)

print ("There are {0} different words in file {1}"
       "".format(len(voc), filename))
```

Longest word: solution

```
import codecs
import sys

with codecs.open(sys.argv[1], "rt",
                 encoding="utf-8") as file_:
    for line in file_:
        words = line.strip().split()
        if len(words) == 0:
            print
        else:
            longest_word = words[0]
            for word in words[1:]:
                if len(word) > len(longest_word):
                    longest_word = word
            print longest_word
```

Most frequent character: solution

```python
import codecs
import sys

char_count = dict()
with codecs.open(sys.argv[1], "rt",
                 encoding="utf-8") as file_:
    for char in file_.read():
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1

char_max = max(char_count, key=char_count.get)

print ("Character '{}' is the most "
       "frequent one with {} occurences."
       "".format(char_max, char_count[char_max]))
```