Python Tutorial

Nicolas Pécheux
nicolas.pecheux@limsi.fr

LIMSI/CNRS — Univ. Paris Sud

March 21, 2015

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 $\Rightarrow$ 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:

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

Usage:

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

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

Garbage Collection

```
1        a = 3
2        # ...
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

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

Warning:

```
1       >>> 3 / 4
2       0
```

Math module

A module adds new functions and types.

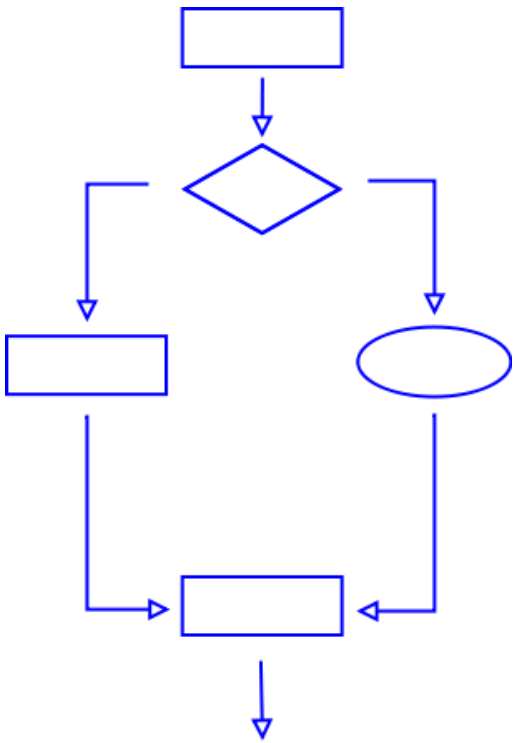
For common math functions, you should use the *math* module

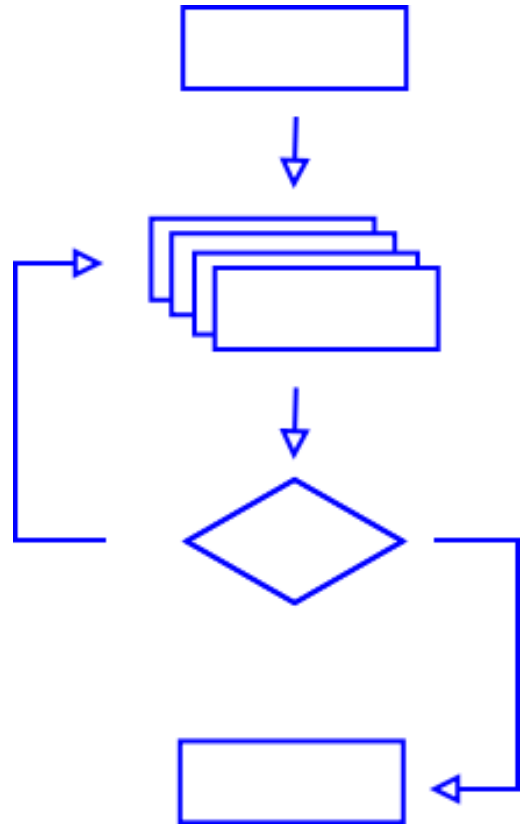
```
1       import math
2       x = math.sin(1 + math.e)
3       y = math.cos(2 * math.pi)
4
5       from math import sqrt
6       norm = sqrt(x ** 2 + y ** 2)
```

Part II

Control flow

Control flow





Conditionals

```
1        num = 0.5
2        if num < 0:
3            print "negative"
4        elif num > 0:
5            print "positive"
6        else:
7            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

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

- Note that indentation is very important in Python

Blocs interweaving

Print all odd numbers between 0 and 9:

```
1        num = 0
2        while num <= 10:
3            if (num % 2) == 1:
4                print num
5            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 !

```
1       >>> a = " hello all   my friends  "
2       >>> a.split()
3       ["hello", "all", "my", "friend"]
4
5       >>> b = "17,3,7,19"
6       >>> a.split(",")
7       ["17", "3", "7", "19"]
```

- Split a string into a list of substrings
- Delimiter $\rightarrow$ parameter, whitespaces by default

Formating

Build a string to be formated:

```
1       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:

```
1       import math
2       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)
```

List operations II

- Iterate over values

```
>>> 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

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

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

Solution

```
1  def mean(lst):
2      m = 0.0  # Why not m = 0 ?
3      for el in lst:
4          m += el
5      return m / len(lst)
6
7  # Shorthand
8  def mean(lst):
9      return 1.0 * sum(lst) / len(lst)
10
11 def var(lst):
12     s = 0.0
13     m = mean(lst)  # Why now and not in the loop ?
14     for el in lst:
15         s += (el - m) ** 2
16     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"]
```

Output:

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

Solution

```
1  def count_books(authors):
2
3      counts = dict()
4      for aut in authors:
5          if aut in counts:
6              counts[aut] = counts[aut] + 1
7          else:
8              counts[aut] = 1
9
10     # Output
11     for aut in counts:
12         cnt = counts[aut]
13         if cnt == 1:
14             print "{0} wrote 1 book".format(aut)
15         else:
16             print "{0} wrote {1} books".format(aut, cnt)
17
```

Part V

Files

Read a file

- Routine operation: read a file line by line

```
1   import codecs
2
3   count = 0
4   with codecs.open("file_name.txt", "rt",
5                    encoding="utf8") as my_file:
6       for line in my_file:
7       line = line.strip()  # remove '\n'
8       # some processing
9       count += 1
10  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

```
1  import codecs
2  import sys
3
4  filename = sys.argv[1]
5
6  n_words = 0
7  with codecs.open(filename, "rt",
8                   encoding="utf-8") as text:
9      for line in text:
10         n_words += len(line.strip().split())
11
12 print ("There are {0} words in file {1}"
13        "".format(n_words, filename))
```

Different words count: solution

```
1  import codecs
2  import sys
3
4  filename = sys.argv[1]
5
6  voc = set()
7  with codecs.open(filename, "rt",
8                   encoding="utf-8") as text:
9      for line in text:
10         for word in line.strip().split():
11             voc.add(word)
12
13 print ("There are {0} different words in file {1}"
14        "".format(len(voc), filename))
```

Longest word: solution

```
1  import codecs
2  import sys
3
4  with codecs.open(sys.argv[1], "rt",
5                   encoding="utf-8") as file_:
6      for line in file_:
7          words = line.strip().split()
8          if len(words) == 0:
9              print
10         else:
11             longest_word = words[0]
12             for word in words[1:]:
13                 if len(word) > len(longest_word):
14                     longest_word = word
15             print longest_word
```

Most frequent character: solution

```
1  import codecs
2  import sys
3
4  char_count = dict()
5  with codecs.open(sys.argv[1], "rt",
6                   encoding="utf-8") as file_:
7      for char in file_.read():
8          if char in char_count:
9              char_count[char] += 1
10         else:
11             char_count[char] = 1
12
13 char_max = max(char_count, key=char_count.get)
14
15 print ("Character '{}' is the most "
16        "frequent one with {} occurences."
17        "".format(char_max, char_count[char_max]))
```