Introduction to programming with Python

Ing. Lelio Campanile
Main Goal

- Introduce you to programming
- introduce you to the most essential feature of python programming
Before to start

The name "Python" does not refer to the snake, but it comes from "(Monty) Python"
Why Python?

- Software Quality:
  - Python code is designed to be readable
  - by design python implements readable syntax
Why Python?

- Productivity:
  - Python code is typically one-third the size of equivalent C++ or Java code
Why Python?

- Program portability:
  - Python programs run on all major computer platform
  - Support for portable GUI, DB, web system
Why Python?

- Support Library:
  - Python comes with an extensive collection of libraries, The Standard Library
  - A lot of third-party projects and libraries
Why Python?

Enjoyment

let’s have fun! Programming in python is fun!
Is Python a Scripting language?
NO! But you can use it to write a script
Is Python a Scripting language?

- sometimes applied in scripting roles

- General Purpose programming language that blends procedural, functional and object-oriented paradigms
Who does use python?
Python main features

- succinct yet readable syntax
- Great introspection
- Great standard library
  - Built-in data structures
  - Language *battery included*
- Interactive shell
Python is an Interpreted language
Compiled vs Interpreted Language

<table>
<thead>
<tr>
<th>Compiled:</th>
<th>Interpreted:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td><strong>Pros</strong></td>
</tr>
<tr>
<td>- Faster execution</td>
<td>- Steep learning curve</td>
</tr>
<tr>
<td>- Can <strong>produce</strong> a distributable <strong>executable</strong> standalone files</td>
<td>- Takes automatically care of <strong>memory usage</strong></td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td><strong>Cons</strong></td>
</tr>
<tr>
<td>- More <strong>complicated</strong> to build (many files)</td>
<td>- Usually <strong>slower</strong></td>
</tr>
<tr>
<td>- User has to administrate <strong>Memory usage</strong></td>
<td>- Does not produce standalone programs</td>
</tr>
</tbody>
</table>
Install python which version?

- nowadays python has 2 different branch version 2.x and version 3.x

- We’ll use the 3.x version

- the python 2.x branch ends support in the next year
Install python

- official python distribution

- Anaconda distribution
Anaconda python distribution

- Anaconda is a python distribution by Continuum Analytics.

- Anaconda is a completely free enterprise-ready Python distribution for large-scale data processing, predictive analytics, and scientific computing.

- Apart from that, Anaconda ships with easy-to-use installers for almost every platform, that would drastically reduce the burden of setting up the environment (exp. on Windows)
Get Anaconda

https://www.continuum.io/downloads
Open the terminal and type `python3`
Python Shell

- python shell is useful for testing
- executes immediately the commands that you type
- it doesn't save the code
IDLE Editor

- default editor installed with python
- simple and efficient
For a more serious job

- Atom (open source)

- Pycharm (free and pro)
save your file with a .py extension
execute it with the command:

python3 filename.py
Variables

a variable holds a value

message = "Hello python world"
print(message)
Exercise

write your first program:

store and print your own version of "Hello world"
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.

>>> message = "Hello python world"

>>> print(message)
Hello python world

>>>
Exercise

Store a message in a variable, and then print that message.

Store a new message in the same variable, and then print that new message.
you can change the value of a variable at any point

message = "Hello python world"
print(message)

message = "Hello python"
print(message)
>>> message = "Hello python world"
>>> print(message)
Hello python world

>>> message = "Hello python"
>>> print(message)
Hello python

Naming Rules

- Variable can only contain letters, numbers, underscores
- Variable names can start with a letter or an underscore, but not with a number
- Spaces are not allowed in variable names
- We use underscores characters for that
- You cannot use Python Keywords as names
- Variable names should be descriptive
- Can contain Unicode Literals
  \[-\pi = 3.141592653589793\]
your first error: Name Error

```python
Python 3.6.2 (v3.6.2:5fd33b5926, Jul 16 2017, 20:11:06)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> message = "Hello python world"

>>> print(message)
Hello python world

>>> message = "Hello python"
>>> print(message)
Hello python

>>> print(message)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    print(message)
NameError: name 'message' is not defined

>>> 
```
What are we missing?

```python
message = "Hello python world"
print(message)

message = "Hello python"
print(message)
```

??Type declaration??
Python Typing Mechanism

Dynamic Typing
- It is not required to specify the types of variables/functions
- It is automatically inferred by operations

Strong Typing
- Once the type has been inferred, it cannot change without explicit CAST
Python Typing Mechanism

Duck typing

If it walks like a Duck, it quacks like a Duck

then

It's a Duck (inference)
Numeric Type

**Integer:** 345, 1, -32

**Float:** 1.2, 1.32, 19e5, 12.1e-6

**boolean:** True, False
You can use parenthesis to modify the standard order of operation

```python
standard_order = 4+5*2
print(standard_order)
14

my_order = (4+5)*4
print(my_order)
18
```
Operations

- + adition
- - subtraction
- * multiplication
- / division
Exercise

\[
\begin{align*}
    a &= 4 \\
    b &= 2 \\
    c &= 1.5 \\
    a / b &\quad a + b & c / b & a / b
\end{align*}
\]
```python
>>> a=4
>>> b=2
>>> c=1.5
>>> a/b
2.0
>>> a+b
6
>>> c/b
0.75
>>> a/b
2.0
```
how to check the type?

```
a = 4
b = 2
c = a / b
print(c)
2.0
type(c)
<class 'float'>
type(a)
<class 'int'>
type(b)
<class 'int'>
```
When the operands are of different type, the python interpreter before converts them into the most complex type, and then it executes the calculation.
Integer operator

```python
a = 4
b = 2
c = a // b
print(c)
2
type(c)
<class 'int'>
```
other operations

- ** pow operator

- % modulus (remainder operator)
Exercise

Given 3 numbers (12, 32, 2.0) write a program that print:
- the sum
- the multiplication
Exercise

- Given the base = 10 and the height = 15 of a triangle, write a program that prints the area.

- Given 2 dates (2019-02-28, 2019-04-20) write a program that prints the days between them

- Given the radius = 5 of a circle, write a program that prints the area and the circumference
Booleans and Logical Tests

- True and False are Python keywords
- bool type in Python corresponds to int
- True = 1
- False = 0
Logical operators

- == equality
- != inequality
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- *in* test if an item is in a list
Equality

5 == 5
True

3 == 5
False

5 == 5.0
True

'lelio' == 'lelio'
True

'lelio' == Lelio'
False
Inequality

Two items are unequal if they do not have the same value

'alpha' != 'beta'
True

1.0 != 1
False

1 != 2
True
### Other Logical Tests

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10 &gt; 3$</td>
<td>True</td>
</tr>
<tr>
<td>$10 \geq 10$</td>
<td>True</td>
</tr>
<tr>
<td>$10 \geq 3$</td>
<td>True</td>
</tr>
<tr>
<td>$10 \geq 11$</td>
<td>False</td>
</tr>
<tr>
<td>$10 &lt; 3$</td>
<td>False</td>
</tr>
<tr>
<td>$3 &lt; 10$</td>
<td>True</td>
</tr>
<tr>
<td>$3 \leq 10$</td>
<td>True</td>
</tr>
<tr>
<td>$10 \leq 10$</td>
<td>True</td>
</tr>
</tbody>
</table>
Logical operators

**and** **not** **or**

Use them to write more complex logical tests

```python
a == b and a != b
False

a == b or not a == b
True
```
True and False python's definition

- False: False (bool), None, 0, empty object

- True: everything else (number > 0, a not empty object or string, etc.. )
Python operator precedence

- *  %  /  //

- +  -

- <  <=  >  >=  ==  !=

- not

- and

- or

To force the precedence order, you can use the parentheses
To Comment your code is a good habit!

Make sure that you add a comment to your code when:
- you want to remember why you write your code in that manner
- when there more than one way to solve a problem
write comments short and clear!
Strings

Strings are sets of characters

To define a string you can use single or double quotes

```python
string = "this is a string"
string = 'this is a string too'
```

You can combine single and double when you have a string that contains a quotation:
python

quote = "Einstein once said: 'Try not to become a man of success, but rather try to become a man of value.'"
String Combination

first_name = "Lelio"
last_name = "Campanile"

full_name = first_name + " " + last_name
print(full_name)
Lelio Campanile
String Combination

```python
string = full_name + " is my full name"
print(string)
Lelio Campanile is my full name

string_multiplication = "=" * 20
print(string_multiplication)
```

============
Exercise

- Find a quote that you like. Store the quote in a variable, with an appropriate introduction such as "Ken Thompson once said, 'One of my most productive days was throwing away 1000 lines of code'". Print the quote.

- Store your first name and last name in separate variables, and then combine them to print out your full name.

- Use concatenation to make a sentence about you, and store that sentence in a variable.
Exercise

- Store the results of at least 5 different calculations in separate variables. Make sure you use each operation at least once.

- Print a series of informative statements, such as "The result of the calculation 5+7 is 12."
a = 1
b = 1

str_a = str(a)

print("this is a cast " + str(a))
this is a cast 1

print("this is a cast " + str_a)
this is a cast 1