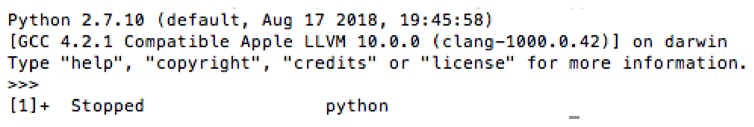
Introduction to programming, Lesson 1

Python is a very simple language, and has a very straightforward syntax. There are two major Python versions, Python 2 and Python 3. Python 2 and 3 are quite different. In this course, we will use Python 2, because this is the default version of Python for ENS computers.

To check which version of Python you have, open the terminal command line.

> **In class:** Double-click the Terminal icon on the desktop.
> **Mac:** The Terminal app is in the Utilities folder in Applications. To open it, open your Applications folder, then open Utilities and double-click on Terminal.
> **Windows:** Click start. In the search or run line, type cmd and press enter.

In the terminal command line, type python. You will see something like this:



```
Python 2.7.10 (default, Aug 17 2018, 19:45:58)
[GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
[1]+  Stopped                 python
```

If the first line says Python 2..., it is Python 2. If you use your personal computer and you have Python 3, you will have to make adjustments to the code. Type control + z and press enter.

Hello, world!

Our first program will print out "Hello, world".

1. Open the terminal command line.
2. Create a new directory: type `mkdir 'initiation_a_la_programmation'` in the terminal, press enter
3. Open the new folder: type `cd 'initiation_a_la_programmation"` in the terminal, press enter
4. Each folder has it address ("path"). You can find the path to the folder you are currently in as follows: type `pwd` in the terminal, press enter. Memorize this path. Do not close the terminal.
5. Open a text editor.
    - **In class:** Double-click on Gedit editor icon on the desktop.
    - **Mac & Windows:** Download a free text editor Sublime text 2.
6. Create a new file. Copy the line below to the file, and save it as helloworld.py in the folder that you created in step 2 (use the path to find it).

   ```
   print "Hello, world!"
   ```
7. Go back to the terminal. You must be still in the folder that you created in step 2. Type `python helloworld.py`

Terminal: more useful tricks

cd .. – Leave the current folder and go to the parent folder
ls – Show all files and folders in the current folder
tab key – Instead of typing the full name of a folder / file, you can type several first letters and press tab, the terminal will autocomplete the name for you.
python – if you want to test something quickly, without saving into a file, you can type `python` in the terminal and press enter. You can then type in your code line by line (press enter after each line). To exit this

mode, press control + z.

Variables and Types

Python is completely object oriented, and not "statically typed". You do not need to declare variables before using them, or declare their type. Every variable in Python is an object.

Numbers. Python supports two types of numbers - integers and floating point numbers. (It also supports complex numbers, which will not be explained in this tutorial). To define an integer, use the following syntax:

```
myint = 13
print myint
```

To define a floating point number, you may use the following notation:

```
myfloat = 7.0
print myfloat
```

Strings. Strings are defined either with a single quote or a double quotes.

```
mystring = 'hello'
print mystring
mystring = "hello"
print mystring
```

The difference between the two is that using double quotes makes it easy to include apostrophes or quotes (whereas these would terminate the string if using single quotes).

```
mystring = "Don't worry about apostrophes"
print mystring
mystring = 'She said, "Apostrophes is not a problem"'
print mystring
```

Exercise 1. Does this script print out “She said, "Hello world"”? Change it so it does.

```
prin("She said, "Hello world"")
```

Exercise 2. Create a string, an integer, and a floating point number. The string should be named mystring and should contain the word "hello". The floating point number should be named myfloat and should contain the number 100.7, and the integer should be named myint and should contain the number 8.

```
# change this code
mystring = None
myfloat = None
myint = None
print mystring, myfloat, myint
```

Basic operators

Python has all the arithmetic operators.

Introduction to programming, Lesson 1

```
number = 1 + 2 * 3 / 4.0
print number
remainder = 11 % 3
print remainder
squared = 7 ** 2
cubed = 2 ** 3
print squared
print cubed
```

Also, Python has many operators for strings. Here are some of them:

```
astring = "Hello" + " " + "world!" # creates one string “Hello world!”
print len(astring) # prints the length of the string, including punctuation signs and spaces
print astring.index("o") # prints the position of the first letter "o"
print astring.count("l") # prints the number of letters "l" in astring
print astring[3:7] # prints a substring of astring that starts at index 3 and ends at index 7
print astring.startswith("Hello") # prints True if astring starts with "Hello"
print astring.endswith("end") # prints True if astring ends with "end"
afewwords = astring.split(" ") # splits astring into a list of strings using space as a separator
```

Lists

Lists are very similar to tables. They can contain any type of variable, and they can contain as many variables as you wish. Lists can also be iterated over in a very simple manner. NB! Python uses indentation for blocks. The standard indentation requires standard Python code to use four spaces.

```
numbers = []
numbers.append(1)
numbers.append(2)
print mylist[0] # prints 1
print mylist[1] # prints 2

# prints out 1,2
print mylist

letters = ['a', 'b', 'c']
for x in letters:
    # indented four spaces
    print x
```

Accessing an index which does not exist generates an exception (an error).

```
mylist = [1,2,3]

print mylist[10]
```

Lists can be joined with the addition operation:

```
even_numbers = [2,4,6,8]
odd_numbers = [1,3,5,7]
all_numbers = odd_numbers + even_numbers
print(all_numbers)
```

Introduction to programming, Lesson 1

Exercise 3. In this exercise, you will need to add numbers and strings to the correct lists using the "append" list method. You must add the numbers 1, 2, and 3 to the "numbers" list, and the words 'hello' and 'world' to the strings variable.

You will also have to fill in the variable second_name with the second name in the names list, using the brackets operator []. Note that the index is zero-based, so if you want to access the second item in the list, its index will be 1.

```
numbers = []
strings = []
names = ["John", "Eric", "Jessica"]

# write your code here
second_name = None

# this code should write out the filled arrays and the second name in the names list (Eric).
print numbers
print strings
print second_name
```

String formatting

If you want to print something fancy...

```
# This prints out "Hello, John!"
name = "John"
print "Hello, %s!" % name
```

To use two or more argument specifiers, use a tuple (parentheses):

```
# This prints out "John is 23 years old."
name = "John"
age = 23
print "%s is %d years old." % (name, age)
```

Any object which is not a string can be formatted using the %s operator as well. For example:

```
# This prints out: A list: [1, 2, 3]
mylist = [1,2,3]
print "A list: %s" % mylist
```

Here are some basic argument specifiers you should know:

%s - String (or any object with a string representation, like numbers)

%d - Integers

%f - Floating point numbers

%.<number of digits>f - Floating point numbers with a fixed amount of digits to the right of the dot.

Exercise 4. You will need to write a format string which prints out the data using the following syntax: Hello John Doe. Your current balance is $53.44.