Table of contents (../toc.ipynb)

Yet another Python course?

As said, Python is so popular that there are plenty of online courses, tutorials and books available. Please find here a short list of free and great Python learning sources

- A free online tutorial for Python beginners on tutorialspoint.com (https://www.tutorialspoint.com/python/index.htm)
- Book: A Whirlwind Tour of Python (https://jakevdp.github.io/WhirlwindTourOfPython/) [VanderPlas2016] (../references.bib)
- IPython cloud service Python anywhere (https://www.pythonanywhere.com/try-ipython/)
- Interactive set of Python tutorials on learnpython.org (https://www.learnpython.org/)
- Python data science cheat sheets on datacamp.com (https://www.datacamp.com/community/data-science-cheatsheets) [PyCheatSheats] (../references.bib)
- Official Python Documentation on docs.python.org (https://docs.python.org/3/tutorial/index.html)

There is much more!

There is also a bunch of commercial tutorials available to learn Python, e.g. on

- Datacamp.org (https://www.datacamp.com/courses/intro-to-python-for-data-science)
- Udacity.com (https://www.udacity.com/course/introduction-to-python--ud1110)
- Cousera.org (https://www.coursera.org/learn/python)
- ...

and some of these Python introduction courses are free as well.

The following sections on Python syntax and semantics are partly adapted from *A Whirlwind Tour of Python* [VanderPlas2016] (../references.bib), which is under CC0 license.

Python syntax and semantics reference

Please find overall syntax and semantics reference in The Python Language Reference (https://docs.python.org/3/reference/index.html) [PyReference] (../references.bib), and all functionality of Python in the The Python Standard Library reference (https://docs.python.org/3/library/index.html#library-index) [PyStandardLib] (../references.bib).

However, we will briefly review the syntax herein to get you started.

The syntax is the structure of a language, or how to write it. The semantics is the meaning of the language or how it is interpreted.

Python syntax

Python was designed for high readability. Hence, parenthesis, semicolons and the like are rarely used and keywords are preferably easy to read English words. Therefore, Python code looks often like pseudo code.

Keywords

Python has reserved words which cannot be used as variable or identifier names (names of functions or classes).

```
 and as assert async await break class continue def del elif else
except False finally for from global if import in is lambda None
nonlocal not or pass raise return True try while with yield
```

Easy to read

Let us compare an example of a for loop in C and Python.

```
/* for loop execution in C*/
for( a = 10; a < 20; a = a + 1 ){
   printf("value of a: %d\n", a);
}

# same for loop in Python
for a in range(10, 20):
    print("value of a", a)
```

- Both code blocks use indentation to make the code easy to read. However, in C, the indentation is good programming style and not required by the compiler. In C, the code block is encapsulated with curly braces `{}`. In contrast, Python requires that code blocks are indented.
- Python simply uses end of line (carriage return). C requires a semicolon.
- Python uses fewer special symbols (braces, semicolons,...).
- The `a in range(10, 20)` is better human readable or pseudo code style than `( a = 10; a < 20; a = a + 1 )`.

In [65]:
```python
# Now, lets run the Python loop
for a in range(10, 20):
    print("value of a:", a)
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Indentation

Hence, indentation matters! Code blocks are usually started with colon `:` and indented by (commonly) four spaces. However, you are free to use any consistent indentation.

The two versions of the for loop produce different results because of indentation. The second `print(y)` is not indented and hence called after the loop.

```
In [66]: y = 0
         for i in range(0, 3):
             y += i
             print(y)
```

```
0
1
3
```

```
In [67]: y = 0
         for i in range(0, 3):
             y += i
         print(y)
```

```
3
```

Comments

In-line comments are marked with a hash `#`. Multi-line comments are not supported by Python but you can use triple quoted strings instead `"""`, or `'''`

In [68]:
```
""" Here a comment
with line break."""

''' Here another comment
over two lines.'''

some_var = 1
# and here a comment in one line
some_other_var = 2  # and here another comment
```

Line continuation

If your code is too long, put a `\` as line wrap. Or encapsulate it in braces with indentation.

In [69]:
```
# Here some long equation
complex = 1 + 4 - 8 +\
    2 - 7
```

In [70]:
```
# Or the alternative version with braces
complex = (1 + 4 - 8 +
           2 - 7)
```

Multiple statements

You can either enter one statement after the other or use a semicolon to add them in one line. However, multiple statements in one line are discouraged by Python style guides.

```
In [71]: a = [1, 2, 3]; b = "my string"

         # is the same as
         a = [1, 2, 3]
         b = "my string"
```

Whitespace

As said, whitespace is used for indentation. Within lines, whitespace does not make a difference.

In [2]:
```
my_var = 1 + 3
# this is interpreted in the same way
my_var = 1       + 3

print(my_var)
```

```
4
```

Exercise: Try syntax (10 minutes)

Please try out what you have seen by yourself.

To solve this task you have (at least) three options to run you code:

- Write a Python file and call it through ipython with `%run yourfile.py`.
- Use the interactive Jupyter Notebooks of this course on Binder (https://mybinder.org/v2/gh/StephanRhode/py-algorithms-4-automotive-engineering/master).
- Use IPython cloud service Python anywhere (https://www.pythonanywhere.com/try-ipython/) Note that you need the magic command `%cpaste` to be able to type multiple lines in IPython.

Here the task:

- Write a for loop which adds the loop index to a defined constant.
- Try to include one line comments and multi line comments.
- Use `\` or `()` for line continuation.
- Play with indentation (also try to add wrong indentation) and whitespace.
- Add the print command.

Solution

Please find one possible solution in `solution_syntax.py` (solution_syntax.py) file.

```
In [1]: %run solution_syntax.py
```

```
My loop variable  3
My loop variable  4
My loop variable  5
My loop variable  6
My loop variable  7
My loop variable  8
My loop variable  9
My loop variable  10
My loop variable  11
```