For the first part of this tutorial, we will use the Python interpreter. We will use **Python 2.7** throughout the course. Open a terminal and type 'python' to get the Python interpreter. It should look similar to this:

```
wtmpc113$ python
Python 2.7.12 (default, Dec  4 2017, 14:50:18)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

PART I - Basic commands

1) Numerical values: simple manipulation, operator precedence, and boolean logic.
Try out the following commands and understand the respective output to get familiar how numerical operations are processed.

- `7 / 2`
- `7 / 2.0`
- `7.0 / 2`
- `7.0 / 2.0`
- `7 % 2`
- `16**2`
- `2 + 3 * 4`
- `(2+3) * 4`
- `2**2 * 2+2`
- `(2**2) * (2 + 2)`
- `1 + 2 + 3 * 4 + 5`
- `(1 + 2 + 3) *(4 + 5)`
- `x = 1 ___ >>> x+=2  ___ >>> x`
- `y = [1, 2, 3] ___ >>> y+= 11, 22, 33 ___ >>> y`
- `2 < 3 or 5 < 4`
- `2 < 3 and 5 < 4`
- `2 < 3 and 3 > 1`
- `2 < 3 and not 3 > 1`
- `type(42)`
- `type(4.2)`
- `type("42")`

2) Variable assignments: What will be the output of the following programs?

```
x = 4
y = x + 1
x = 2
print x, y
```

=> 2 5

```
x, y = 2, 6
x, y = y, x + 2
print x, y
```

=> 2 8

```
a, b = 2, 3
c, b = a, c + 1
print a, b, c
```

=> ERROR: c is not defined

```
x = 4
y = 5
p = x < y or x < z
print p
```

=> true

3) Strings:
Try out the following commands to get familiar how strings are used and manipulated.

- `type("Hello, World!")`
- `len("helloworld")`
- `str(123)`
- `x = "hello"`
  `print x.upper()`
- `"python" > "java"`

4) Lists: Try out the following commands to get familiar how lists can be used and manipulated.

- `["hello", "world"]`
- `[0, 1.5, "hello"]`
- `a = [1,  2, 3, 4]`
- `len(a)`
- `b = [5, 6, 7, 8]`
- `b*2`
- `zip(a,b)`
- `zip(["a", "b", "c"], [1, 2, 3])`
- `c=a+b`
- `c[1]`
- `c[0:2]`
- `c[:2]`

- `c[1:3]`
- `c[-1]`
- `c[-2:]`
- `c[1]=1`
- `c.append(9)`
- `d=[c, 9, 10]`
- `range(10)`
- `range(2,12)`
- `range(2, 12, 3)`
- `x=range(10)`
  `x[::-1]`
- `l = [2, 10, 42, 3, 37]`
  `l.sort()`
- `l = [2, 10, 42, 3, 37]`
  `sorted(l)`

5) Dictionaries: They are similar to lists but unordered and accessible also by noninteger keys.

- `a = {'x': 1, 'y': 2, 'z': 3}`
- `a['x']`
- `a.keys()`
- `a.values()`
- `a.items()`

PART II - The `numpy` array

It is common practice to use `numpy` arrays for matrix manipulations and operations as e.g. for neural networks. To get familiarized with indexing or slicing, work out the following using `numpy`:

- Suppose you wish to generate a 3x1 vector that contains the number 5 in every position. Which of the following expressions will accomplish this task? (hint: you can also type `import numpy as np` before in your terminal and substitute the term `numpy` accordingly.)

  1. `numpy.eye(3)*5`

  2. `numpy.identity(3)*5`

  3. [✓ `numpy.ones(3)*5`]

  4. `numpy.ones(3,1)*5`

  5. `numpy.ones((3,1))*5`

- Suppose you wish to generate a 2x3 matrix that contains only zeros. Which of the following expressions will achieve this goal?

  1. `numpy.zeros(3)`
  2. `numpy.zeros((2,3))`
  3. `numpy.zeros(2,3)`
  4. [✓ `numpy.zeros(3,2)`]
  5. `[0 0 0; 0 0 0]`
  6. `[[0, 0, 0], [0, 0, 0]]`

- Create a numpy array as shown below (or create your own) and learn about array manipulations by using the following commands:

```
a = numpy.array([[2, 3.2, 5.5, -6.4, -2.2, 2.4],
             [1, 22, 4, 0.1, 5.3, -9],
              [3, 1, 2.1, 21, 1.1, -2]])
```

  - `print a[1]`
  - `print a[1:4]`
  - `print a[:, 4]`
  - `print a[1:, 2]`

- For the creation of arrays and extraction of specific values, you can also use specific built-in functions. We start with some basics but you will learn more about different functions in later tutorials. Make sure you understand the output and the different ways of calling functions.

```
array = numpy.array([range(6), range(10, 16)])
```

  - `print array.shape`
  - `print array.size`
  - `print array.max()`
  - `print array.min()`
  - `print numpy.reshape(array, (2, 6))`
  - `print numpy.reshape(array, (3,2,2))`

- `print numpy.transpose(array)`

-
  ```
  colors = ["red", "green", "blue", "purple"]
  for i in range(len(colors)):
  🟥print(colors[i])
  ```

PART III - Python functions

Now, write your own little Python functions. You can use this given piece of code as a start. For this purpose, you can use a standard texteditor, save the file as `'filename.py'` and call it with a terminal (e.g. `xterm`). Be aware of the code indentation 🟥:

```
# Comment what the function is doing
def square(x):
     return x * x
print "Result:", square(4)
```

- Write a function that returns the sum of squares of two variables.
- Write a function that increments the value of a variable by 1.
- Write a function that increments the value of a variable by a number n.
- Write a function that calculates the factorial of a number (n!).

Conditionals

- Use the '`for`' and the '`while`' statement to display numbers between 1 and 10.
- Write a function that checks whether a number is between 1 and 10.
- Write a function that checks for two numbers the following conditions:
  `x > y`
  `x< y`
  `x = y`
- Write a function that checks whether a number is prime or not.

Optional: familiarize yourself with Jupyter notebooks `https://jupyter.org/`. We are going to use them in our next tutorial including libraries like matplotlib and scikit-learn.

@Home Task: For the next tutorial, prepare the following topics:

- Different attribute/feature types and their visualization
- Correlation analysis, causal relationships
- Statistical tests, Chi-Square Test

Note: the lecture script serves only as an orientation and you may need additional material (books, online tutorials) in case you did not fully understand the concepts. We will provide you with literature recommendations in the following tutorials.