A picture containing outdoor, object, clock, sitting

Description automatically generated

A guide to the course

Contents

[Introduction 4](#_Toc29153603)

[Why should you learn programming language? 4](#_Toc29153604)

[Structure of the course 4](#_Toc29153605)

[Grading 5](#_Toc29153606)

[Project 5](#_Toc29153607)

[Assignments 5](#_Toc29153608)

[Installation 6](#_Toc29153609)

[Installation of Anaconda 6](#_Toc29153610)

[Preparing of Anaconda 8](#_Toc29153611)

[Version Control 10](#_Toc29153612)

[What is Github? 10](#_Toc29153613)

[Cloning Github repository 10](#_Toc29153614)

[Modules 12](#_Toc29153615)

[Variables 13](#_Toc29153616)

[String 13](#_Toc29153617)

[Numerical values 14](#_Toc29153618)

[Boolean 15](#_Toc29153619)

[Conversion 15](#_Toc29153620)

[Naming conventions 15](#_Toc29153621)

[Operators 16](#_Toc29153622)

[Arithmetic operators 16](#_Toc29153623)

[Assignment operator 16](#_Toc29153624)

[Comparison operators 16](#_Toc29153625)

[Logical operators 17](#_Toc29153626)

[Flow Control 18](#_Toc29153627)

[If – Else 18](#_Toc29153628)

[ELif 19](#_Toc29153629)

[Objects 20](#_Toc29153630)

[Loops 24](#_Toc29153631)

[Functions 29](#_Toc29153632)

[Classes 31](#_Toc29153633)

# Introduction

Welcome to the *Introduction to the Computational Science*course by School of Energy Systems! My name is IK and I will guide you through the course.

Programming is literacy of the XXI century. Without programming we would not live in the world as it is now. Software is used everywhere and for various purposes, starting from clocks on your phone to controlling spaceships. For us, as engineers, it is useful tool for controlling and designing of the mechanical systems.

During the course you will need only a computer I strongly recommend to use your own computer, because it will make studying easier. Before starting the course

## Why should you learn programming language?

* Self-learner’s advantage: tons of material in internet
* Gaining problem solving and thinking skills
* Becoming more efficient and productive
* Combining technical skills with creativity
* Understanding how software works
* Freedom and flexibility in life
* Improving your communication and collaboration skills
* Becoming self-employed, starting your own business or work as freelancer
* Improving your portfolio

## Structure of the course

We will cover following topics:

* Syntax
* Files System
* Version Control
* Best Coding Practices
* Variables
* Python Objects (String, Dictionary, Tuple, Sequence, Set, Lists)
* Control Flow (Boolean Expressions/Operators, If, Else, Else If)
* Loops (For, While),
* Functions
* Classes
* Matlab Introduction (optional)

## Grading

## Project

Should be submited as Python file 🡪 No report needed

Group of 3 – 4 students

Deadline: 15.02.2020

Will be evaluated by following criterias:

* Creative idea, which is interesting for everybody. Useful for your studies, hobby or everyday life.
* Code is easy to read and maintain
* If used Github or Stackoverflow code must be changed for your needs. **NO COPYPASTING!**

## Assignments

Made during the lecture before break

Will be done in Microsoft Visual Studio Code

Submit Python file (.py extension) via Github, assignments folder

Deadline: Next week

Will be evaluated by following criterias:

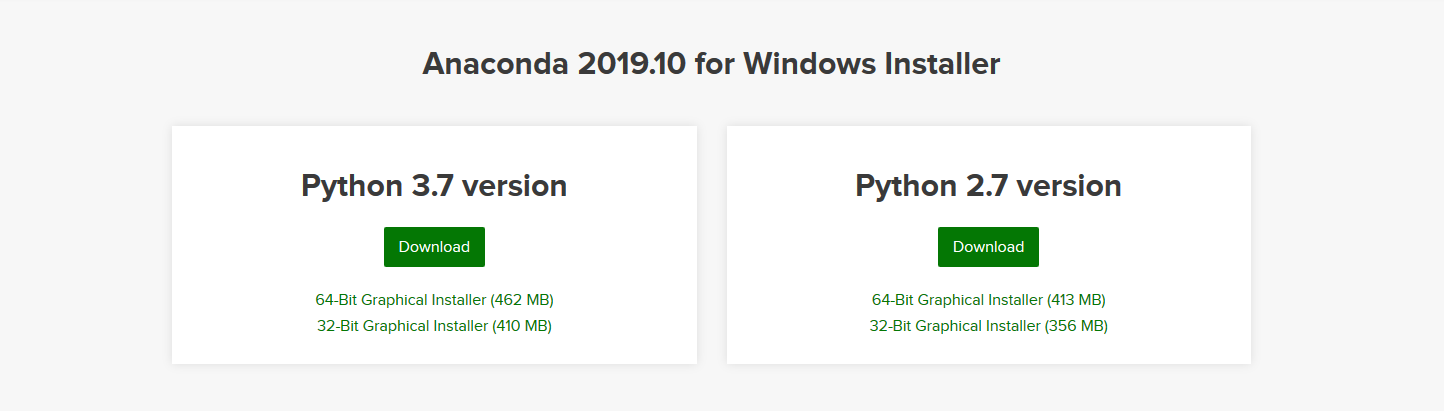
* Runs without exceptions
* Code is easy to read

# Installation

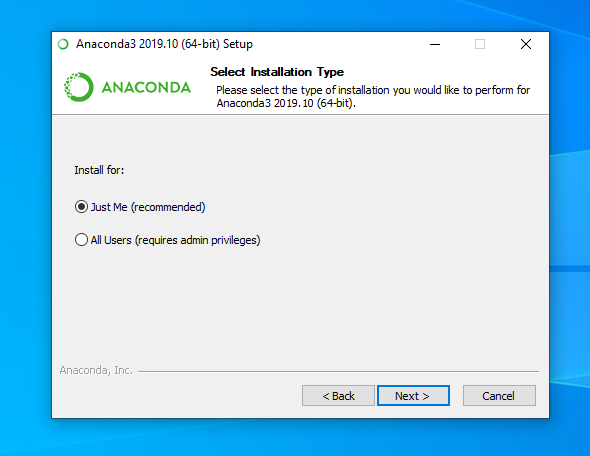
Anaconda is a distribution of Python made for scientific computing applications, such as data science, machine learning and etc. During this course we will use it for learning purposes.

## Installation of Anaconda

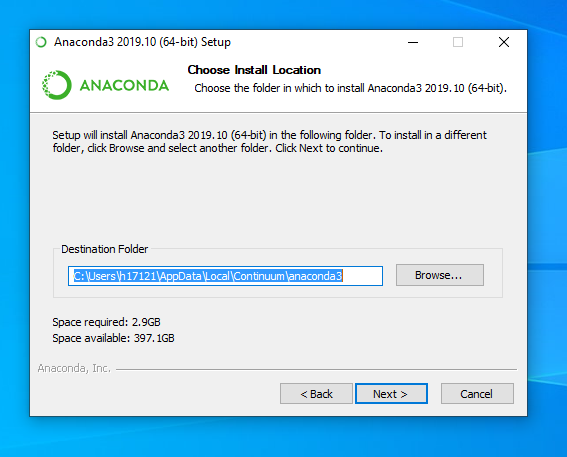
Step 1: Go to <https://www.anaconda.com/distribution/#download-section>. Load Python 3.7 version.



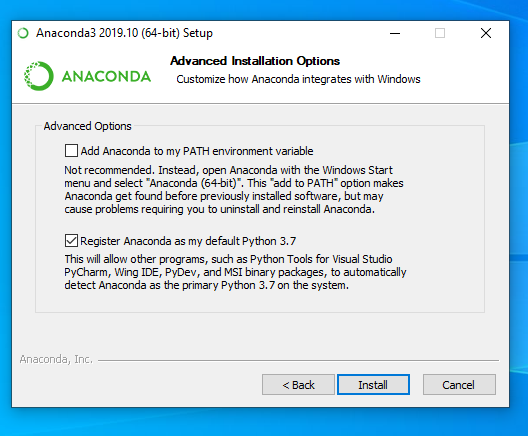
Step 2: Run the .exe file. Use install for **Just Me** option.



Step 3: Install to your user location (default).



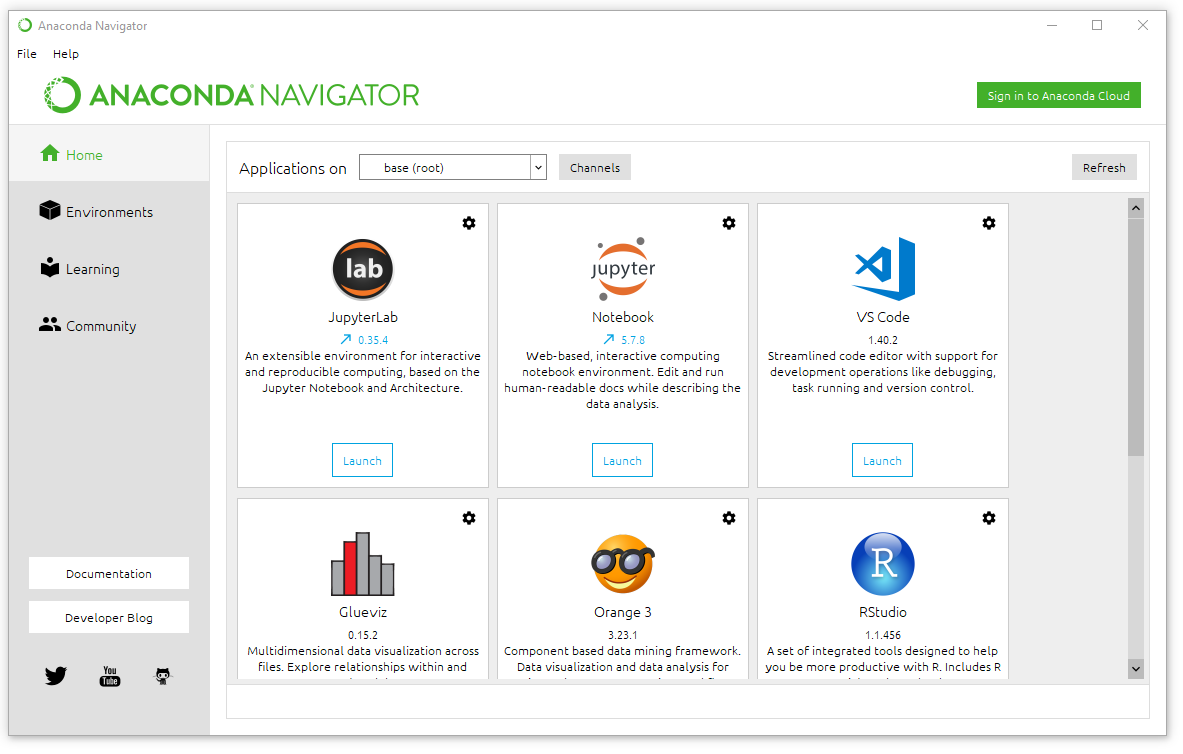
Step 4: Keep everything as default and click install.



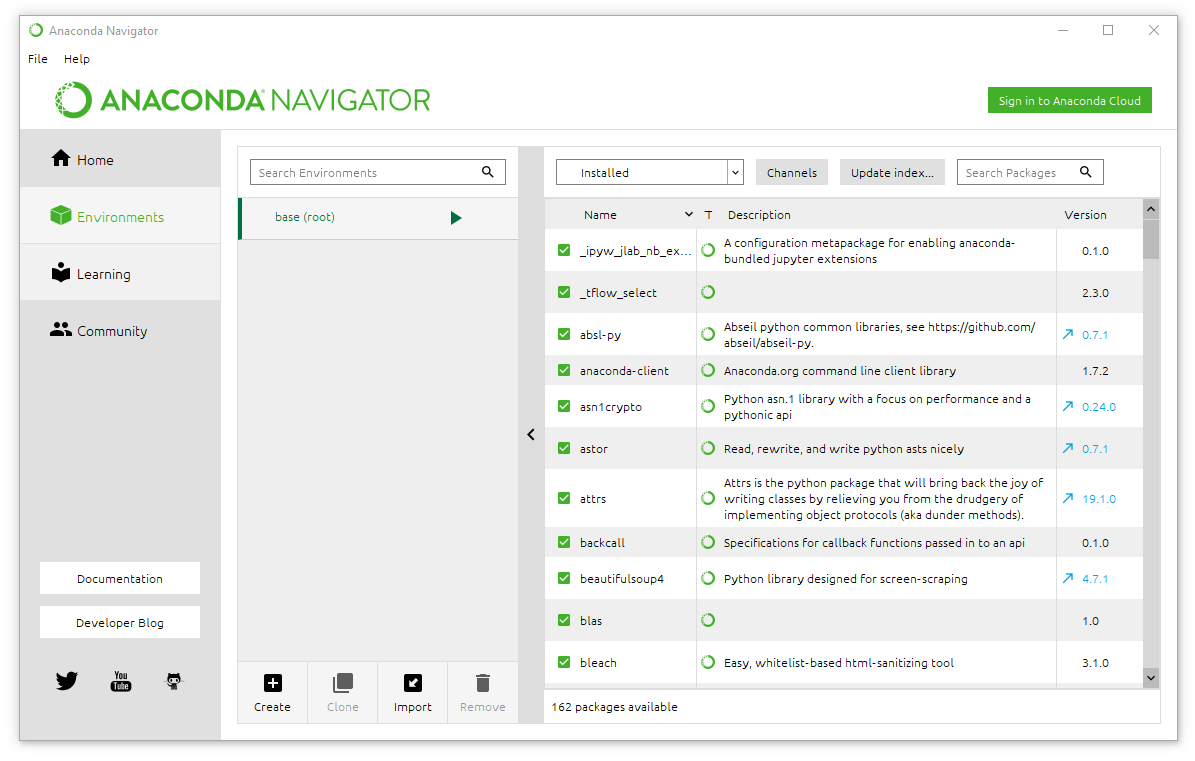
## Preparing of Anaconda

**WARNING! If you use your own computer, you can skip this step.**

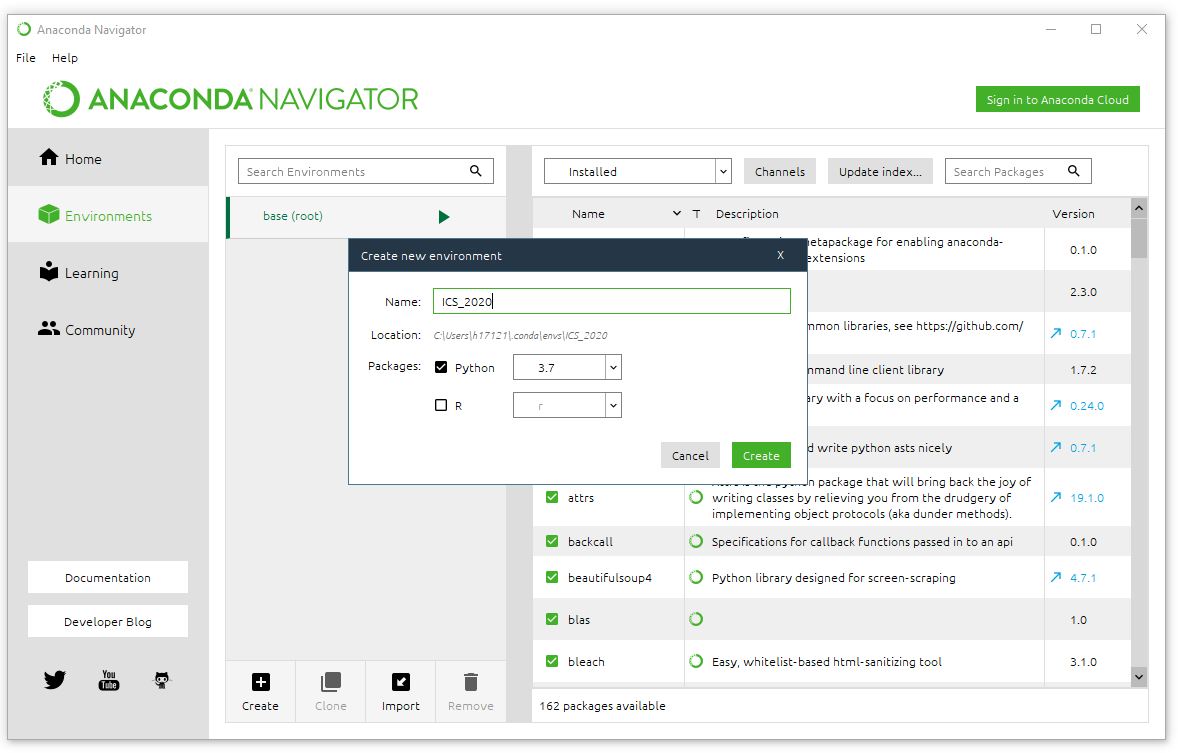
Step 1: Go to Anaconda Navigator. Open the Environments tab.



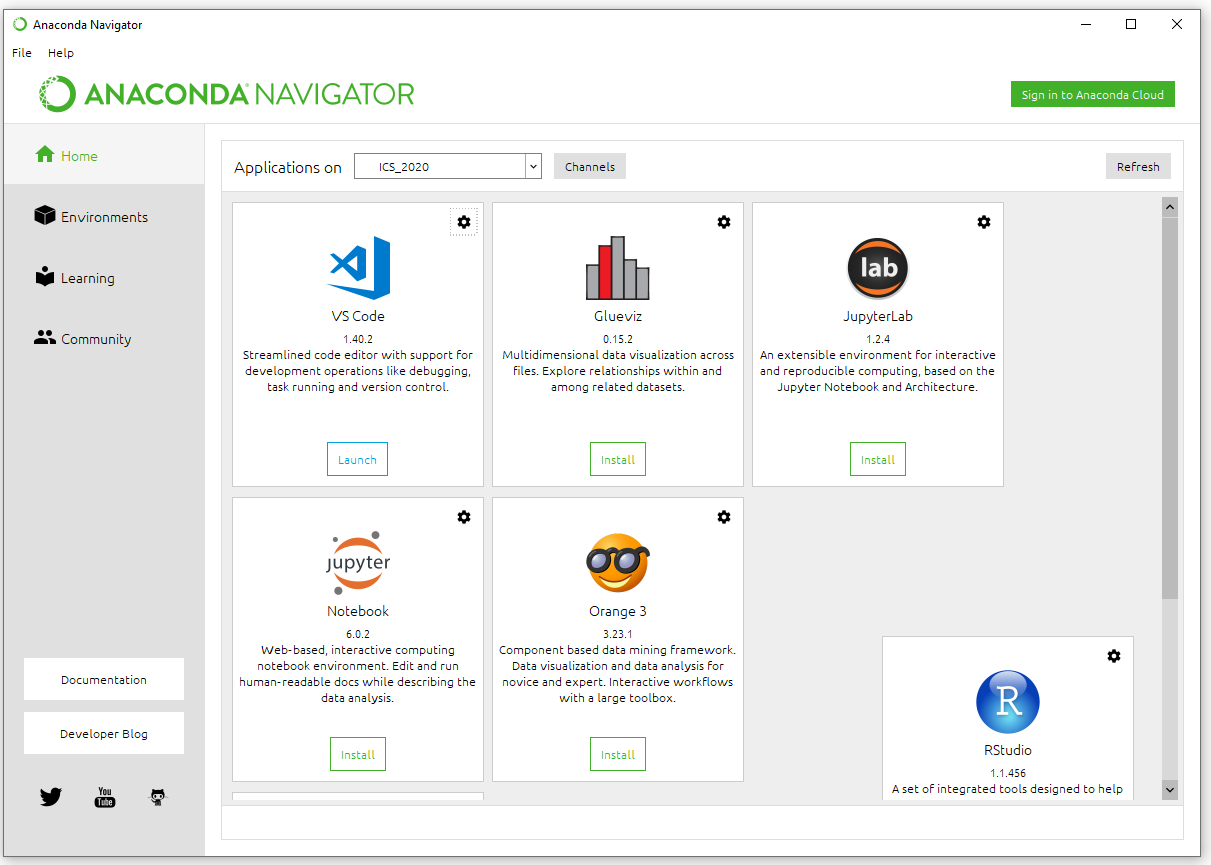
Step 2: Create new environment by hitting Create button.



Step 3: Create a new environment. Make sure that you use Python 3.7 version!



Step 4: The environment is ready for work. Now click Install button under the Jupyter Notebook. If computers asks admin rights, just click cancel.



# Version Control

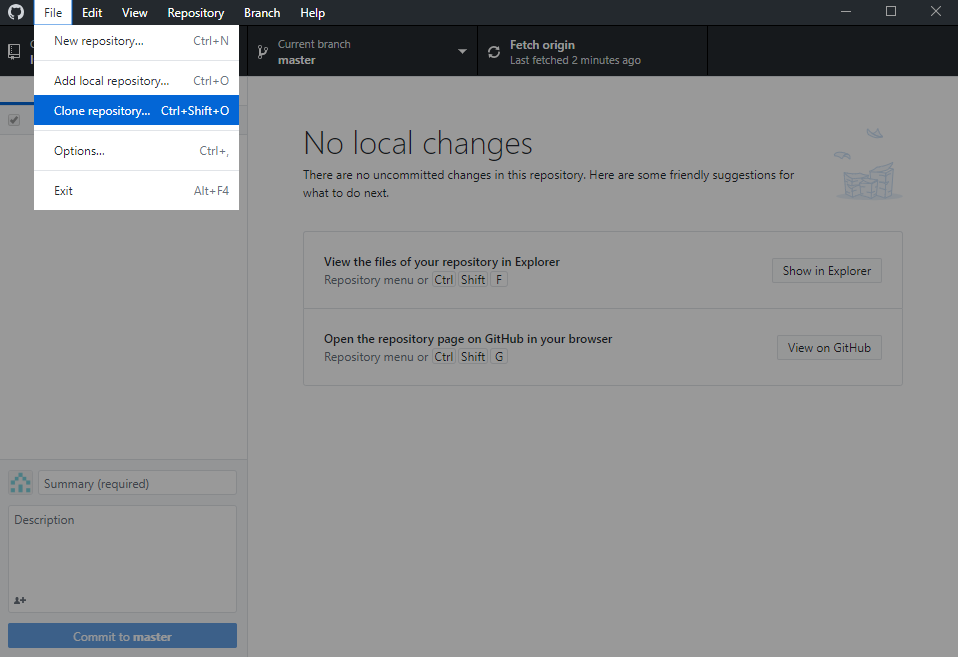
## What is Github?

GitHub is a platform for version control and collaborative work on code. It allows team of developers to work on the same project and track what was done before.

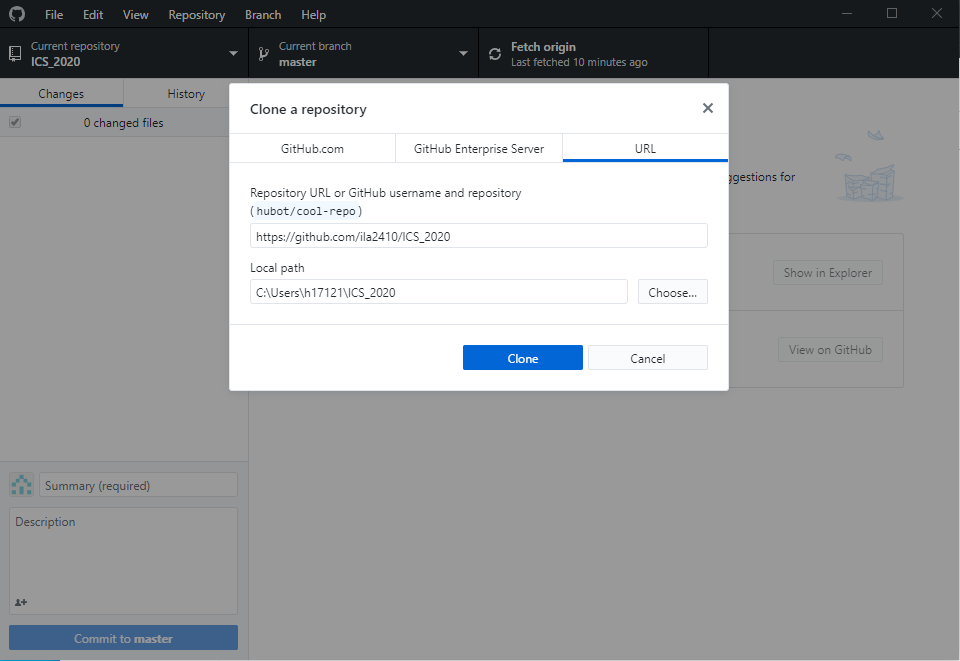
## Cloning Github repository

During this course all materials and assignment are located at the course Github repository. You should clone the repository using Github Desktop. You can clone repository by following these steps:

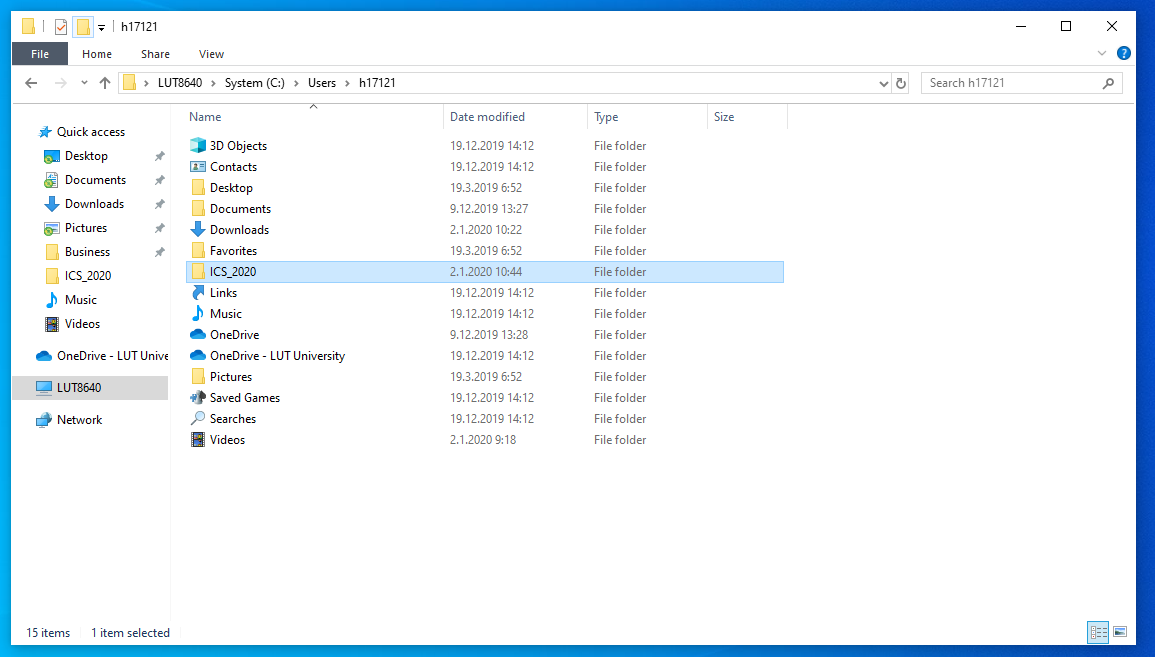
Step 1: Open Github Desktop. Go to File🡪Clone repository



Step 2: Go to URL tab, put URL of the repository (<https://github.com/ila2410/ICS_2020>) and choose path where to clone it. Then click clone button. **WARNING! If you are using LUT computer save to your C:\Users\your\_user\_number (for example C:\Users\n5198) folder, otherwise, it would not save anything.**



Step 3: Open File Explorer and go to the path. Now you will have all needed materials.

0

# Modules

Modules are files, which containing a script done by other programmers. It is used by programmer for not “reinventing the wheel”. You can add module by import command. For example:



For making your program more lightweight you can add only needed parts by using from. For example:

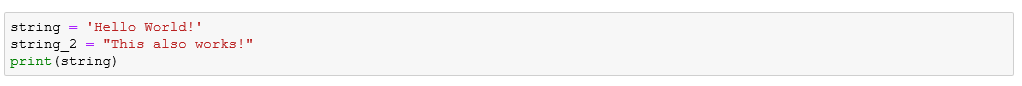


# Variables

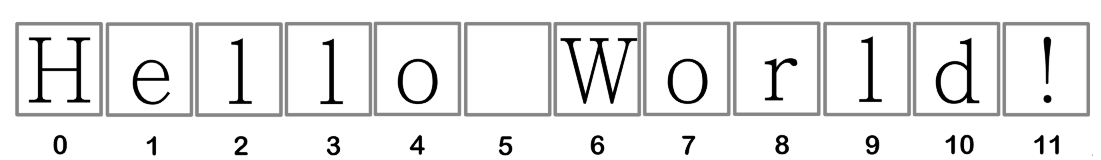
We need some way of storing any information and manipulate it. This is where variables come into the picture. Variables are exactly what the name implies - their value can vary, i.e., you can store anything using a variable. Variables are just parts of your computer's memory where you store some information. Unlike literal constants, you need some method of accessing these variables and hence you give them names. We will cover three major topics here: Numerical data, Text data and naming rules.

## String

**String** represents **text** data. A string is a sequence of characters. Strings are basically just a bunch of words. You will be using strings in almost every Python program that you write, so pay attention to the following part. String always is enclosed into quotation marks '' or "".



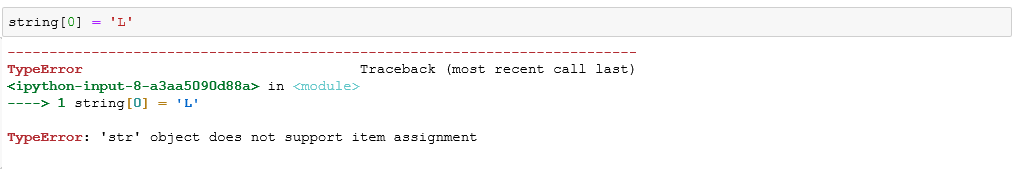
You can access a single character from string. Each character marked from 0 to number of characters.



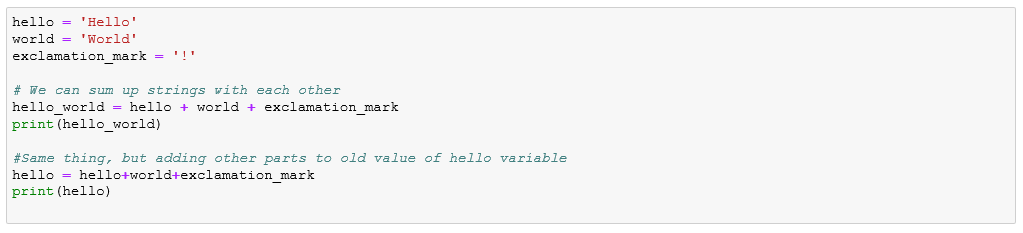
For example:



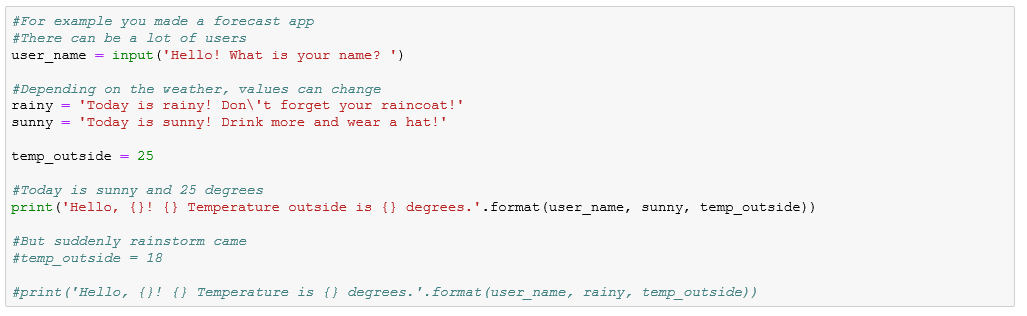
Once String is initiated, you cannot directly change it. It will show following error:



You can concatenate strings. In other words, create string by adding multiple strings together.



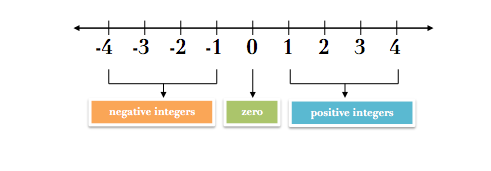
You can format the string by using format() method. By using this method you can add some value to the string in any place. In Python 3 you should add to the string {} brackets. For example:



## Numerical values

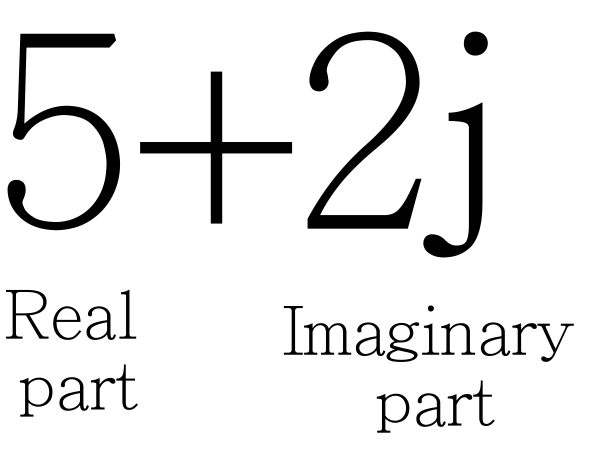
Python supports four different numerical types: integers(int), float, long and complex.

**Integers** are whole numbers, which can be negative or equal to zero, as shown at the figure below.



**Floats** are floating point numbers, meaning they have a decimal place. Same as integers, they are signed and can hold positive and negative values.

**Complex** are numbers with real and imaginary parts.



## Boolean

True or False variables. Used for tracking of events, in other words, check conditions if they are True. They will be very useful in flow control and while loops.



## Conversion

You can convert variables into other types:

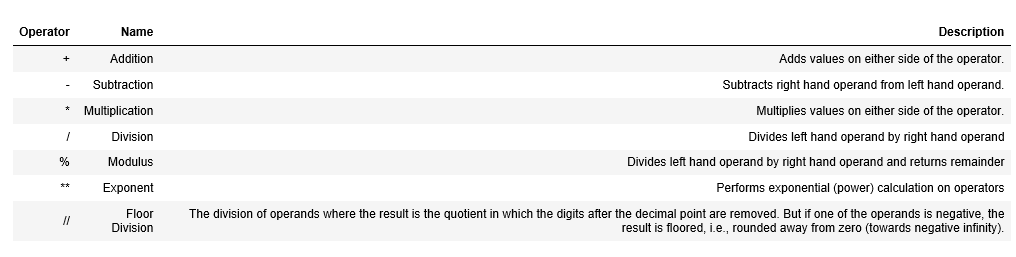
* str() – converts variable into string (bool, int, float, complex)
* int() – converts variable into integer (int, float, complex). You can convert string as well, but make sure, that string have only numerical values
* float()- converts variable into integer (int, float, complex). You can convert string as well, but make sure, that string have only numerical values
* complex() - converts variable into integer (int, float, complex). You can convert string as well, but make sure, that string have only numerical values

## Naming conventions

* Avoid using too general names. For example: variable\_1, number\_1, my\_list. Instead use descriptive names. For example, if you need to store the name of student use: student\_name
* You can use underscores or camelcase: under\_score CamelCase
* Don't start names from number
* Don't start from signs.

# Operators

## Arithmetic operators



## Assignment operator

**Assignment** operators are used to **assign values** to a variables.

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Example | Equivalent to | Explanation |
| = | x=5 | x=5 | - |
| += | x+=5 | x=x + 5 | Adds some value to the existing value of a variable. |
| -= | x-=5 | x=x - 5 | Subtracts some value to the existing value of a variable. |
| \*= | x\*=5 | x=x \* 5 | Multiplies existing value (x) of a variable to some value (5). |
| /= | x/=5 | x=x / 5 | Divides existing value (x) of a variable to some value (5). |
| %= | x%=5 | x=x % 5 | Takes modulus of existing value (x) to some variable (5). |
| //= | x//=5 | x=x // 5 | Floor division of existing value (x) to some variable (5). |
| \*\*= | x\*\*=5 | x=x \*\* 5 | Puts existing value (x) in power of some variable (5). |

## Comparison operators

These operators compare the values on either side of them and decide the relation among them. They are also called Relational operators.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Explanation |
| > | x>5 | Greater than. |
| < | x<5 | Less than |
| == | x==5 | Equal |
| != | x!=5 | Not equal |
| >= | x>=5 | Greater or equal. Meaning, that it includes a value. |
| <= | x<=5 | Less or equal. Meaning, that it includes a value. |

## Logical operators

Used for processing of Boolean or statements.

|  |  |  |
| --- | --- | --- |
| Operator | Example | Explanation |
| AND | x and y | Returns True if both of the values are True |
| OR | x or y | Returns True if one of the values are True |
| NOT | x not y | Returns True if if operand is False |

# Flow Control

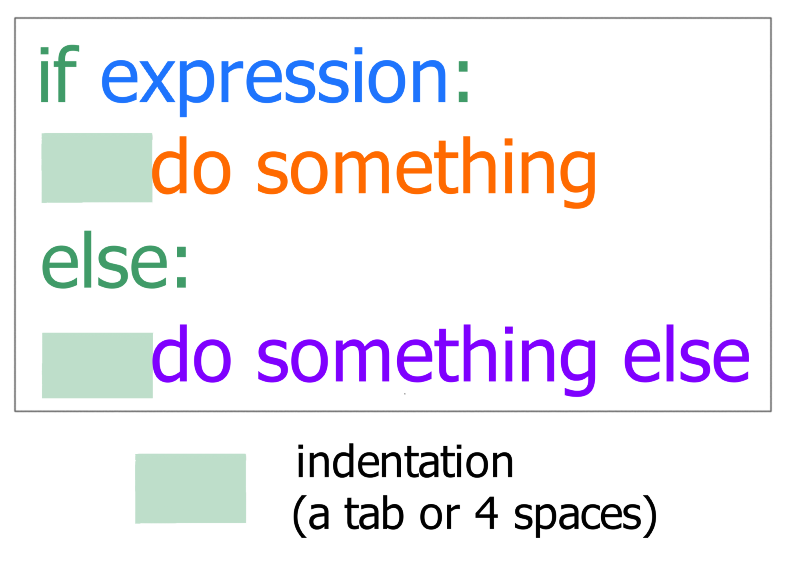
Flow control allow to execute some part of the code depending on the value of some expression. A expression can look like this:



It returns True if value of temperature is more than 25 and False otherwise. Other examples of expressions see in comparison operations chapter.

## If – Else

If/Else statement have the following structure:

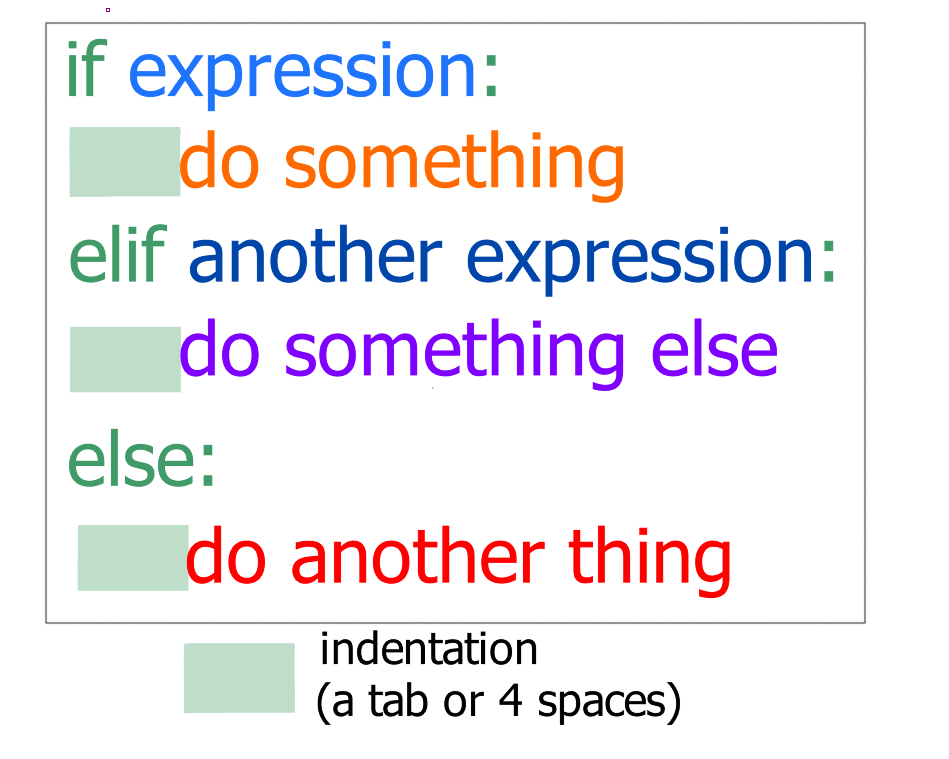


It is important to know, that everything under if must have an indentation. It means, that you need to add tab or 4 spaces. We use if statements in real life without even noticing. For example:



## ELif

Elif statement is somewhat self-descriptive. It is equal to adding another if to the script. It have the following structure:



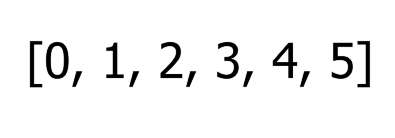
For example:



# Objects

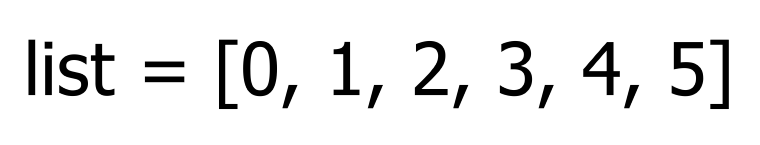
## List

List is a collection of variables which are ordered and changeable. You can store, access and change any type of variables in the list. Variable in the list called element. Basic list looks like this:

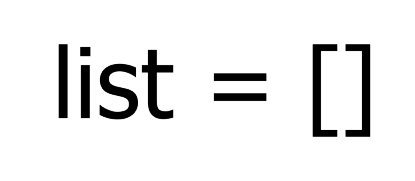


### How to create a list?

A list can be created by listing variables in square bracket as follows:



We can create an empty list simply by passing square brackets without any variables inside:



As it was mentioned earlier, in list can be stored any type of variables. For example, list below contains string values:



Lists can contain a mix of variable with different types. This is also a valid list:



List can contain other lists as well:

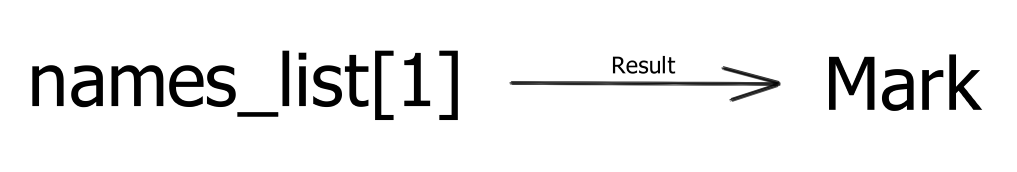


### How to access elements from a list?

There are two methods to access list’s elements. The first type is accessing element by index. At the figure below you can see how elements are indexed in the list. Note that list always starts from 0.



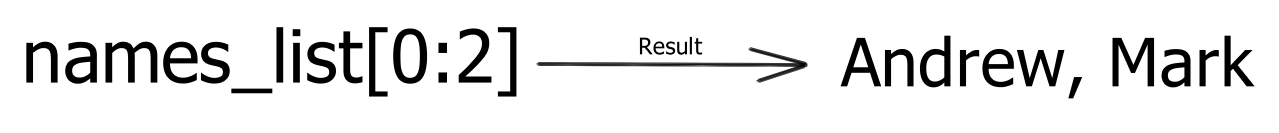
To access the element in the list you need to write a name of the list with index of a needed element in square brackets:



You can access elements with negative indexes as well. The negative indexes are the same thing as before, but they are listing from the end of the list:

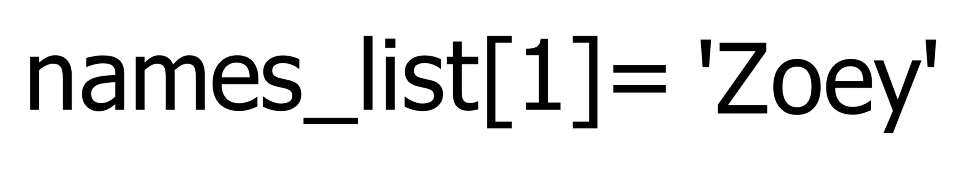


Another method of accessing elements in the list is slicing. With this method you can access multiple elements. You can make slicing as shown below. First digit inside of brackets defines where the starting elements and second is the stop element. Note, that last element is not included, therefore, in this example command outputs two values instead of three:



### How to change or add elements to a list?

Any element can be changed by accessing it by the index and assigning a new value. For example:



the resulting list after operation looks like this:



You can add elements by append() method. This method adds only one element at the end of the list.



To add multiple elements at the end of the list you need to use extend function:



### How to delete or remove elements from a list?

You can delete elements by using the del keyword. For example, if we want to delete ‘Stan’ for the list, we need to write following command:

**del names\_list[2]**

We can delete multiple elements. For example, we can delete ‘Andrew’ and ‘Mark’:

**del names\_list[0:2]**

We can delete whole list:

**del names\_list**

We also can use remove() function, which deletes elements by their values.

**names\_list.remove(‘Stan’)**

Another way to remove element is pop() function, which removes and returns value of an element by index. If index is not provided, it returns last element of the list.

Also, we can clear all data from the list by using clear() function:

**names\_list.clear()**

## Dictionary

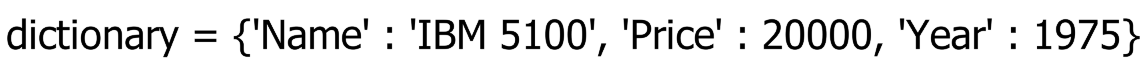
Dictionary is a collection of variables which are unordered, indexed and changeable. Variable in a dictionary called item. Instead of indices, dictionary using keys – a name of the item.

### How to create dictionary?

A dictionary is created by stating key-value pair (in our case is name – IBM5100) inside of the curly brackets:

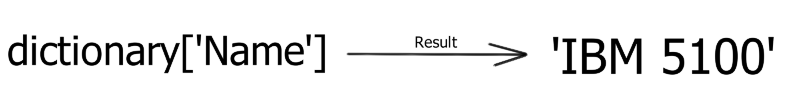


You can add as much key-value pairs as you like:

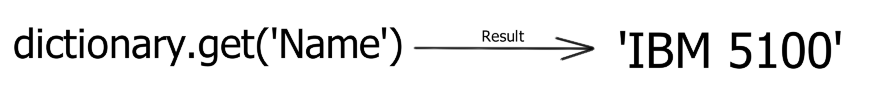


### How to access item of a dictionary?

You can access item by writing a name of the dictionary with key of a needed element in square brackets:

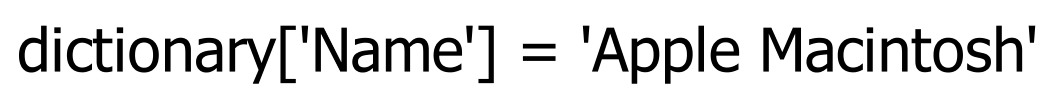


Same result we can get via get() function:



### How to change the value of an item?

Similarly to list, you can change item by it’s name:



### How to add an item to a dictionary?

You can add a new item by using new key-value pair:



### How to remove an item?

You can remove items as in lists with del, remove() or pop().

**del dictionary[‘Price’]**

**del dictionary**

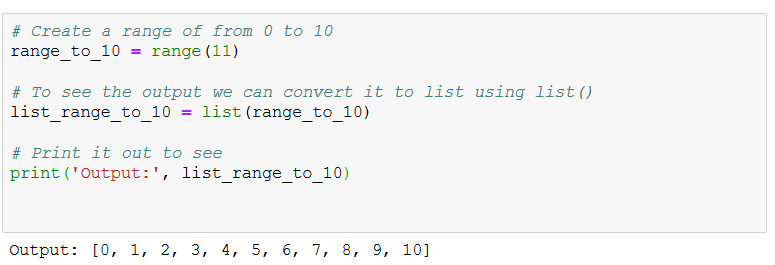
**dictionary.pop(‘Price’)**

**dictionary.popitem()**

# Loops

## Range()

Let’s first go through creating the range of numbers. We can define some sequence of numbers for example from 0 to 10 by using the range function. This function creates following sequence of numbers:



Note, that number which passed to the function is 11. This is because every counting in programming starting from 0. In other words, this is the stop number, which is not included.

We can make a range starting from certain number. This can be done by passing first starting number and then ending number as follows:





The starting number is 3 and 7 is stop number. As you can see, it wasn’t included.

Also we can provide a step for a function. As a default it is 1. Let’s change the step and see what happens:





## For loop

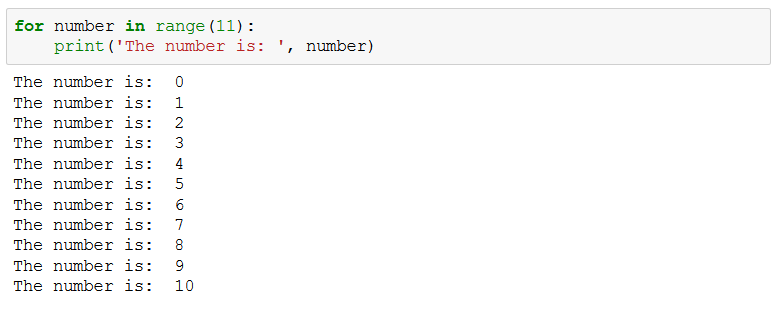
For loop is a iteration over some sequence. This sequence can be a list, dictionary, range of numbers or even a string. It is executing a set of some statements **for each element** in a sequence. This means that for each element of the sequence will be done some operations. The syntax is following:



The variable is an element from a sequence which is used in iteration. You can call it as you like. The sequence here can be range, list, dictionary or string.

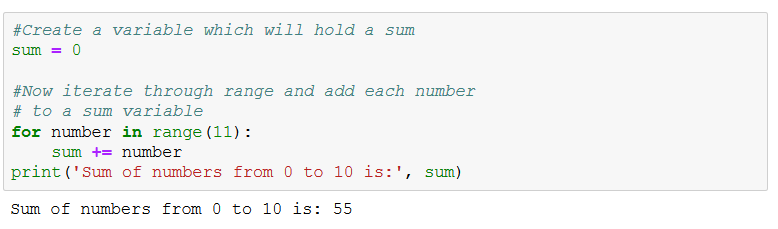
Let’s go through iteration over the range of numbers. As we done before, we can create a range of numbers, for example, from 0 to 10. With loops we can access each number in order and do some operations on it.

First, let’s try printing them out and see what happens:

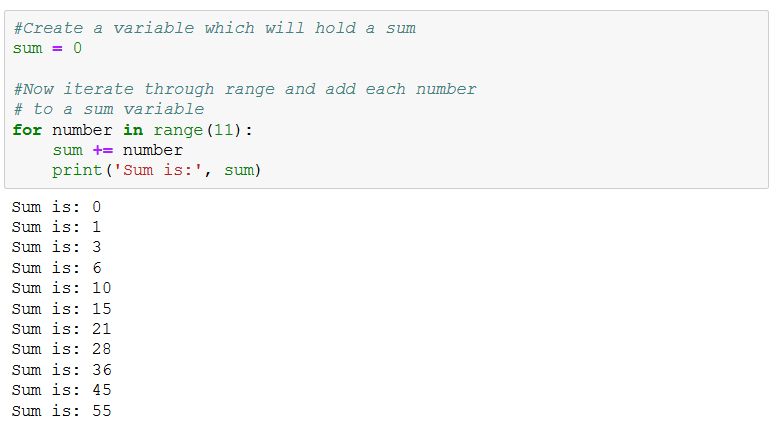


As you can see, it took a number, printed it out, took next number and printed it out till all numbers are used.

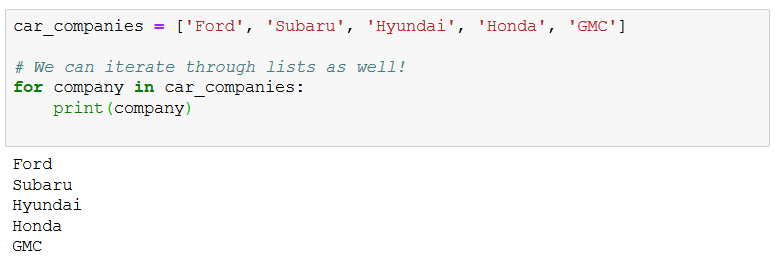
We can use the for loop for more practical task as defining sum of these numbers as follows:



This code added each number in a sequence to a sum variable one by one. We printed sum outside of the loop to see the final result. If we will print at each iteration, then you can see how sum is changing after each iteration:

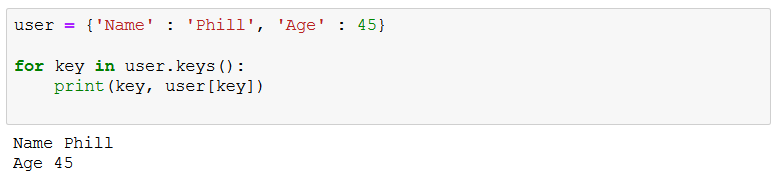


As we been discussing it before, we can iterate over the lists. For example:

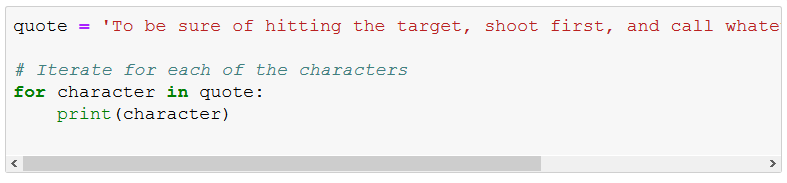


The idea is the same, we go through each element of the list and do some actions on it. In example before, we printed out each element of the list.

We can iterate over dictionary as well. For example, we can take keys of dictionary using keys() function and do something with values in the dictionary:



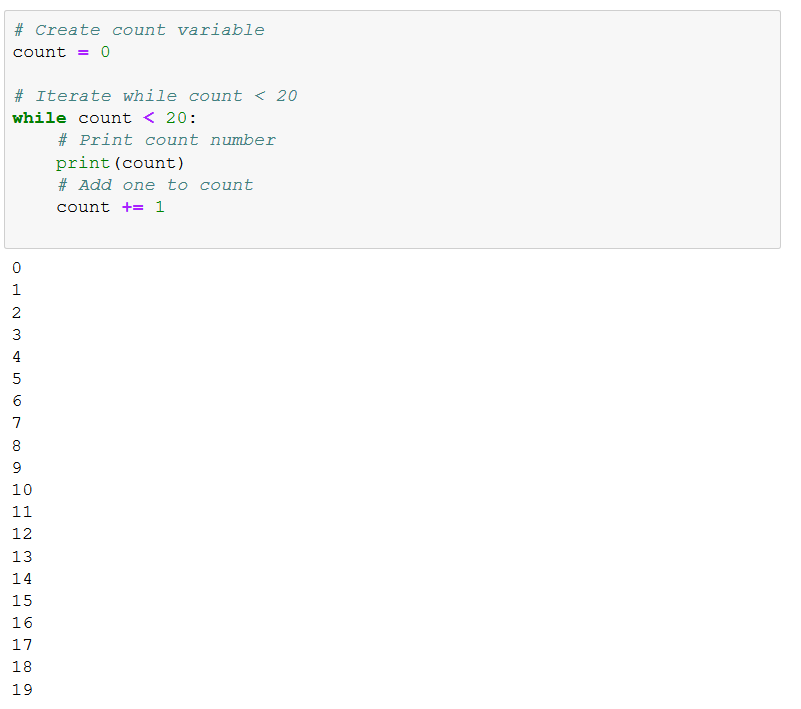
Also, we can iterate through string. In this example we will do something with each character of the string:



As an output, we will have each character printed separately.

## While loop

While loop is a bit different. It is executing code till some condition is met.

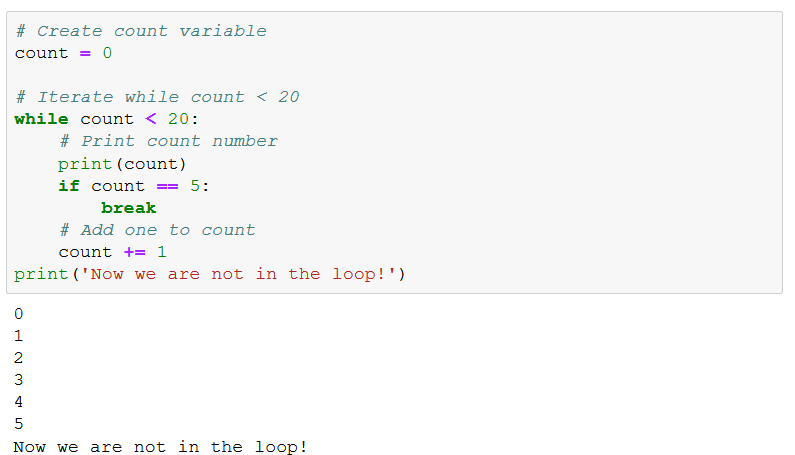


In this example we printed value of count variable and added one at each iteration.

## Break and continue

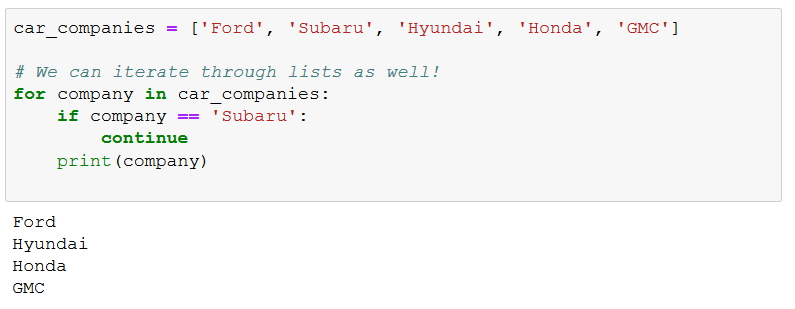
Iteration in loops can be additionally controlled by two keywords: break and continue.

Break is used for stopping iterating (going out of a loop) and do code after a loop. We can try it on the previous code, for example we don’t want to continue iteration after count equals 5:



As we can see, our loop ended when count equals 5.

Another keyword is continue. It is skipping current iteration and going to the next iteration. On the example of our car companies iteration we can see how it works:



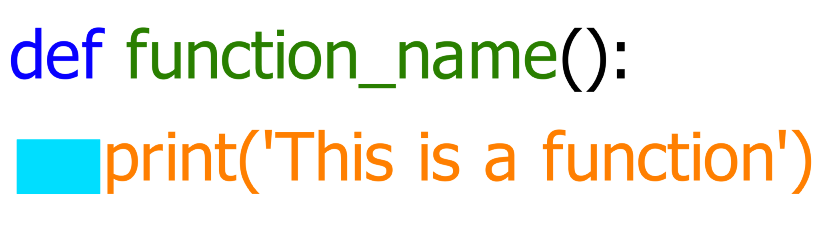
On the figure above we can observe, that when company is Subaru, code skips iteration and not printing value of the company.

# Functions

A function is a part of the code, which runs when it is called. Originally, concept of functions came from math. It is made for easier calculation of complex problems by dividing them into small blocks. For example, basic math function has following form:

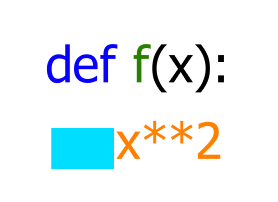
Functions may have input and output. In the case of example math function, input is a value of x and x^2 is an output.

In Python, functions have the same idea. Function has a name, may have input and output. Before calling a function, you should initiate it by following code:

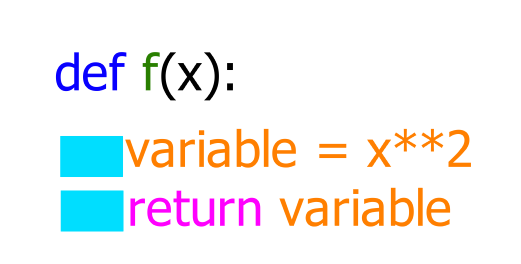


where, def is keyword for initialization of the function, function\_name is a name of the function which you can choose and parenthesis are a place for argument.

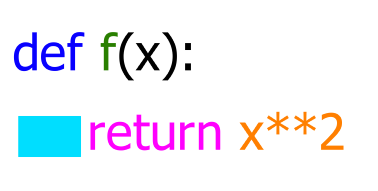
If we want to create math function from example, we need to pass x value to parenthesis. It will take this form:



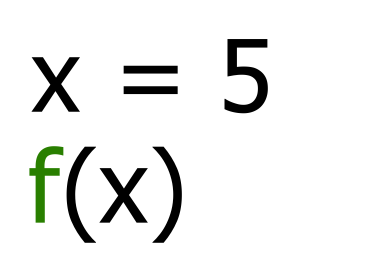
Now this function doesn’t output anything, because we need to provide variable to output. It can be done by return keyword:



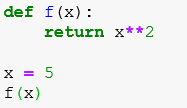
Or



Here we told computer to output a result of the mathematical operation and fully modelled behavior of math function. If we will run the code it won’t output anything, because we initiated a function but didn’t call it, meaning, we need to ask computer to execute this function:



Above, we provided value of x and provided it to the function. As the output, we will have an output of 5^2 🡪 25. Resulting code should look like this:



# Classes

COMING SOON