1 About Python

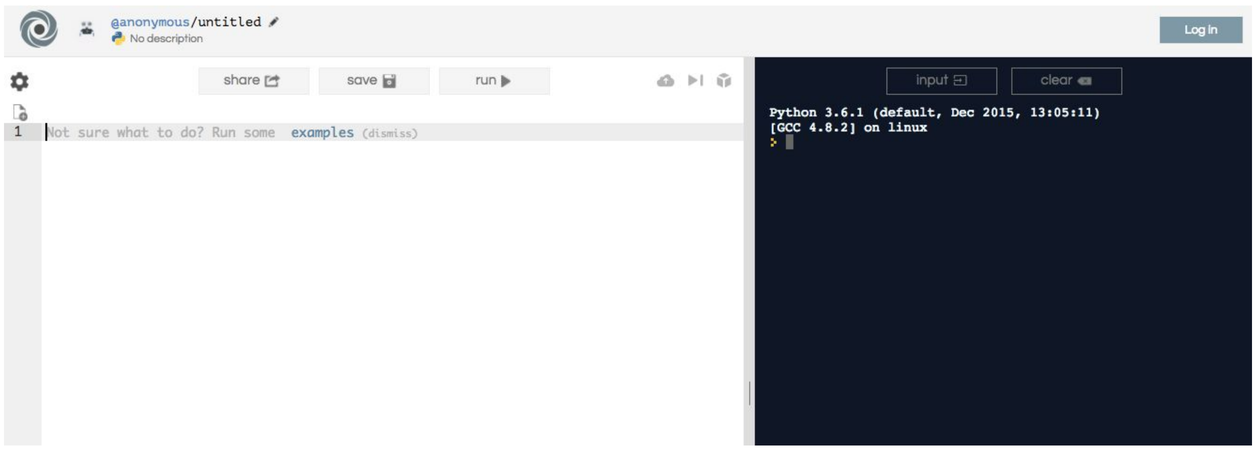
Python is a very popular programming language. Some known applications that use Python are BitTorrent, Dropbox and also some stuff on Youtube.

To work with python you need to download some Python interpreter that reads your Python files and tells your computer what to do. It's like a middleman between you and computer.

For start, we used a simple online python editor, something like Google Doc for Python. You can access it through the following link.

https://repl.it/languages/python3

This is what you should see. You don't have to sign up.



```
Not sure what to do? Run some examples (dismiss)
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
```

The **left hand side** window if where you'll be writing your code for the program and it will be saved. You need to *run* the program then to see things happening. If we don't say otherwise, stick to the left hand window.

The **right hand side** window is Python console. This is interactive version of Python. You will see things happening there when you run your code. You can also write in there straight away and it executes things immediately. It's useful for experimenting but does not allow you to save your work.

TASK 1

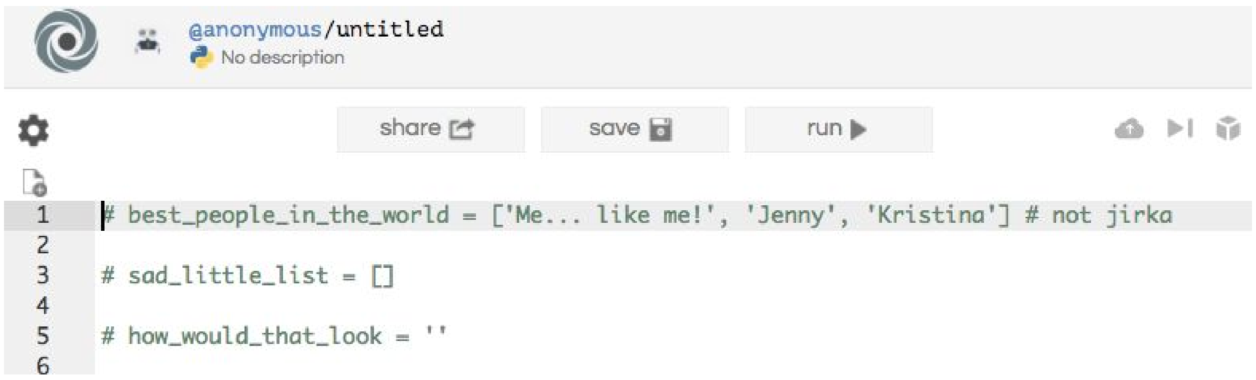
a) In the left window, type
   print (1+1)
   And then press the button *run* that's above the editor

b) In the right window, type
   1+1

Observe the difference in what you write and the output. You can use Python as a calculator.

It's quite easy to work with python because it communicates with you. It tells you when you made a mistake in your code and tries to explain where. So don't be scared to experiment and try your own thing!

NOTE: Everything you write in the left window will affect your program unless you put a **hash** # in front of the line. Programmers use this to leave comments for one another to explain human things.



```
# best_people_in_the_world = ['Me... like me!', 'Jenny', 'Kristina'] # not jirka

# sad_little_list = []

# how_would_that_look = ''

```

If you write something like that, there is essentially nothing for the program to read so if you press *run*, nothing will happen.

TIP: You can hash tag all your lines at once with a shortcut **CMD** or **CTR** + /

2 Variables

Variables are one of the most fundamental concept of all programming languages. They allow us to store and retrieve values.

```
>>> name = "Martin"
>>> print(name)
Martin
```

If you write this, the program will always remember that any time you type *name*, you are referring to *Martin*. If you decide to change from Martin to Jirka, you can always change it later on. You can name your variable however you want. It helps to choose a logical and intuitive name.

The following example will show you a basic application of having such variables:

```
>>> pi = 3.14 # precise enough
>>> r = 10
```

TASK 2

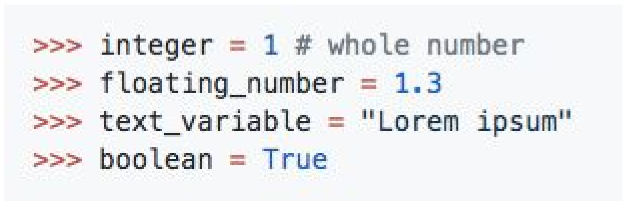
Can you create a variable called *area* and then equal it to a formula that includes the above variables in it and actually yields the area of a circle?
Can you then write:
print(area)
run the program and obtain the correct area of a circle with r = 10?

Name requirements:

- variables can't start with numbers - `1apple` is invalid while `apple1` is valid.
- you can use undescores but not dots or `-` signs. So `speed_of_light` is valid while `speed-of-light` and `speed.of.light` are both invalid.
- names are case sensitive `Speedoflight` and `speedoflight` are two different variables!

3 Kinds of variable - basic data types

What are types? They are representation of different values - in real world you have many types of objects - numbers, cars, devices, pigment types, genders... In computer science they more precisely defined, this is to ensure that computer understands the meaning of the value.



```
>>> integer = 1 # whole number
>>> floating_number = 1.3
>>> text_variable = "Lorem ipsum"
>>> boolean = True
```

You can't add text with numbers - that simply does not make sense! And so Python will complain to you with an Error message.

Python will differentiate the types according to the form you write them in. As you can see, floating numbers will have a dot to show decimals while text variables, also called strings, will always be marked by either ‘ .. ‘ or “ .. “ . You can choose whichever. Booleans are special type and can have value of either True or False. We’ll work with them later.

Question 1

What types are the following variables?

1. Date_of_birth = “12.10.90”
2. Date = 24.12
3. I_love_you = False

4 Lists

Lists are Python way of storing multiple values in an ordered array. Lists are defined by square brackets, and comma separated values. Usually it makes sense to store values of same type.

```
>>> fruits = ["bananas", "apples", "oranges", "melones"]
>>> numbers = [1,2,3,4,5,11,-1]
>>> mixed = [1,2,3, "bananas"]
```

Question 2
Why are some of the items in the lists in quotation marks and some are not?

If you want to play around, below are some things you can do with lists. You are not required to remember these just yet.

Simple operation with lists

A. **Adding** more items to list
- Write *.append()* after the name of the list and put the item in the brackets. For example:

fruits.append("pear")

B. **Extending** list with another list
- Write *.extend()* after the name of the list you want to extend and put the name of the other list in the brackets. For example:

fruits.extend(numbers)

C. **Selecting** one element from the list
- Write the name of the list followed by [ ] with a number expressing the order of the item in the list.
REMEMBER: The first item in the list always has order 0, not 1.

fruits[0] will give you *bananas*, not *apples*

D. Find out the **number of items** in a list
- Write len( ) and put the name of the list in the brackets. For example:

len(fruits) will give you 4

5. INPUT

Input is a function that when you run it, it asks you for something and expects to receive an answer from you. For example, if you are making a survey and you want someone to fill in their place of birth, you can create variable called *place_of_birth* and ask *Where were you born?* using this input function in the following format:

```
Place_of_birth = input (“Where were you born?”)
```

The program will not continue to run until it receives an answer for its question.

Try to apply this knowledge in the following task.

TASK 3

Write a little program that asks you to input your first name, then surname and then prints your full name on the 3rd line.

Well done for making it here!

We recommend you take a break now.

6 Conditions IF, ELIF, ELSE

Try to go over these notes from last year
http://progsoc.visgean.me/learn/lecture02/

When you finish, complete the task below. If you manage, you'll be ready for the next session. If not, come a bit early and we'll help you.

TASK 4

Create a program that asks you to input your age and then say: "Tough luck!" if your age is more than 30 and that says "You shouldn't be here!" if your age is less than 5. It should say "Welcome to our society" otherwise.

Hint: The Input function expects to receive a text variable as an answer not a number. To convert text into integers, you need to somewhere place a function
int() for which you place the text in the brackets and it will convert that text into an integer so you can compare the two integers in your condition.

For example, if you define a variable called *answer* as
answer = input ("What's 1+1?")

and you respond 2, it will think of 2 as a text, not as a number. You then have to create another variable called number_answer as
number_answer = int(answer)

And now the program will know that it's dealing with integers not text.