**Introduction to basic data types**

Download the program from the link <https://www.anaconda.com/download/>

**Step 1. Types of numbers in Python**

**FORMULATION OF THE PROBLEM**

In the first few steps, we will solve the following problem:

The project is monetized in two main regions - Astana and Almaty. In both regions, the number of users and the revenue received are known. It is necessary to understand in which region monetization is more effective. That is, calculate the ratio of revenue to the number of project users in each region and compare these values ​​(this metric is called ARPU - Average Revenue per User).

In Astana, the project has 4,900 users and revenue for the reporting period is 63,000 tenge, in Almaty - 3,500 users and 48,000 tenge of revenue.

We will get acquainted with the basic data types in Python and master the operations of formatting the results, formatting the comparison in the following format:

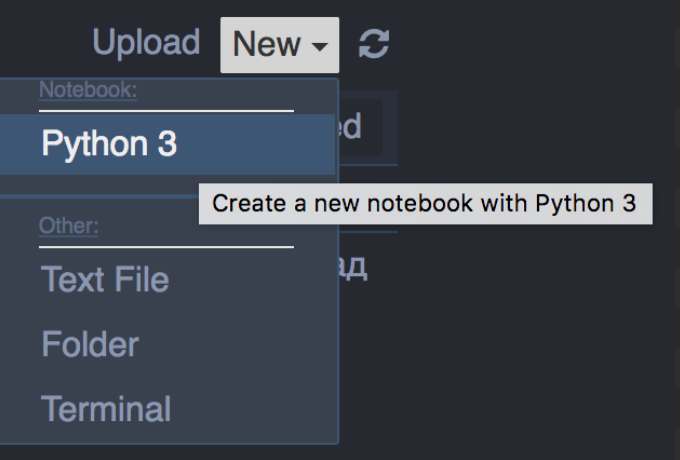
**Revenue per user by region: Astana - 12.86 tenge, Almaty - 13.71 tenge.**

**Step 2. Displaying values on the screen**

**DISPLAYING VALUES ON THE SCREEN**

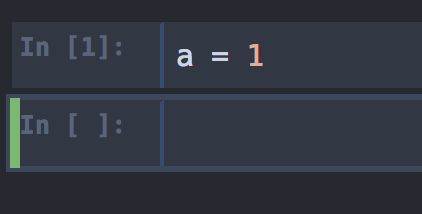
To calculate the revenue per user, it is necessary to save the number of users and the revenue in variables so that in the future it will be easy to perform various operations with them. In our problem, the number of users and the revenue are integers. There is a separate type for them in python, which is called *integer*.

Open a new file in Jupyter notebook and run the following example in it.



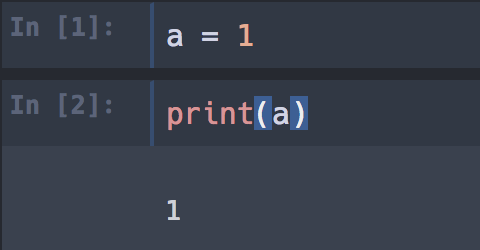
Example (enter this line into Jupyter notebook and press Shift + Enter):

a = **1**



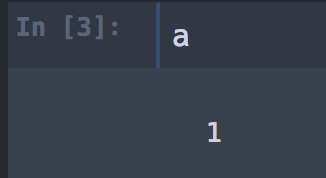
The variable a now stores the value 1. Let's print the value to the screen:

**print**(a)



**Advice**

In Jupyter notebook, to display the value of a variable on the screen, you can simply use the name of the variable a, print is optional:



**Step 3. A little practice**

Go back to your Jupyter notebook and test the basic math in variable a:

* + addition a + 1

a + **1**

* + subtraction a - 20

a - **20**

* + multiplication (a + 1) \* 2

(a + **1**) \* **2**

* + division (a + 5) / 3

(a + **5**) / **3**

* + exponentiation (a + 1) \*\* 5

(a + **1**) \*\* **5**

An exercise

What value should have come out in the last example (a + 1) \*\* 5? Recall that initially

a = 1

**Hint (1 of 1):** the value of a in our example was 1. It remains to raise (a + 1) to the fifth power

**Step 4. Code design**

**CHOOSING NAMES OF VARIABLES**

In simple one-line examples, the name doesn't really matter. When writing code, it is better to give variables meaningful names. This will greatly help you and your colleagues to understand how your code works. For example, our problem has 4 variables: the number of users and the revenue for Astana and Almaty. In order not to get confused, which variable belongs to which region, use more detailed names of these variables.

Astana users (4,900 users and 63,000 tenge of revenue):

• users\_astana - the number of users from Astana;

• revenue\_astana - revenue of users from Astana.

Similarly for Almaty (3,500 users and 48,000 tenge of revenue):

• users\_almaty - number of users from Almaty;

• revenue\_almaty - revenue of users from Almaty.

Typing long names in the code is not a problem, since Jupyter notebook has tab completion. By typing the first few letters of the variable name, you can immediately get the ready variable name through tabulation. An example of writing a variable like usersAstana is called **CamelCase**. You can also use variable notation like users\_astana (**snake\_case**). You can choose any method that is convenient for you.

An exercise

Set the users\_astana and revenue\_astana parameters to 4,900 and 63,000. In the users\_almapb and revenue\_almapb parameters, write down the values ​​3,500 and 48,000.

What is the sum of the revenue of both regions?

CONCLUSION OF RESULTS

To work with words and texts, there is a string data type, or string. A string is usually understood as a set of characters in quotes. For example, "hello". In this case, you can use double quotes ", or single quotes '. That is, the entries my\_string =" hello "and my\_string = 'hello' are identical.

In Python, strings can be concatenated with each other using the addition + operation. Let's set the variable project\_name to a string.

project\_name = 'Python for data analysis'

An exercise

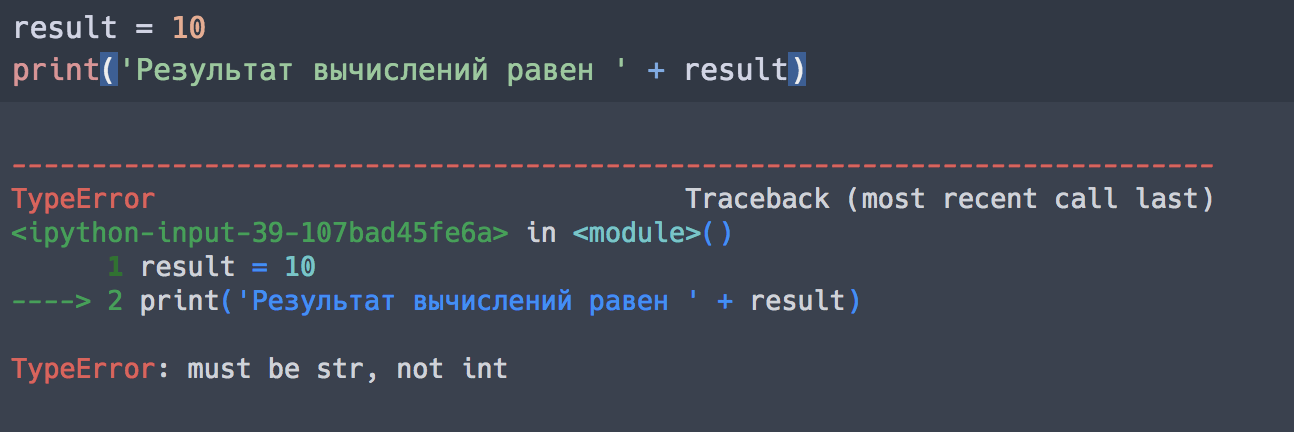
What will the system output for the next line of code?

**print**('Course name '+ project\_name)

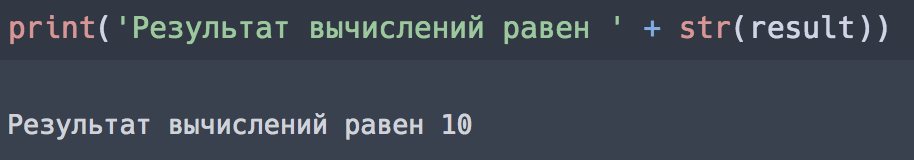
**Step 2. Output of numerical results**

**OUTPUT OF NUMERICAL RESULTS**

As noted in the last step, the *addition* operation can be applied to strings. However, if we try to comment on the numerical result, we get a type incompatibility error:



That is, we are trying to "add" variables of string and numeric types. The system does not understand how to add them and gives an error. In order to still display the numerical result with text, convert the result variable to a string using the str function:



Now everything is correct: we "add" two lines and get the desired text on the result.

**EXERCISE (WITHOUT ONLINE CHECK)**

Follow these steps:

• take the revenue and the number of users of Astana from the previous block (4,900 users and 63,000 tenge)

• display the ARPU value with a comment in the form:

ARPU value of Astana users is 12.857142857142858

**Step 3. Output format**

**FORMATTING STRINGS**

The result in the last exercise looks very ugly. For a visual display of the results of calculations, we need to set the output format. For example, even the same result can be displayed as a fractional number and as a percentage. There are several methods for this. We will use the most popular and easy to remember option. Let's take the data from our example:

users\_astana = **4900**

revenue\_astana = **63000**

arpu\_astana = revenue\_astana / users\_astana

Let's format the result of our simple calculations in different formats. Let's try to write the phrase: 'Astana: 4900 users, revenue 63000 tenge, ARPU 12.86'

Let's write the words first, and instead of the values 4,900, 63,000 and 12.86, put two curly braces (we will not execute this code for now):

**print** ('Astana: {} users, revenue {} tenge, ARPU {}')

The curly braces will contain the values of the users\_astana, revenue\_astana and arpu\_astana variables, which we will pass using the *format* method:

**print**('Astana: {} users, revenue {} tenge, ARPU {}'.format(users\_astana, revenue\_astana, arpu\_astana))

Accordingly, the sequence of desired values in place of the curly braces must match the sequence of variables that we specify in format. This format is very convenient for displaying results: when writing code, we do not need to translate different types of variables into strings. Moreover, with a large number of variables, this format is much more convenient than our previous version with "adding" strings.

Now, when executing this code, we will receive a line with our variables: What will be the results?

**Step 4. Setting the format of the variables**

Let's now give our output a more readable format. For numbers, the number of digits after the decimal point is set as follows:

• in curly braces we write: 2f, where 2 is the required number of decimal places, f is an indication of the float type;

• if you need to specify the percentage format, then instead of f we put%.

Let's write down the result of calculations more clearly:

**print**('Astana: {} users, revenue {} tenge, ARPU {:.2f}'.format(users\_astana, revenue\_astana, arpu\_astana))

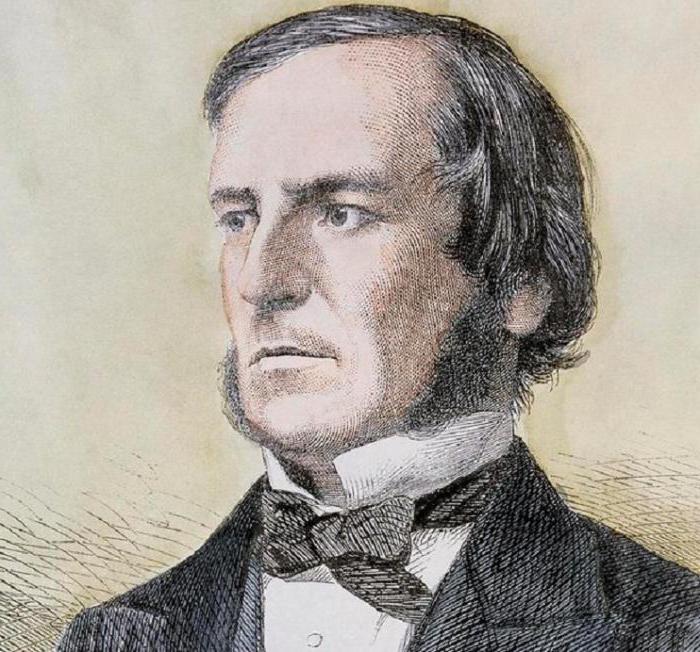
An exercise

Print a similar result for Almaty users, rounding the result to the fourth decimal place.

What is the ARPU value for Almaty when rounded to the fourth decimal place?

**Step 5. Simple checks**

It remains for us to compare the efficiency of monetization of users in Astana and Almaty. For this, you can use the so-called boolean data type (bool). This name comes from the surname of the mathematician George Boole.



Variables of this type take only two values: True (true) or False (false).

This type of variables is convenient to use in various tests for conditions. Of course, it can be replaced with any other type. For example, use the int type with values 0 and 1, or a string type with the same values 'True' and 'False'. But using a boolean variable type can significantly reduce the amount of code, making it more descriptive.

For example, to check the inequality of the values of the variables a and b, you can write:

a = **1**

b = **2**

What will the system give out if you write a==b?

Compare ARPU

Previously, we calculated revenue per user for Astana (arpu\_astana) and Almaty (arpu\_almaty). How can we now check that arpu\_almaty is greater than arpu\_astana?

The script should return True or False as a result.

* float(arpu\_almaty) / float(arpu\_astana)
* print(arpu\_almaty - arpu\_astana)
* arpu\_almaty > arpu\_astana