

## **Practice 1- Introduction to the python programming language**

### **Brief theory**

Python is an object-oriented, interpreted, portable, ultra-high-level language. Programming in Python allows you to quickly and efficiently get the necessary program modules.

Included with the Python interpreter is IDLE (Integrated Development Environment). At its core, it is similar to an interpreter running in interactive mode with an extended set of features (syntax highlighting, viewing objects, debugging, etc.).

To run IDLE on Windows, you need to go to the Python folder in the Start menu and look for a shortcut named “IDLE (Python 3.X XX-bit)”.

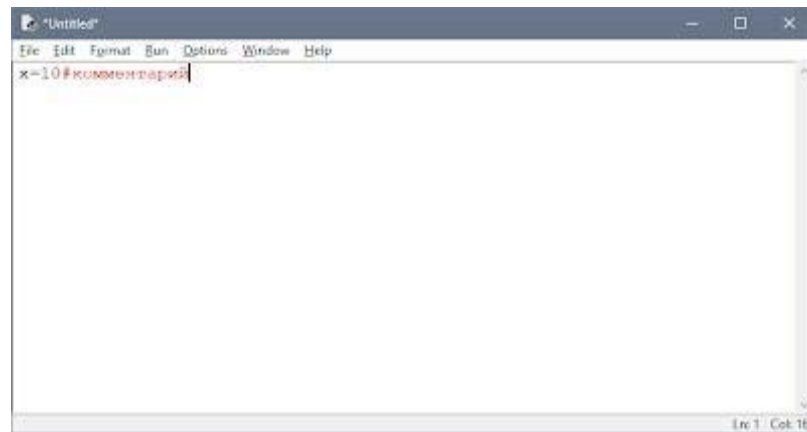
To launch the program (code) editor, execute the File->New File command or the Ctrl+N key combination.

Any Python program consists of a sequence of valid characters, written in a certain order and according to certain rules.

The program includes:

- comments;
- commands;
- punctuation marks;
- identifiers;
- keywords.

Comments in Python are preceded by the # character and continue to the end of the line (i.e., in Python, all comments are single-line), while quotes are not allowed before the # character:



## Punctuation marks

The Python alphabet includes a fair number of punctuation marks that are used for a variety of purposes. For example, the "+" or "\*" signs can be used for addition and multiplication, and the comma "," can be used to separate function parameters.

## Identifiers

Identifiers in Python are names used to refer to a variable, function, class, module, or other object.

## Keywords

Some words have a special meaning in Python and are language control constructs.

Keywords in Python:

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

## Data types

1. None (undefined variable value)
2. Boolean variables (Boolean Type)
3. Numbers (Numeric Type)
  1. int - integer
  2. float - floating point number
  3. complex - complex number
4. Lists (Sequence Type)
  1. list - list
  2. tuple - tuple
  3. range - range

5. Strings (Text Sequence Type)
  1. str

### **Data input and output**

Data input is performed using the input(input list) command: a = input()

```
print(a)
```

In the brackets of the function, you can specify a message - a comment to the input data:

```
a = input("Enter quantity: ")
```

The input() command, by default, accepts input as a string of characters. Therefore, to enter an integer value, you must specify the data type int():

```
a = int(input())
```

To enter real numbers, use the command a=float(input())

Data output is carried out using the command print(output list): a = 1

```
b = 2
```

```
print(a)
```

```
print(a + b)
```

```
print('sum = ', a + b)
```

It is possible to write commands on one line, separating them with ;. However, you should not use this method often, it reduces readability:

```
a = 1; b = 2;
```

```
print(a) print(a+b)
```

```
print('sum = ', a + b)
```

For the print command, a so-called separator can be set - a separator between output elements:

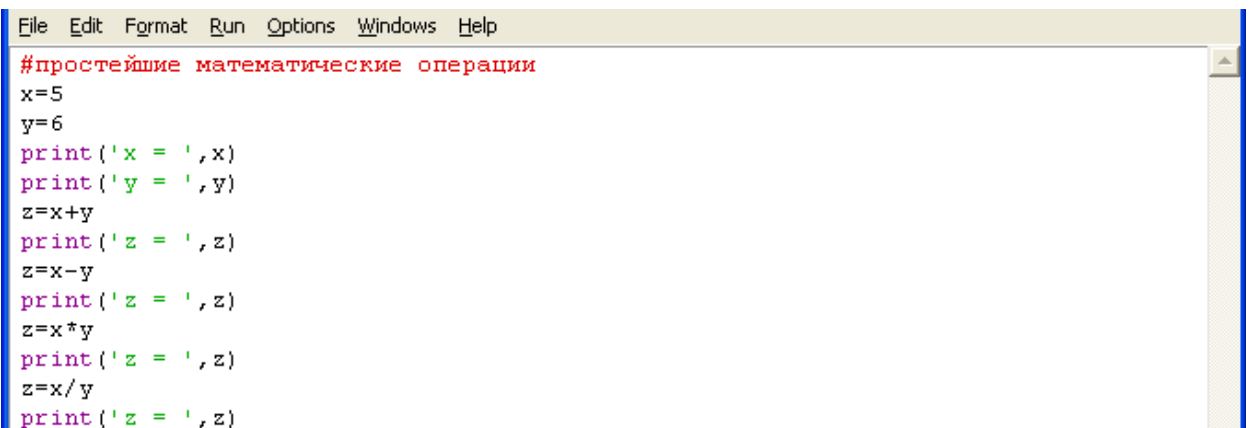
```
x
=2
y=5
```

```
print ( x, "+", y, "=", x+y, sep = " " )
```

The result will be displayed with spaces between elements:  $2 + 5 = 7$

### Simple arithmetic operations on numbers

$x+y$	Addition
$x-y$	Subtraction
$x*y$	Multiplicati on
$x / y$	Division



```
File Edit Format Run Options Windows Help
#простейшие математические операции
x=5
y=6
print('x = ', x)
print('y = ', y)
z=x+y
print('z = ', z)
z=x-y
print('z = ', z)
z=x*y
print('z = ', z)
z=x/y
print('z = ', z)
```

Python Program Example

```
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
x = 5
y = 6
z = 11
z = -1
z = 30
z = 0.8333333333333334
>>> |
```

The result of executing a program using simple arithmetic operations

For formatted output, use format:

The format() string method returns a formatted version of the string, replacing the identifiers in curly braces {}. Identifiers can be positional, numeric indexes, dictionary keys, variable names.

The syntax of the format command is:

fieldsubstitutions:= {" [field name] ["!" conversion] [":" specification] "}"

namefields:= arg\_name ( "." attribute name | "[" index "]" )\*

transformation := "r" (internal representation) | "s" (human representation)

specification:= see below

There can be more arguments to format() than there are identifiers in the string. In this case, the rest are ignored.

Identifiers can be either argument indexes or keys:

```
example_format.py - K:\Лабораторные Python\example_format.py (3.7.1)
File Edit Format Run Options Window Help
x=11
print(x) #вывод без форматирования
print("{:4}".format(x)) #перед значением переменной x будет присутствовать 2 пробела,
                        #так как число 11 занимает 2 знакоместа
```

```

Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: K:\Лабораторные Python\example_format.py =====
11
 11
>>> |

```

As a result, the number 11 will be displayed, and before it there are two spaces, since it is indicated to use four character spaces for output.

Or with multiple arguments:

```

example_format1.py - K:\Лабораторные Python\example_format1.py (3.7.1)
File Edit Format Run Options Window Help
x=2
print ("{:4d}{:4d}{:4d}".format (x,x,x))

```

```

Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: K:\Лабораторные Python\example_format1.py =====
 2  2  2
>>>

```

As a result, each of the values is derived from the calculation of 4 familiarity. Format specification:

<b>specification</b>	<b>:=</b> [[fill]align][sign][#][0][width][,][.precision][type]
aggregate	<b>:=</b> character other than '{' or '}'
alignment	<b>:=</b> "<"   ">"   "="   "^"
sign	<b>:=</b> "+"   "-"   " "
width	<b>:=</b> integer

accuracy	:= integer
type of	:= "b"   "c"   "d"   "e"   "E"   "f"   "F"   "g"   "G"   "n"   "o"   "s"   "x"   "X"   "%"

type of	Mean ing
'd', 'i', 'u'	Decimal number.
'o'	Number in octal number system.
'x'	Number in hexadecimal notation (lowercase letters).
'X'	Number in hexadecimal notation (uppercase letters).
'e'	Floating point number with exponent (lower case exponent).
'E'	Floating point number with exponent (exponent in upper case).
'f', 'F'	Floating point number (normal format).
'g'	Floating point number. with exponent (lower case exponent) if it's less than -4 or precision, otherwise normal format.
'G'	Floating point number. with exponent (exponent in the top case) if it is less than -4 or precision, otherwise normal format.
'c'	Character (string of one character or number - character code).
's'	Line.
'%', ,	The number is multiplied by 100, a floating point number is displayed followed by a % sign.

The following command is used to format real floating point numbers:

```
print('{0:.2f}'.format(real number))
```

```
format_chisla.py - K:/Лабораторные Python/format_chisla.py (3.7.1)
File Edit Format Run Options Window Help
x=10
y=7
print("{0:.2f}".format(x/y))
```

The result will be a number with two decimal places.

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct
1)] on win32
Type "help", "copyright", "credits"
>>>
===== RESTART: K:/Лаборатор
1.43
>>> |
```

### Example

Write a program that asks the user:

Full name ("Your last name, first name")

age ("How old are you?")

place of residence ("Where do you live?")

After that, it would display three lines: "Your name"

"Your age"


"You live in"



 \*1.py - C:/Users/a.tleubayeva/Documents/advanced python

File Edit Format Run Options Window Help

```
a= input('Your last name, first name')
b= input ("How old are you?")
c= input ("Where do you live?")
print('Your name', a)
print ("Your age", b)
print ('Your live in',c)
|
```

 IDLE Shell 3.10.6

File Edit Shell Debug Options Window Help

Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:/Users/a.tleubayeva/Documents/advanced python/1.py =====

Your last name, first name Arailym

How old are you? 27

Where do you live? Nur-Sultan

Your name Arailym

Your age 27

Your live in Nur-Sultan

>>>

|

### Tasks for independent work

Install Python [| Python.org](https://www.python.org)

Write a program that would ask the user: First name, Last name, Age, Location  
last name, first name ("Your last name, first name?")

age ("How old are you?")

place of residence ("Where do you live?") After that, it would display three lines: "Your  
last name, first name"

"Your age" "You live in"