Python programming for beginners

Stefan Zhauryd
Instructor

Module 1

Introduction to Python and computer programming

In this module, you will learn about:

the fundamentals of computer programming, i.e., how the computer works, how the program is executed, how the programming language is defined and constructed;

the difference between compilation and interpretation

what Python is, how it is positioned among other programming languages, and what distinguishes the different versions of Python.

Natural languages vs. programming languages

A language is a means (and a tool) for expressing and recording thoughts. There are many languages all around us. Some of them require neither speaking nor writing, such as body language; it's possible to express your deepest feelings very precisely without saying a word.

Another language you use each day is your mother tongue, which you use to manifest your will and to think about reality. Computers have their own language, too, called machine language, which is very rudimentary.

What makes a language?

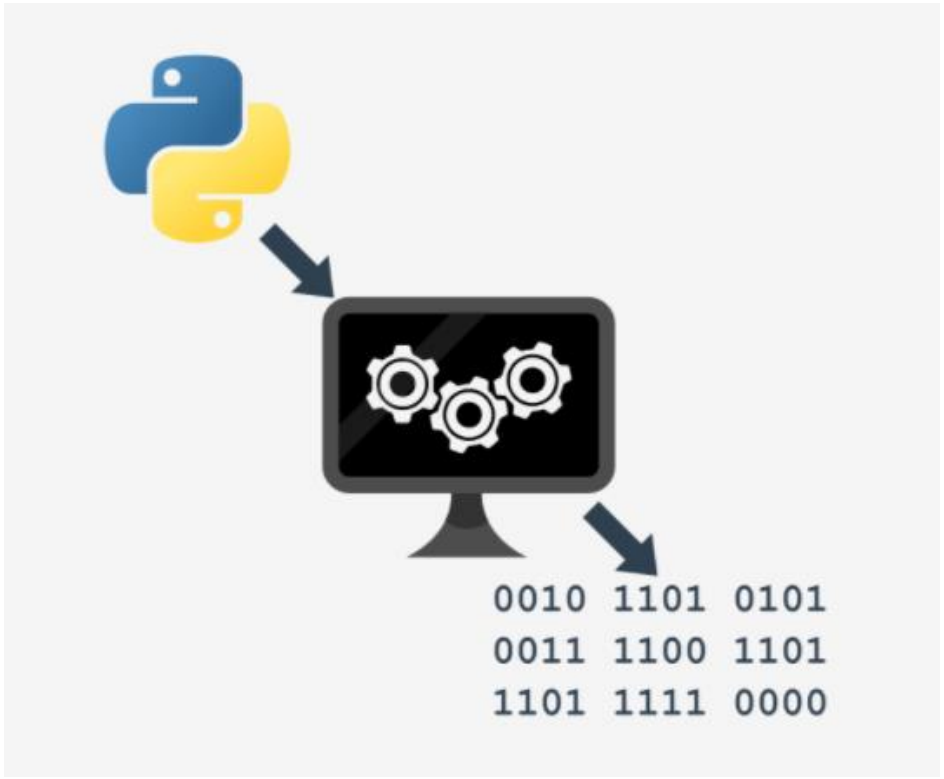
- an alphabet
- a lexis
- a syntax
- semantics

Compilation vs. interpretation

There are two different ways of transforming a program from a high-level programming language into machine language:

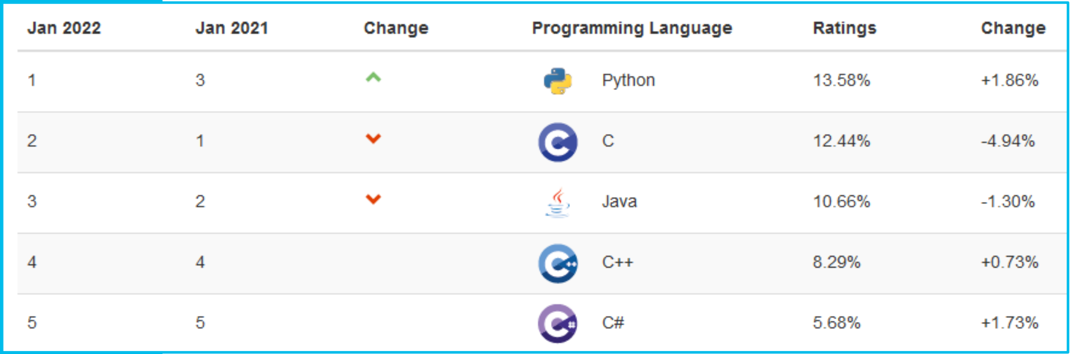
- COMPILATION
- INTERPRETATION

What does the interpreter actually do?



Compilation vs. interpretation - advantages and disadvantages

What is Python? Who created Python?

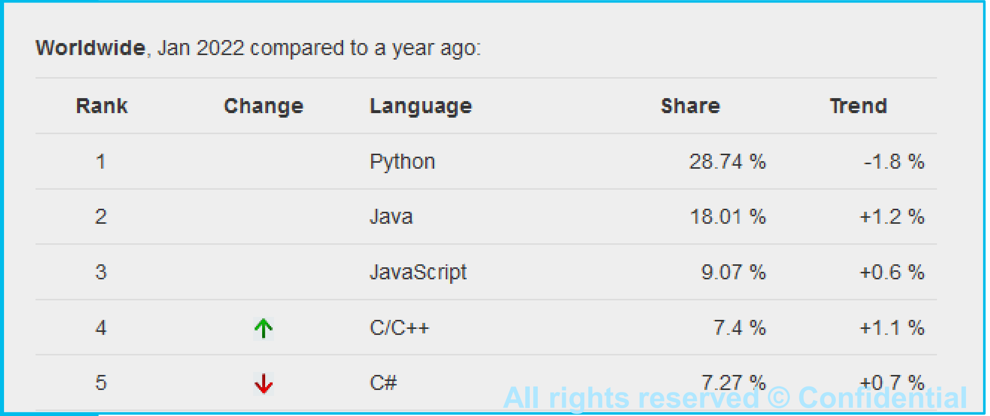


Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	^	Python	13.58%	+1.86%
2	1	v	C	12.44%	-4.94%
3	2	v	Java	10.66%	-1.30%
4	4		C++	8.29%	+0.73%
5	5		C#	5.68%	+1.73%

https://www.tiobe.com/tiobe-index/

Either way, it still occupies a high rank in the top ten PL

https://pypl.github.io/PYPL.html



Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %

What makes Python special?
CPython - std

How to install Инструкция

IDLE:

- **VS Code**

https://code.visualstudio.com/docs/setup/windows

https://www.python.org/downloads/

https://code.visualstudio.com/docs/python/python-tutorial

- PyCharm
- VIM (**как выйти из VIM☺**)
- NANO
- Sublime

and so on

How to use without installation

- Tkinter:

  https://trinket.io/python/41462f0f16

- Cocalc:

  Cocalc.com

- **Official docs:**

  https://docs.python.org/3/

- Useful link:

  https://developers.google.com/edu/python/set-up

Well done!

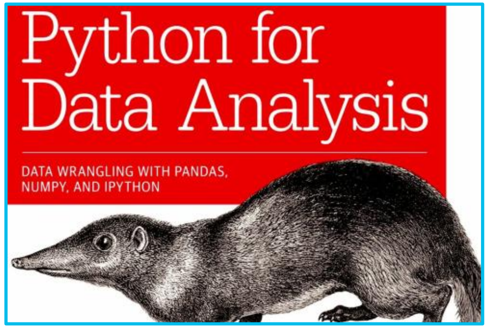
Home work:

- setup python and IDLE on your PC
- read docs
  https://docs.python.org/3

Selenium Python

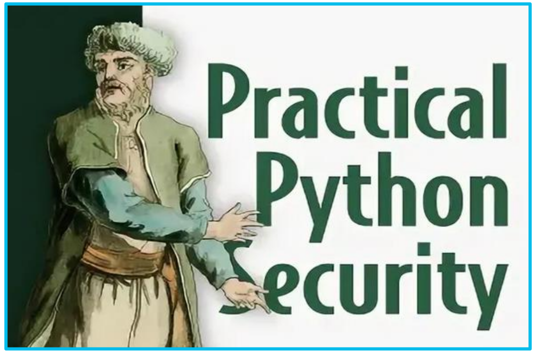
python™ django

PYTHON DATA SCIENCE



Python for Data Analysis

DATA WRANGLING WITH PANDAS, NUMPY, AND IPYTHON



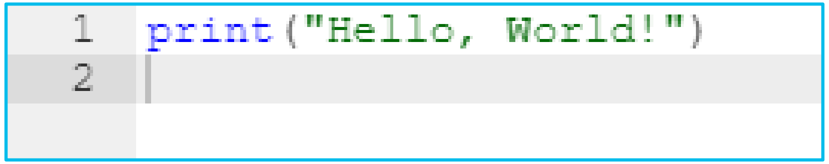
Practical Python Security

- QA
- DATA
- BACK
- Security
- DevOps - DevNet
- and so on (chatbot, export…)

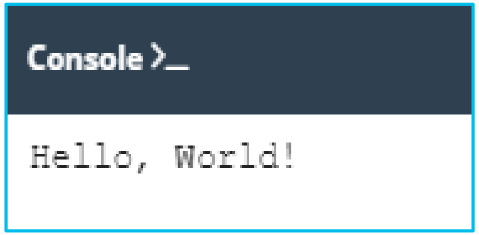
Module 2

Data types, variables, basic input-output operations, basic operators

Hello, World!



```
print("Hello, World!")
```



```
Hello, World!
```

The **print()** function

- effect
- result
- argument(s)

```
function_name(argument)
```

What happens when Python encounters an invocation like this one below?

Home work 1.0
Print

Write the simple program - **hello world**.

The program should print string "Hello world!" or similar.

Use the **print** function.

Home work 1.0

The print() command, which is one of the easiest directives in Python, simply prints out a line to the screen.

In your first lab:

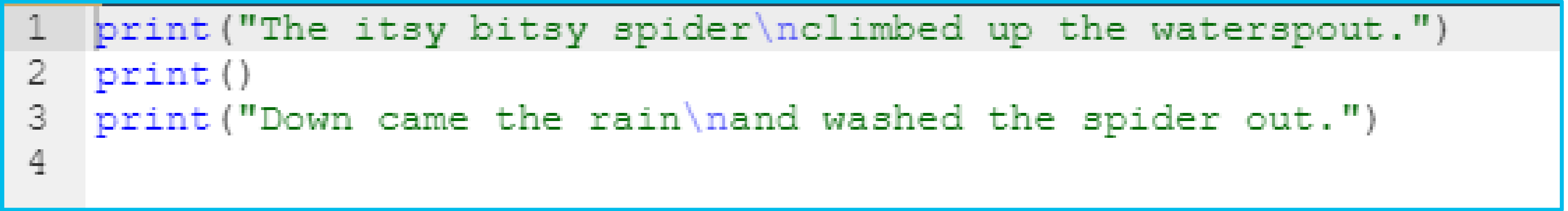
- use the **print()** function to print the line "**Hello, Python!**" to the screen. Use double quotes around the string;
- having done that, use the print() function again, but this time print your first name;
- remove the double quotes and run your code. Watch Python's reaction. What kind of error is thrown?
- then, remove the parentheses, put back the double quotes, and run your code again. What kind of error is thrown this time?
- experiment as much as you can. Change double quotes to single quotes, use multiple print() functions on the same line, and then on different lines. See what happens.

The **print()** function

What is the effect the print() function causes?

What arguments does print() expect?

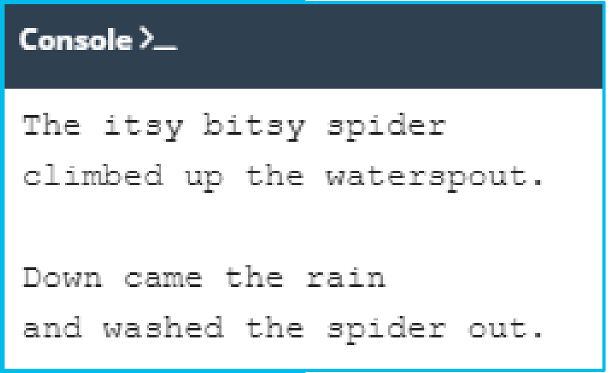
What value does the print() function return?



```
print("The itsy bitsy spider\nclimbed up the waterspout.")
print()
print("Down came the rain\nand washed the spider out.")
```

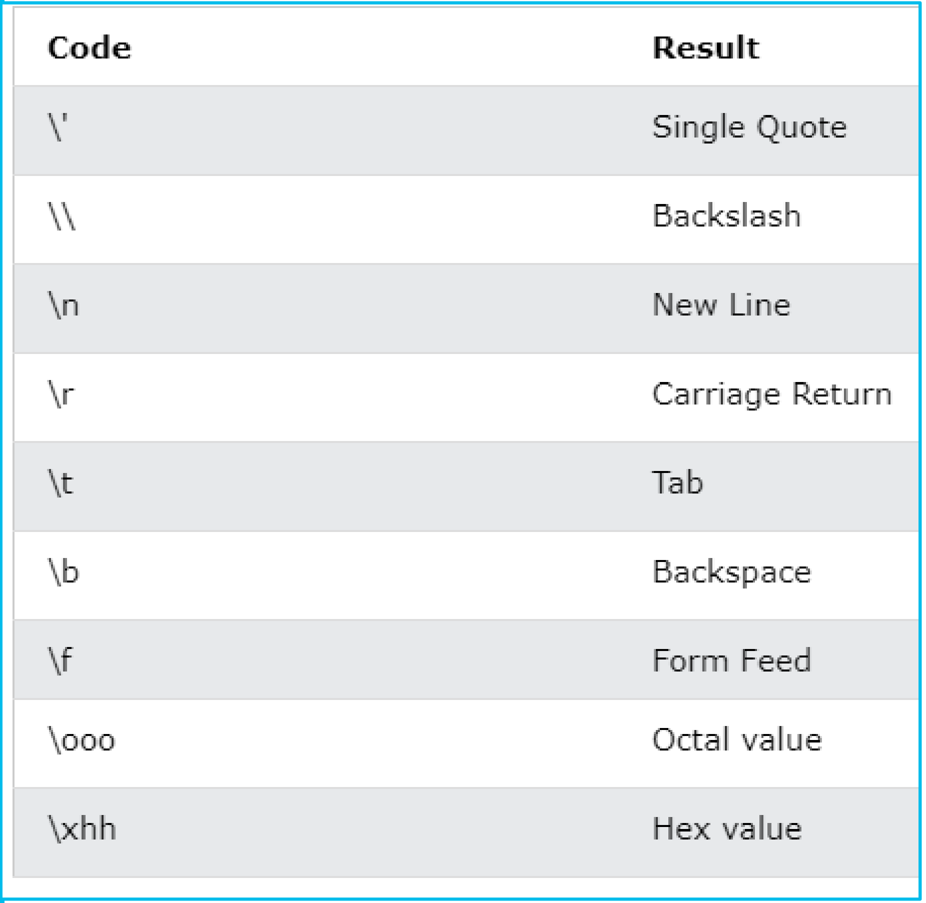
The print() function - the escape and newline characters

The backslash (\) has a very special meaning when used inside strings - this is called the **escape character**.



```
The itsy bitsy spider
climbed up the waterspout.

Down came the rain
and washed the spider out.
```

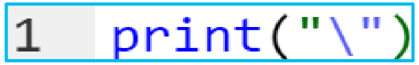
The print() function - the escape and newline characters



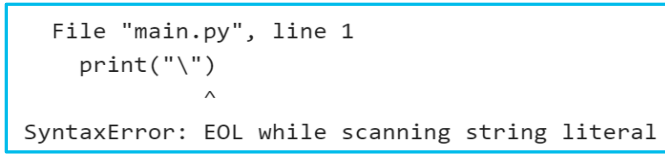
Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

The print() function - the escape and newline characters

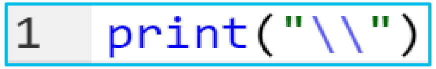
If you want to put just one backslash inside a string, don't forget its escaping nature - you have to double it, e.g., such an invocation will cause an error:

- Example 

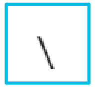
```
print("\")
```

  output 

```
  File "main.py", line 1
    print("\")
             ^
SyntaxError: EOL while scanning string literal
```

- Example 

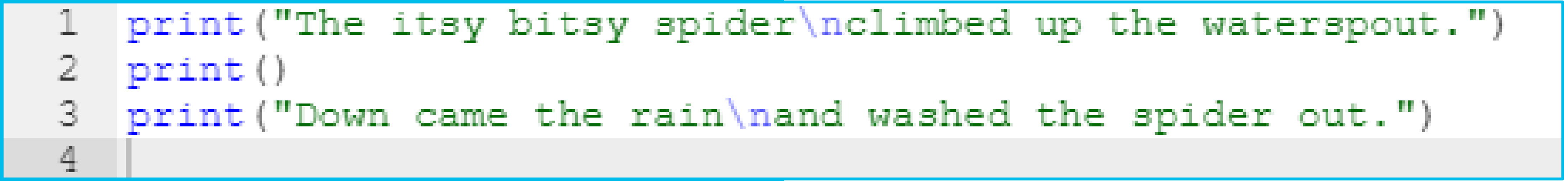
```
print("\\")
```

  - output 

```
\
```

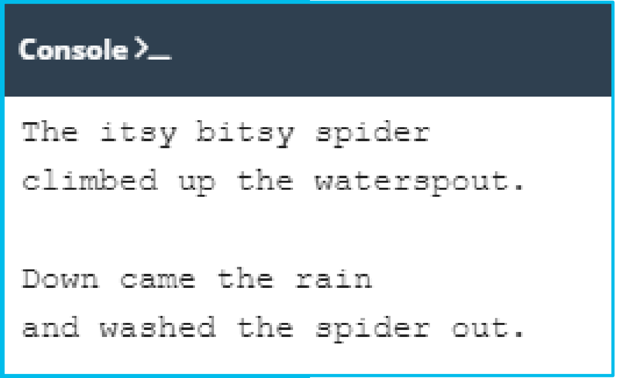
Not all escape pairs (the backslash coupled with another character) mean something.

HW- Experiment with your code in the editor, run it, and see what happens.



```python
print("The itsy bitsy spider\nclimbed up the waterspout.")
print()
print("Down came the rain\nand washed the spider out.")

```

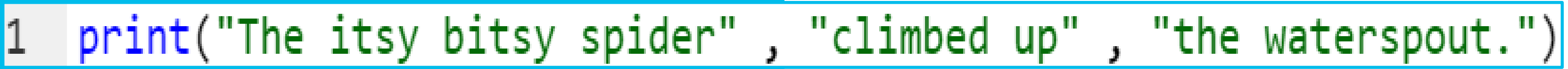
Example



```
The itsy bitsy spider
climbed up the waterspout.

Down came the rain
and washed the spider out.
```

The print() function - using multiple arguments

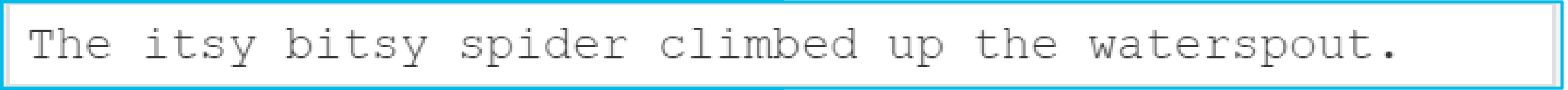
Type the following code in your editor, then run it:



```
print("The itsy bitsy spider" , "climbed up" , "the waterspout.")
```

- There is one print() function invocation, but it **contains three arguments**.
- All of them are strings.
- The arguments are **separated by commas**

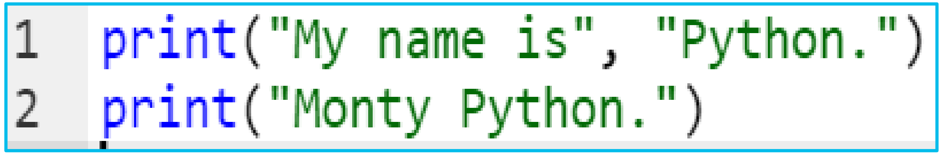
Output:



```
The itsy bitsy spider climbed up the waterspout.
```

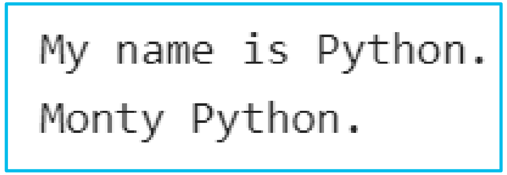
The print() function - the positional way of passing the arguments

- Type the following code in your editor, then run it:



```
print("My name is", "Python.")
print("Monty Python.")
```

- Output:



```
My name is Python.
Monty Python.
```

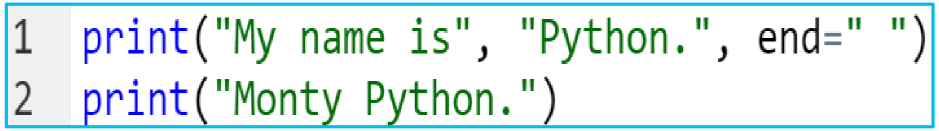
- The way in which we are passing the arguments into the print() function is the most common in Python, and is called the **positional way**

The print() function - the keyword arguments

- The **print() function has two keyword arguments that you can use for your purposes**:
  - **end**
    - determines the characters the print() function sends to the output once it reaches the end of its positional arguments
  - **sep**
    - defines the separator between the arguments or items in an iterator to print
    - Python 3 **sep** defaults to “space” which is known as “**soft spacing**”

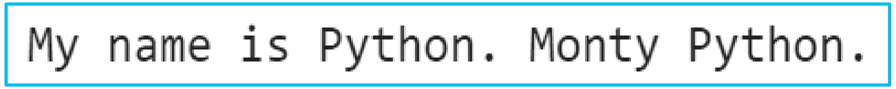
The print() function - the keyword arguments

- Type the following code in your editor, then run it:



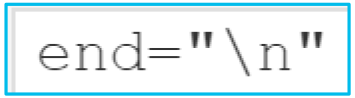
```python
print("My name is", "Python.", end=" ")
print("Monty Python.")
```

Output:



```
My name is Python. Monty Python.
```

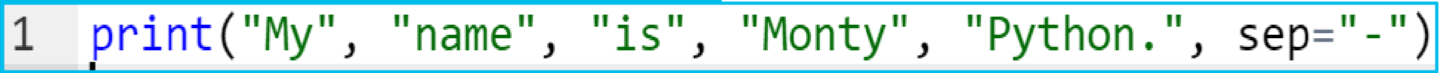
- The default behavior reflects the situation where the **end** keyword argument is implicitly used in the following way:



```
end="\n"
```

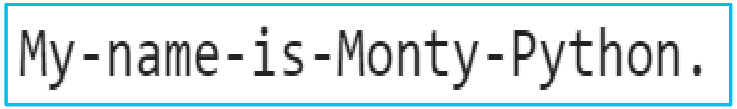
The print() function - the keyword arguments

- Type the following code in your editor, then run it:



```python
print("My", "name", "is", "Monty", "Python.", sep="-")
```

- Output:



```
My-name-is-Monty-Python.
```

The print() function - the keyword arguments

- Both keyword arguments may be mixed in one invocation, just like here in the editor window.
- Type the following code in your editor, then run it:

```
print("My", "name", "is", sep="_", end="*")
print("Monty", "Python.", sep="*", end="*\n")
```

Output:

```
My_name_is*Monty*Python.*
```

Home work 1.1
The print() function

- Modify the first line of code in the editor, using the **sep** and **end** keywords, to match the expected output. Use the two **print()** functions in the editor.

- Don't change anything in the second print() invocation.
- Expected output:

```
Programming***Essentials***in...Python
```

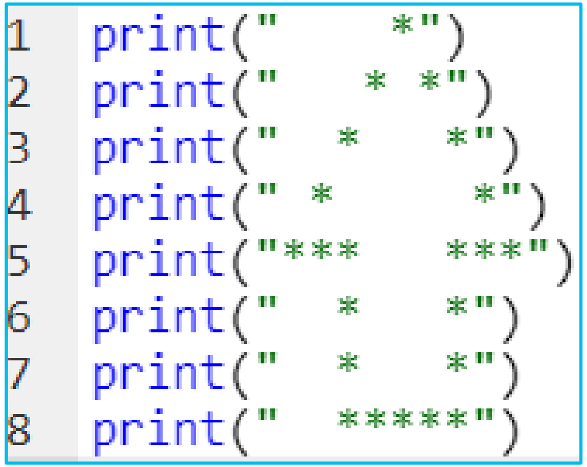
Home work 1.2

The print() function

sep, end

I strongly encourage you to play with the code we've written for you, and make some (maybe even destructive) amendments.

Feel free to modify any part of the code, but there is one condition - learn from your mistakes and draw your own conclusions.



```
print("    *")
print("   * *")
print("  *   *")
print(" *     *")
print("***   ***")
print("  *   *")
print("  *   *")
print("  *****")
```

Literals - the data in itself

A literal is data whose values are determined by the literal itself.

Literals - the data in itself

- Example:
  - 123 ➔ literal (integer literal)
  - C ➔ not literal
  - “2444” ➔ literal (string literal)
  - 2.5 ➔ literal (float literal)

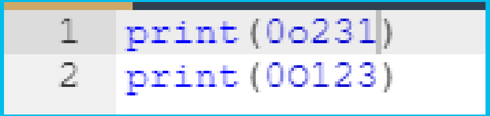
Integers

- a number that can be written without a fractional component

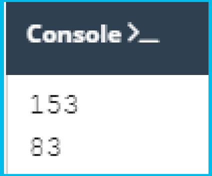
Example:

- print(123), output: 123
- print(11_111), output: 11111
- print(+75000), output: 75000

Integers: **octal** and **hex**adecimal numbers



```
print(0o231)
print(0O123)
```



```
153
83
```

Octal

- an integer number is preceded by an 0O or 0o prefix (zero-o)
- This means that the number must contain digits taken from the [0..7] range only.
- Example:
  - 0o123 is an octal number with a (decimal) value equal to 83

Hexadecimal

- An integer numbers is be preceded by the prefix 0x or 0X (zero-x)
- Example:
  - 0x123 is a hexadecimal number with a (decimal) value equal to 291
  - print(0x123), output: 291

Floats

- The numbers that have (or may have) a fractional part after the decimal point
- Example:
  - print(2.5), output: 2.5
  - print(-0.4), output: -0.4
  - print(.4), output: 0.4
  - print(4.), output: 4.0

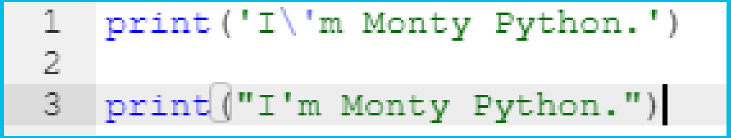
You can use **scientific notation**

- When you want to use any numbers that are very large or very small
- for example, the speed of light, expressed in meters per second. Written directly it would look like this: 300000000.
- To avoid writing out so many zeros, physics textbooks use an abbreviated form, which you have probably already seen: 3×10^8.
- In Python, the same effect is achieved in a slightly different way - take a look: **3E8 or 3e8**
- Note:
  - the exponent (the value after the E) has to be an integer;
  - the base (the value in front of the E) may be an integer.

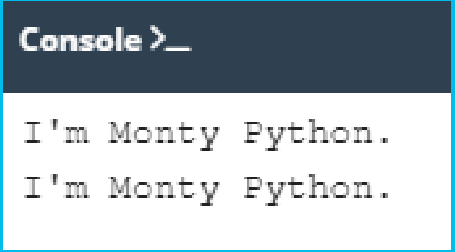
You can use **scientific notation**

- A physical constant called Planck's constant (and denoted as h), according to the textbooks, has the value of: 6.62607 x 10^{-34}.
- If you would like to use it in a program, you should write it this way: 6.62607E-34
- Note: the fact that you've chosen one of the possible forms of coding float values doesn't mean that Python will present it the same way.
- Python may sometimes choose different notation than you.
- For example, let's say you've decided to use the following float literal: **0.0000000000000000000001**
- When you run this literal through Python: **print(0.0000000000000000000001)**
- this is the result: **1e-22**

Strings



```
print('I\'m Monty Python.')

print("I'm Monty Python.")
```



```
Console >_
I'm Monty Python.
I'm Monty Python.
```

- Strings are used when you need to process text (like names of all kinds, addresses, novels, etc.), not numbers.
- Example:
  - print("Hello")
  - print('Hello')
  - print(

```
    """
+================================+
| Welcome to my game, muggle!      |
| Enter an integer number          |
| and guess what number I've       |
| picked for you.                  |
| So, what is the secret number?   |
+================================+
    """)
```

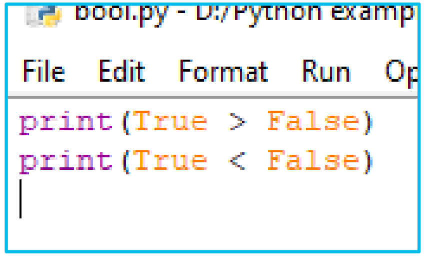
  - print("I like \"Monty Python\"")

    output: I like "Monty Python“
  - print('I like "Monty Python"')

    output: I like "Monty Python“

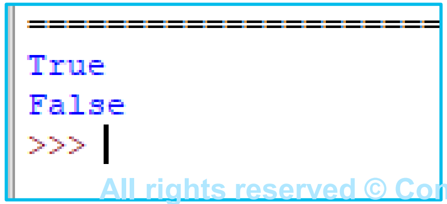
Boolean Value

- These two Boolean values have strict denotations in Python:
  - **True**
  - **False**



```
print(True > False)
print(True < False)
```

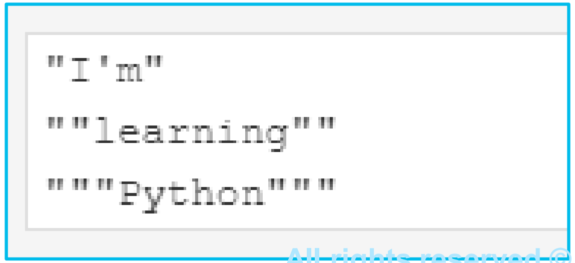
output:



```
======================
True
False
>>>
```

Home work 1.3

- Write a one/tree-line piece of code, using the print() function, as well as the newline and escape characters, to match the expected result outputted on three lines.
- Expected output:



```
"I'm"
""learning""
"""Python"""
```

Key takeaways

Literals are notations for representing some fixed values in code. Python has various types of literals - for example, a literal can be a number (numeric literals, e.g., 123), or a string (string literals, e.g., "I am a literal.").

The binary system is a system of numbers that employs 2 as the base. Therefore, a binary number is made up of 0s and 1s only, e.g., 1010 is 10 in decimal.

Octal and hexadecimal numeration systems, similarly, employ 8 and 16 as their bases respectively. The hexadecimal system uses the decimal numbers and six extra letters.

Integers (or simply ints) are one of the numerical types supported by Python. They are numbers written without a fractional component, e.g., 256, or -1 (negative integers).

Floating-point numbers (or simply floats) are another one of the numerical types supported by Python. They are numbers that contain (or are able to contain) a fractional component, e.g., 1.27.

To encode an apostrophe or a quote inside a string you can either use the escape character, e.g., 'I\'m happy.', or open and close the string using an opposite set of symbols to the ones you wish to encode, e.g., "I'm happy." to encode an apostrophe, and 'He said "Python", not "typhoon"' to encode a (double) quote.

Boolean values are the two constant objects True and False used to represent truth values (in numeric contexts 1 is True, while 0 is False.

What are **var**iables?

What does every Python variable have?

- **a name;**
- **a value (the content of the container)**

https://pythonchik.ru/osnovy/peremennye-v-python

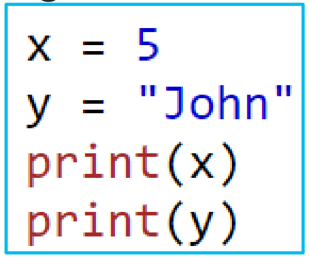
Correct and incorrect variable names

- variable names should be **lowercase, with words separated by underscores to improve readability** (e.g., var, my_variable)
- **function names follow the same convention as variable names** (e.g., fun, my_function)
- it's also possible to use mixed case (e.g., myVariable), but only in contexts where that's already the prevailing style, to retain backwards compatibility with the adopted convention.

Variables

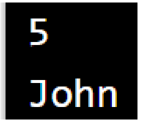
Creating Variables

- Variables are containers for storing data values.
- Unlike other programming languages, Python has no command for declaring a variable.
- A variable is created the moment you first assign a value to it.
- Example:



```
x = 5
y = "John"
print(x)
print(y)
```

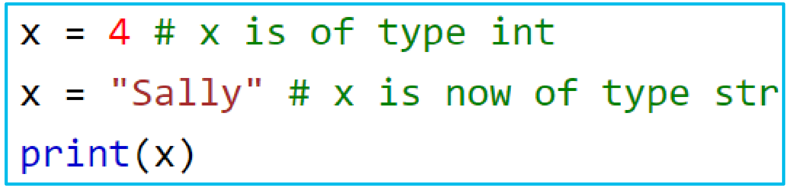
- Output:



```
5
John
```

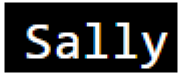
Using variables

- Variables do not need to be declared with any particular type and can even change type after they have been set, example:



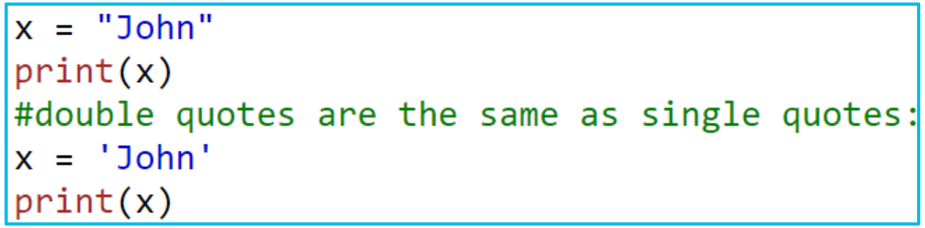
```python
x = 4 # x is of type int
x = "Sally" # x is now of type str
print(x)
```

- Output:



```
Sally
```

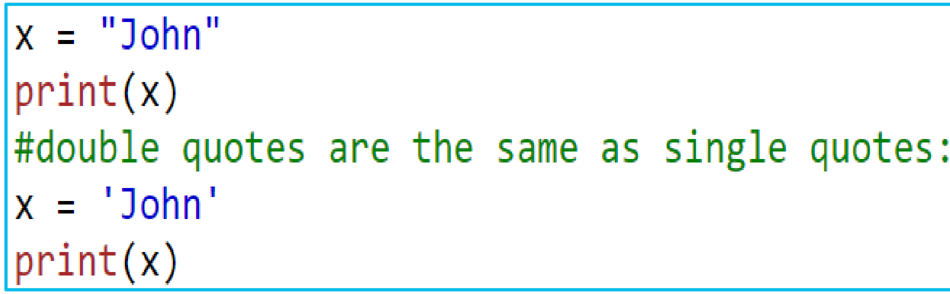
- String variables can be declared either by using single or double quotes, example:



```python
x = "John"
print(x)
#double quotes are the same as single quotes:
x = 'John'
print(x)
```

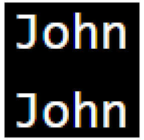
Using variables

- String variables can be declared either by using single or double quotes, example:



```
x = "John"
print(x)
#double quotes are the same as single quotes:
x = 'John'
print(x)
```

- Output:



```
John
John
```

```
x = 12
x, y = 10, "44"

print(x, y)
```

Naming

MyVariable, i, t34, Exchange_Rate, counter, days_to_christmas, TheNameIsSoLongThatYouWillMakeMistakesWithIt, _.

- **A variable can have a short name (like x and y) or a more descriptive name (age, carName, total_volume). Rules for Python variables:**
  - A variable name **must start with a letter or the** underscore character
  - A variable name **cannot start with a number**
  - A variable name can **only contain alpha-numeric characters and underscores** (A-z, 0-9, and _)
  - Variable names **are case-sensitive** (age, Age and AGE are three different variables)

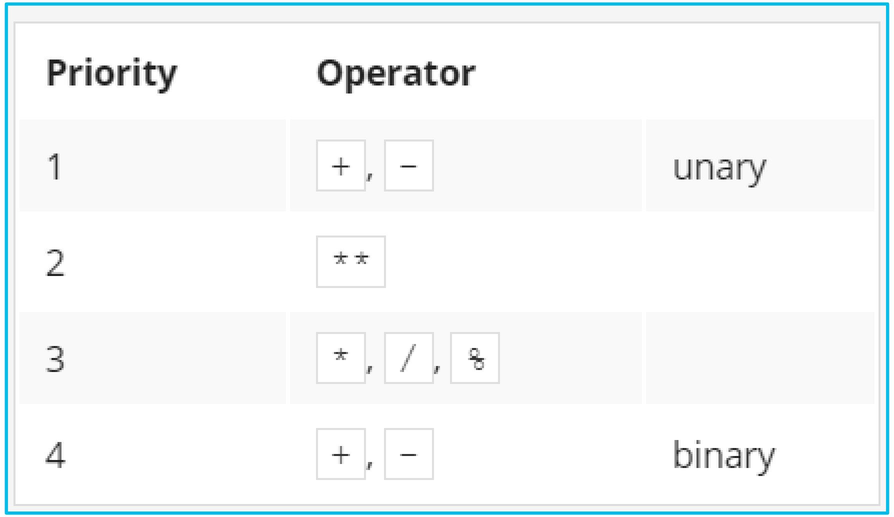
Arithmetic Operators

Operator	Name	Example	Output
+	Addition	print(2 + 2)	4
-	Subtraction	print(5 – 2)	3
*	Multiplication	print(5 * 2)	10
/	Division	print(6 / 2)	3.0
%	Modulus	print(7 % 2)	1
**	Exponentiation	print(2 ** 3)	8
//	Floor division	print(7 // 2)	3

Arithmetic Operators

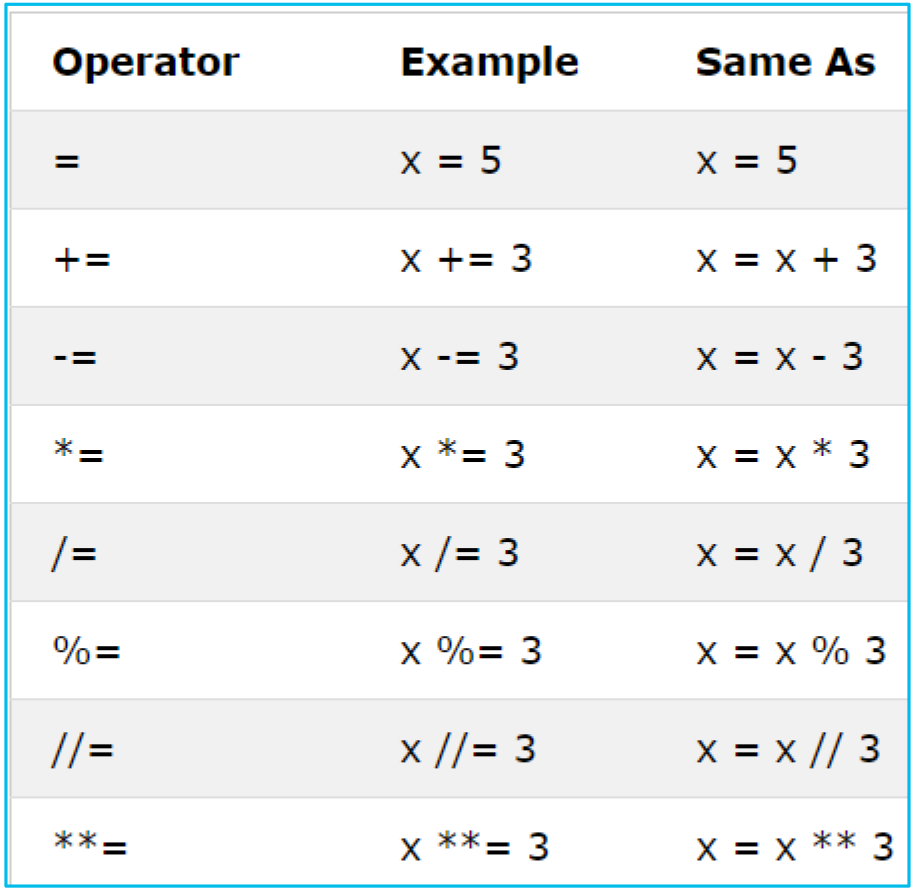
-1
+2

2+2
2-2



Priority	Operator	
1	`+`, `-`	unary
2	`**`	
3	`*`, `/`, `%`	
4	`+`, `-`	binary

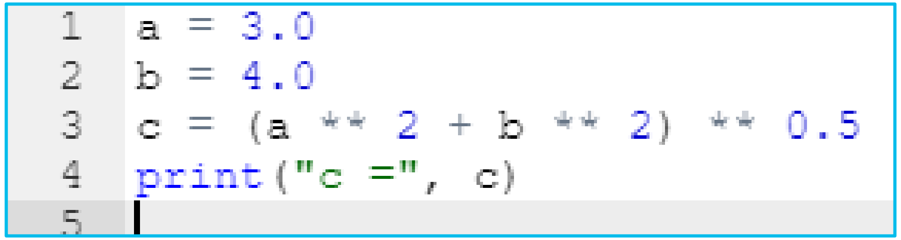
Arithmetic Operators



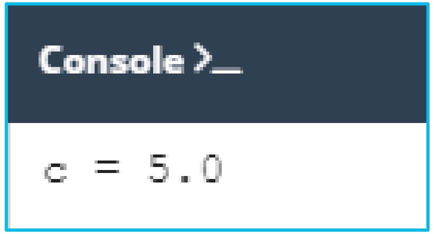
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Solving simple mathematical problems

$$c = \sqrt{a^2 + b^2}$$



```python
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5
print("c =", c)
```



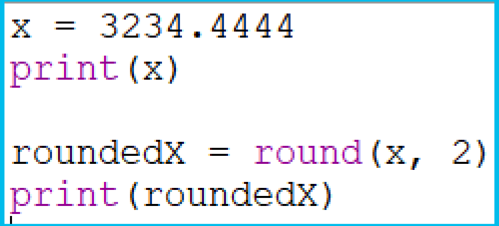
```
c = 5.0
```

Home work 1.4

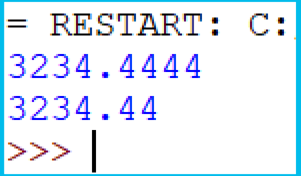
Your task is to:

- create the variables: **john, mary**, and **adam**;
- **assign values to the variables (put an integer in the variables :)).** The values must be equal to the numbers of fruit possessed by John, Mary, and Adam respectively;
- having stored the numbers in the variables, print the variables on one line, and **sep**arate each of them with a **comma**;
- now create a new variable named **total_apples** equal to **addition of the three former variables**.
- **print the value stored** in **total_apples** to the console;
- experiment with your code: create new variables, assign different values to them, and perform various arithmetic operations on them (e.g., +, -, *, /, //, etc.). Try to print a string and an integer together on one line, e.g., "Total number of apples:" and total_apples.

Home work 1.5
total = kilo + miles



```
x = 3234.4444
print(x)

roundedX = round(x, 2)
print(roundedX)
```

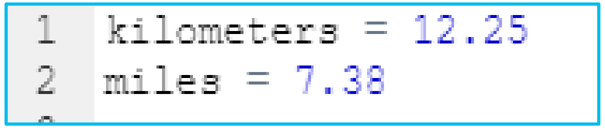


```
= RESTART: C:
3234.4444
3234.44
>>>
```

Miles and kilometers are units of length or distance.

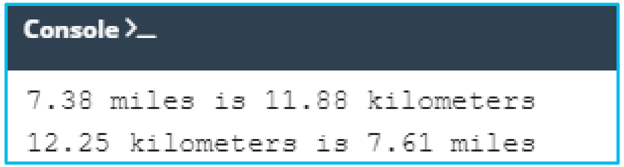
Bearing in mind that 1 mile is equal to approximately 1.61 kilometers, complete the program in the editor so that it converts:

- miles to kilometers;
- kilometers to miles.
- Input data:



```
kilometers = 12.25
miles = 7.38
```

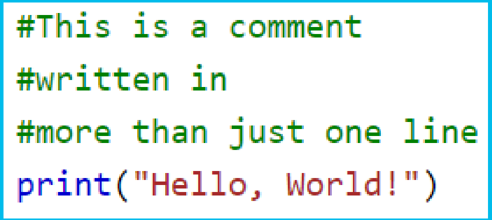
- Output:



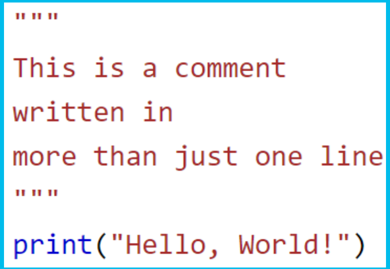
Console >_

```
7.38 miles is 11.88 kilometers
12.25 kilometers is 7.61 miles
```

- ****Use the **round(x, 2)** function to round*
- Just investigate^_^

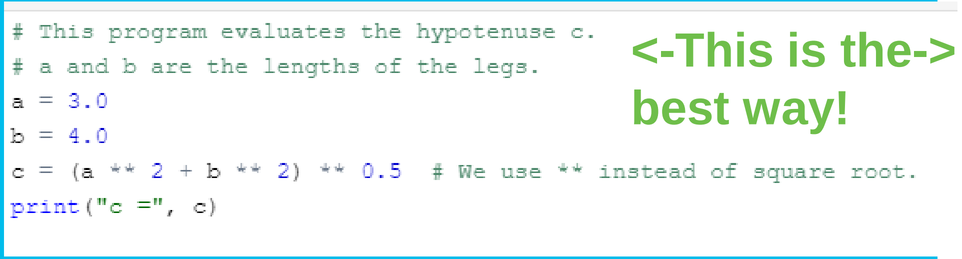


```python
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```



```python
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

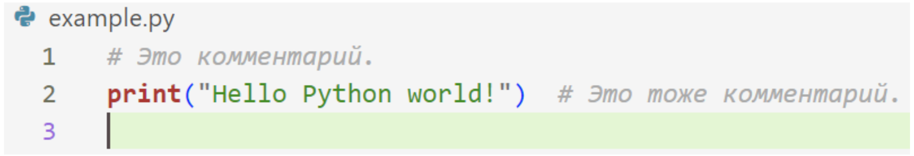
Leaving comments in code: why, how, and when



```python
# This program evaluates the hypotenuse c.
# a and b are the lengths of the legs.
a = 3.0
b = 4.0
c = (a ** 2 + b ** 2) ** 0.5  # We use ** instead of square root.
print("c =", c)
```

<-This is the-> best way!

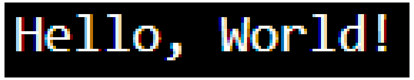
- Comments **can be used to explain Python code**.
- Comments **can be used to make the code more readable**.
- Comments **can be used to prevent execution when testing code**.
- Comments starts with a #, and Python will ignore them, example:



example.py

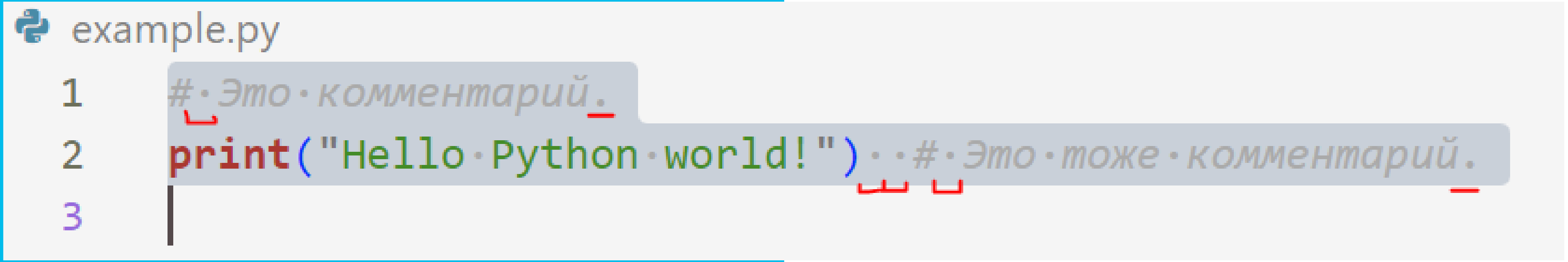
```python
# Это комментарий.
print("Hello Python world!")  # Это тоже комментарий.

```

- Output:



```
Hello, World!
```

Leaving comments in code: why, how, and when



example.py

```
# Это комментарий.
print("Hello Python world!")  # Это тоже комментарий.

```

Home work 1.6

READ this --->
use dots, spaces, comments style, uncomment that you want.

```
# this program computes the number of seconds in a given number of hours.

# this program has been written two days ago

a = 2 # number of hours

seconds = 3600  # number of seconds in 1 hour

print("Hours: ", a) #printing the number of hours

# print("Seconds in Hours: ", a * seconds) # printing the number of seconds in a given number of hours

#here we should also print "Goodbye", but a programmer didn't have time to write any code

#this is the end of the program that computes the number of seconds in 3 hour
```

ЗАДАНИЯ

1) Прорешать всю классную работу
2) Выполнить все домашние задания

Почитать:

1) Byte of Python - книга в нашем telegram канале

Крайний срок сдачи 26/09 в 21:00 (можно раньше, но не позже)

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:
Vasia_Pupkin_class_work_L1.py

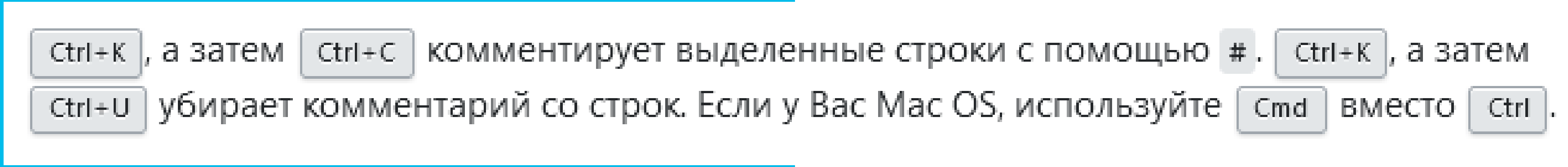
+все задания ОДНИМ ФАЙЛОМ - Vasia_Pupkin_L1.py

Формат сообщения которое вы присылаете мне
(после полного выполнения домашнего задания, только один раз) в Telegram:
Добрый день/вечер. Я Вася Пупкин, и это мои домашние задания к лекции 1.
И отправляете файл/-лы

Крайний срок сдачи 26/09 в 21:00 (можно раньше, но не позже)

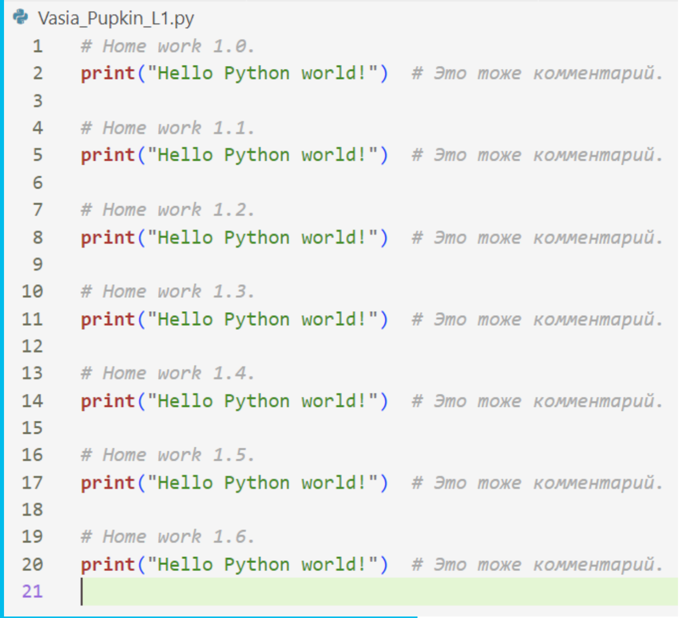
Leaving comments in code: why, how, and when – VS Code

https://ru.stackoverflow.com/questions/953830/Комментарий-на-несколько-строк-visual-studio-code



Ctrl+K, а затем Ctrl+C комментирует выделенные строки с помощью #. Ctrl+K, а затем Ctrl+U убирает комментарий со строк. Если у Вас Mac OS, используйте Cmd вместо Ctrl.

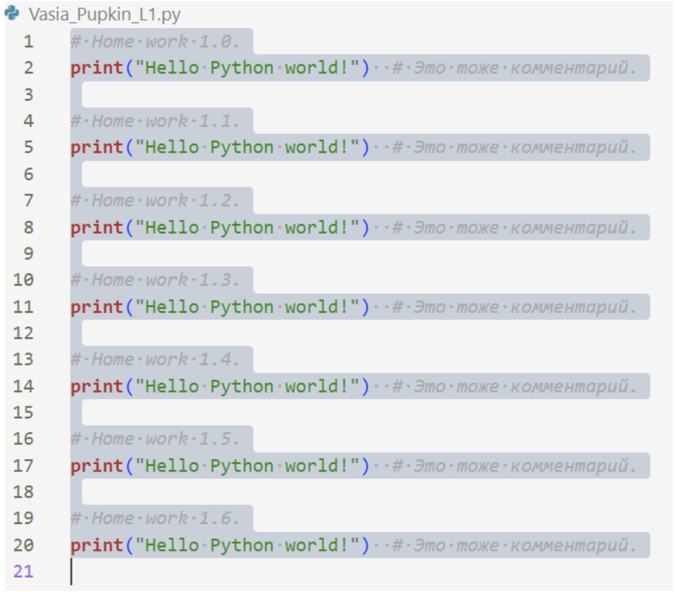
Step 1
Ctrl+A



Vasia_Pupkin_L1.py

```python
# Home work 1.0.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.1.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.2.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.3.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.4.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.5.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.6.
print("Hello Python world!")  # Это тоже комментарий.

```

Result

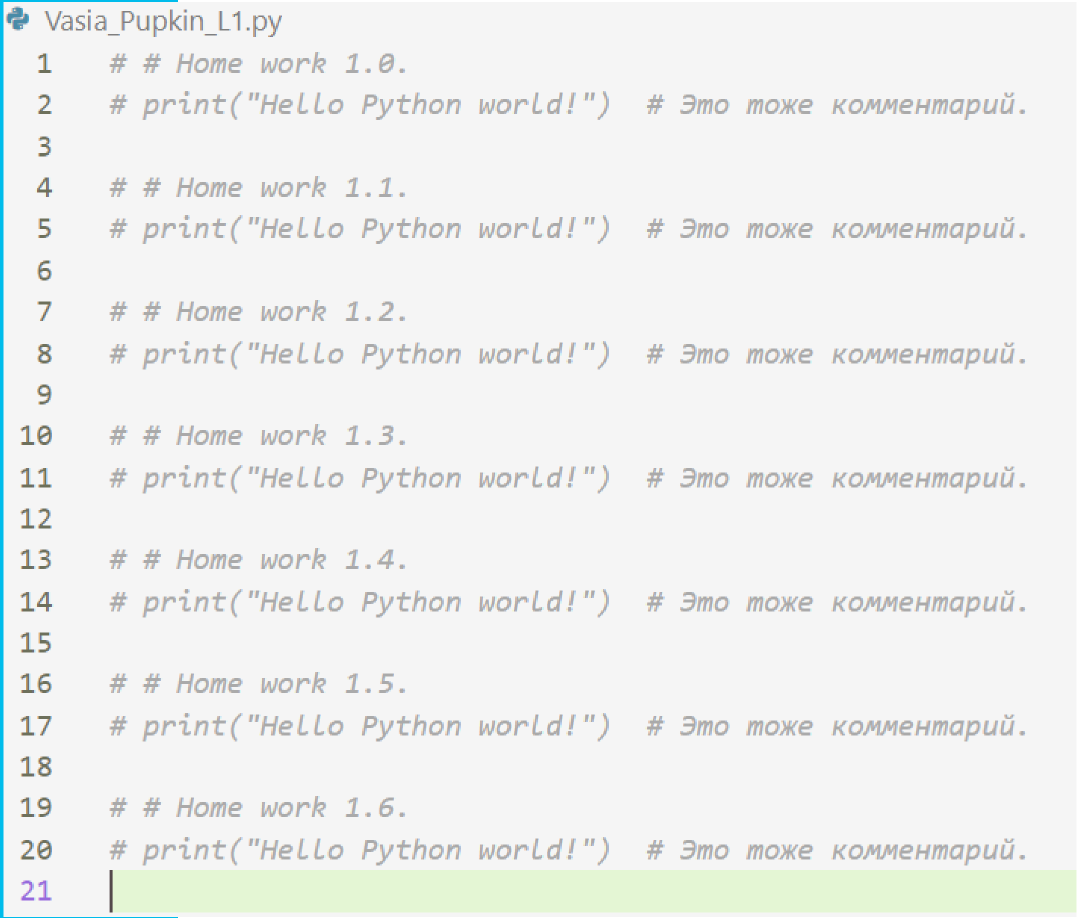


Vasia_Pupkin_L1.py

```python
# Home work 1.0.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.1.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.2.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.3.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.4.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.5.
print("Hello Python world!")  # Это тоже комментарий.

# Home work 1.6.
print("Hello Python world!")  # Это тоже комментарий.

```

Step 2 - Ctrl+K Ctrl+C

Result

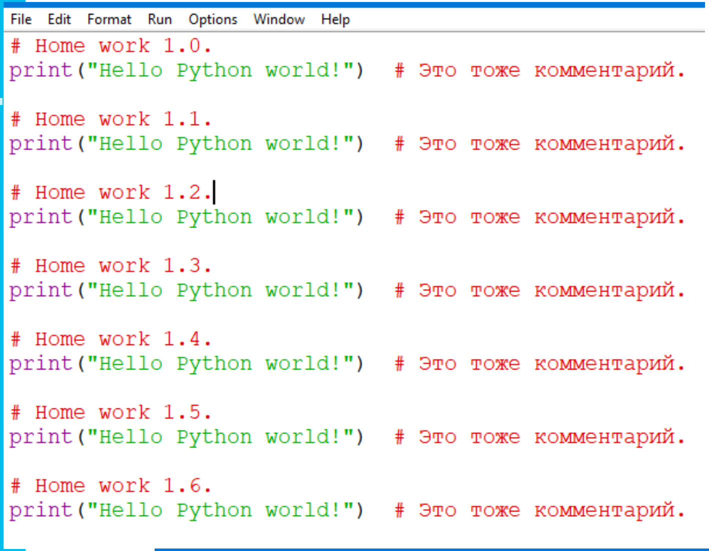


Vasia_Pupkin_L1.py

```
# # Home work 1.0.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.1.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.2.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.3.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.4.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.5.
# print("Hello Python world!")  # Это може комментарий.

# # Home work 1.6.
# print("Hello Python world!")  # Это може комментарий.

```

Step 1 – Ctrl+A

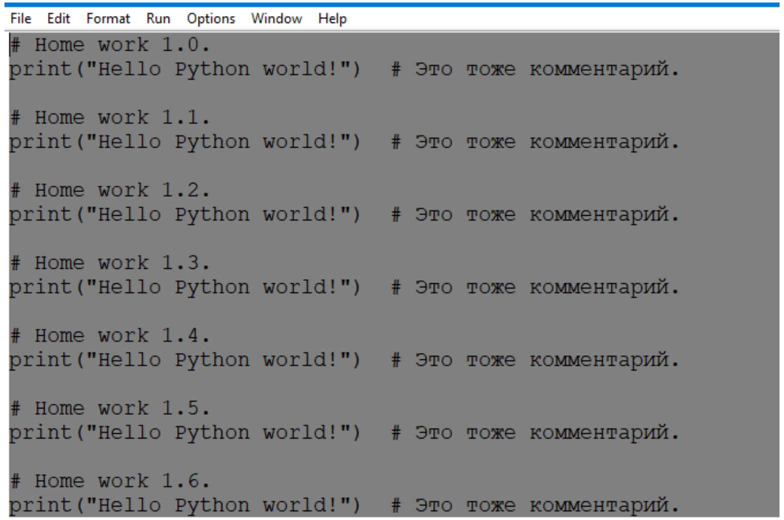
Leaving comments in code: why, how, and when – Python IDLE



```
# Home work 1.0.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.1.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.2.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.3.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.4.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.5.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.6.
print("Hello Python world!")   # Это тоже комментарий.
```

Comment – ALT+3
Uncomment – ALT+4

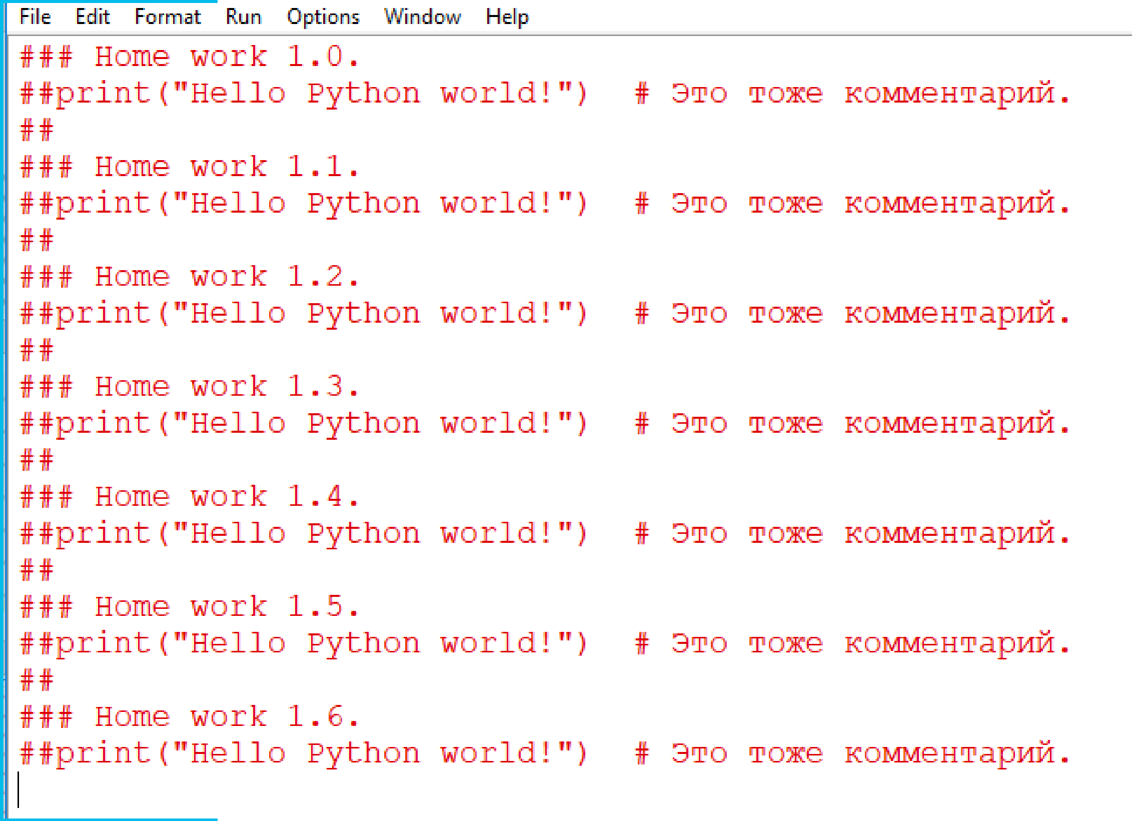
Result



```
# Home work 1.0.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.1.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.2.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.3.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.4.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.5.
print("Hello Python world!")   # Это тоже комментарий.

# Home work 1.6.
print("Hello Python world!")   # Это тоже комментарий.
```

Step 2 - Alt+3

Result



```
### Home work 1.0.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.1.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.2.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.3.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.4.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.5.
##print("Hello Python world!")   # Это тоже комментарий.
##
### Home work 1.6.
##print("Hello Python world!")   # Это тоже комментарий.
```

Q&A

Create your
possibilities.
Bye bye.