Python Programming

Book 3

Table of Contents

What is Python Programming Language....5
1) High-level....5
2) General-purpose....5
3) Interpreted....5
Why Python....6
Python versions....6
Summary....7
Creating a new Python project....7
What is a function....8
Summary....8
Whitespace and indentation....9
Comments....10
Python inline comments....10
Python docstrings....10
One-line docstrings....10
Multi-line docstrings....10
Summary....11
Python Variables....11
What is a variable in Python....11
Creating variables....12
An Example....12
Naming variables....12
Summary....13
Python Data Types....14
Python String....15
Introduction to Python string....15
Summary....17
Integers....17
Python int type....18
Floats....18
Arithmetic Operators....19
Here is a simple example to Find Simple Interest....19
Assignment....20
Python Boolean....20
Introduction to Python Boolean data type....20
Example....21
Summary....21
Python Lists....22
What is a List....22
Accessing elements in a list....22
Summary....24
Python Tuples....25
Introduction to Python tuples....25
Summary....26
Python Dictionary....26

Introduction to the Python Dictionary type....26
Summary....27
Control flow....28
Relational/Comparison Operators....28
Logical Operators....28
We start with Conditional Statements....29
1. Python if statement....30
Example 1: Python if Statement....31
2. Python if...else Statement....31
Example 2. Python if...else Statement....33
3. Python if...elif...else Statement....33
Example 3: Python if...elif...else Statement....35
Students marks Example....36
Assignment....36
Python Nested if statements....37
Nested if Statements....37
Python Loop....38
1. for loop....38
2. while loop....38
Python for Loop....38
Flowchart of Python for Loop....39
Example using range(n)....40
Specifying the starting value for the sequence....41
Specifying the increment for the sequence....42
Code Explanation....42
Example: Loop Over Python List....43
Summary....43
Python while Loop....44
Python while Loop....44
Flowchart of Python while Loop....45
Example: Python while Loop....45
Python Functions....47
What is a function....47
Why do you need functions in Python....47
Defining a Python function....48
1) Function definition....48
2) Function body....48
3) Calling a function....49
Passing information to Python functions....49
Try out this example on VS code....50
Try out this example on VS code....51
Python Modules....52
What is Python Module....52
Create a simple Python module....52
Import Module in Python....53
Syntax of Python Import....53
Importing modules in Python....54

The from-import Statement in Python....54
Importing specific attributes from the module....55
Summary....55
Python Exception Handling....56
Difference between Syntax Error and Exceptions....56
Try and Except Statement – Catching Exceptions....57
Above we put a Try on Line 5, to protect an exception being triggered on Line 6, If an exception is thrown its caught on Line 9. If an exception is not thrown then Line 7 Runs....58
Summary....58
Appendix 1....59
Questions....60
7. Create a List of town and reverse it....60
References....62

What is Python Programming Language

Python is a **high-level, general-purpose, interpreted** programming language.

1) High-level

Python is a high-level programming language that makes it easy to learn. It uses natural language I.e English making it easier to write programs Python doesn’t require you to understand the details of the computer in order to develop programs efficiently.

2) General-purpose

Python is a general-purpose language. It means that you can use Python in various domains including:

- Web applications
- Big data applications
- Testing
- Automation
- Data science, machine learning, and AI
- Desktop software
- Mobile apps

The targeted language like SQL which can be used for querying data from relational databases.

3) Interpreted

Python is an interpreted language. To develop a Python program, you write Python code into a file called source code.

To execute the source code, you need to convert it to the machine language that the computer can understand. And the Python **interpreter** turns the source code, line by line, once at a time, into the machine code when the Python program executes.

Compiled languages like Java and C# use a **compiler** that compiles the whole source code before the program executes.

Why Python

Python increases your productivity. Python allows you to solve complex problems in less time and fewer lines of code. It’s quick to make a prototype in Python.

Python becomes a solution in many areas across industries, from web applications to data science and machine learning.

Python is quite easy to learn in comparison with other programming languages. Python syntax is clear and beautiful.

Python has a large ecosystem that includes lots of libraries and frameworks.

Python is cross-platform. Python programs can run on Windows, Linux, and macOS.

Python has a huge community. Whenever you get stuck, you can get help from an active community.

Python developers are in high demand.

Python versions

Python has two major versions: 2x and 3x.

Python 2.x was released in 2000. The latest version is 2.7 released in 2010. It isn't recommended for use in new projects.

Python 3.x was released in 2008. Basically, Python 3 isn't compatible with Python 2. And you should use the latest versions of Python 3 for your new projects

Summary

- Python is an interpreted, high-level, general-purpose programming language.
- Python becomes the solution in many domains from web applications, data analysis, data science, machine learning, and AI.
- Use Python 3 for the new development.

Creating a new Python project

First, create a new folder called **ClassNameYourNamePython**

Second, launch the VS code and open the , **ClassNameYourNamePython** folder.

Third, create a new Lesson1a.py file and enter the following code and save the file:

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

The **print()** is a built-in function that displays a message on the screen. In this example, it'll show the message **'Hello, Word!'.**

What is a function

When you sum two numbers, that’s a function. And when you multiply two numbers, that’s also a function.

Each function takes your inputs, applies some rules, and returns a result.

In the above example, the print() is a function. It accepts letters(strings) and shows it on the screen.

To run your code in VS Code right click on your Program and select Run in Terminal.
Great! Your First Python Program runs.

Summary

- Use the **print()** function to show a message on the screen.
- Use the Python IDLE to type Python code and execute it immediately.

Whitespace and indentation

Python uses whitespace and indentation to construct the code structure.
The following shows a snippet of Python code:

```
# define main function to print out something
def main():
    i = 1
    max = 10
    while (i < max):
        print(i)
        i = i + 1

# call function main
main()
```

The meaning of the code isn’t important to you now. Please pay attention to the code structure instead.

At the end of each line, you don’t see any semicolon to terminate the statement. And the code uses indentation to format the code.

By using indentation and whitespace to organize the code, Python code gains the following advantages:

- First, you’ll never miss the beginning or ending code of a block.
- Second, the coding style is essentially uniform. If you have to maintain another developer’s code, that code looks the same as yours.

•Third, the code is more readable and clear in comparison with other programming languages

Comments

The comments are as important as the code because they describe why a piece of code was written.

When the Python interpreter executes the code, it ignores the comments.

In Python, a single-line comment begins with a hash (#) symbol followed by the comment. For example:

Python inline comments

```
# This is a single line comment in Python
```

Python docstrings

A documentation string is a string literal that you put as the first lines in a code block. Docstrings are commonly used in function, to be covered later.

One-line docstrings

```
""" This an example of one line doc strings """
```

Multi-line docstrings

```
""" This an example of one line doc strings
Another line here
Another Line here """
```

Summary

- Use comments to document your code when necessary.
- A block comment and inline comment starts with a hash sign (#).
- Use docstrings for functions etc.

Python Variables

What is a variable in Python

When you develop a program, you need to manage values, a lot of them. To store values, you use variables.

In Python, a variable is a label that you can assign a value to it. And a variable is always associated with a value. For example:

```
message = 'Hello, World!'
print(message)

message = 'Good Bye!'
print(message)
```

Output:

```
Hello, World!
Good Bye!
```

In this example, message is a variable. It holds the string 'Hello, World!'.
The print() function shows the message Hello, World! to the screen.

The next line assigns the string 'Good Bye!' to the message variable and print its value to the screen.

Creating variables

To define a variable, you use the following syntax:

```
variable_name = value
```

An Example

count = 10
Above variable count holds 10.

Naming variables

When you name a variable, you need to adhere to some rules. If you don’t, you’ll get an error.

The following are the variable rules that you should keep in mind:

- Variable names can contain only letters, numbers, and underscores (_). They can start with a letter or an underscore (_), not with a number.
- Variable names cannot contain spaces. To separate words in variables, you use underscores for example total_marks.
- Variable names cannot be the same as keywords, reserved words, and built-in functions in Python. Refer **Appendix 1**

The following guidelines help you define good variable names:

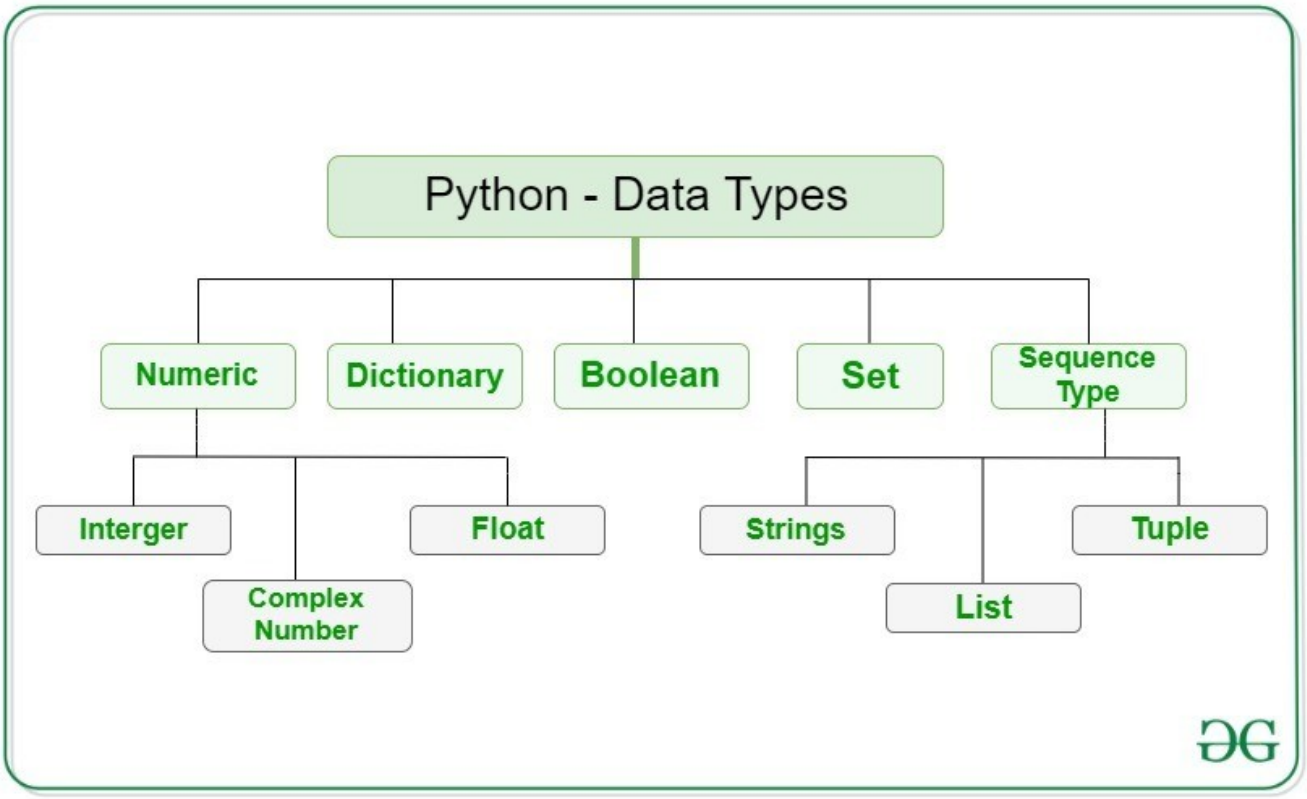
- Variable names should be concise and descriptive. For example, the active_user variable is more descriptive than the au.
- Use underscores (_) to separate multiple words in the variable names.
- Avoid using the letter l and the uppercase letter O because they look like the number 1 and 0.

Summary

- A variable is a label that you can assign a value to it. The value of a variable can change throughout the program.
- Use the variable_name = value to create a variable.
- The variable names should be as concise and descriptive as possible. Also, they should adhere to Python variable naming rules.

Python Data Types

Python Data Types are used to define the type of a variable.



Python String

Introduction to Python string

A string is a series of characters. In Python, anything inside quotes is a string. And you can use either single or double quotes. For example:

```
message = 'This is a string in Python'
message = "This is also a string"
```

If a string contains a single quote, you should place it in double-quotes like this:

```
message = "It's a string"
```

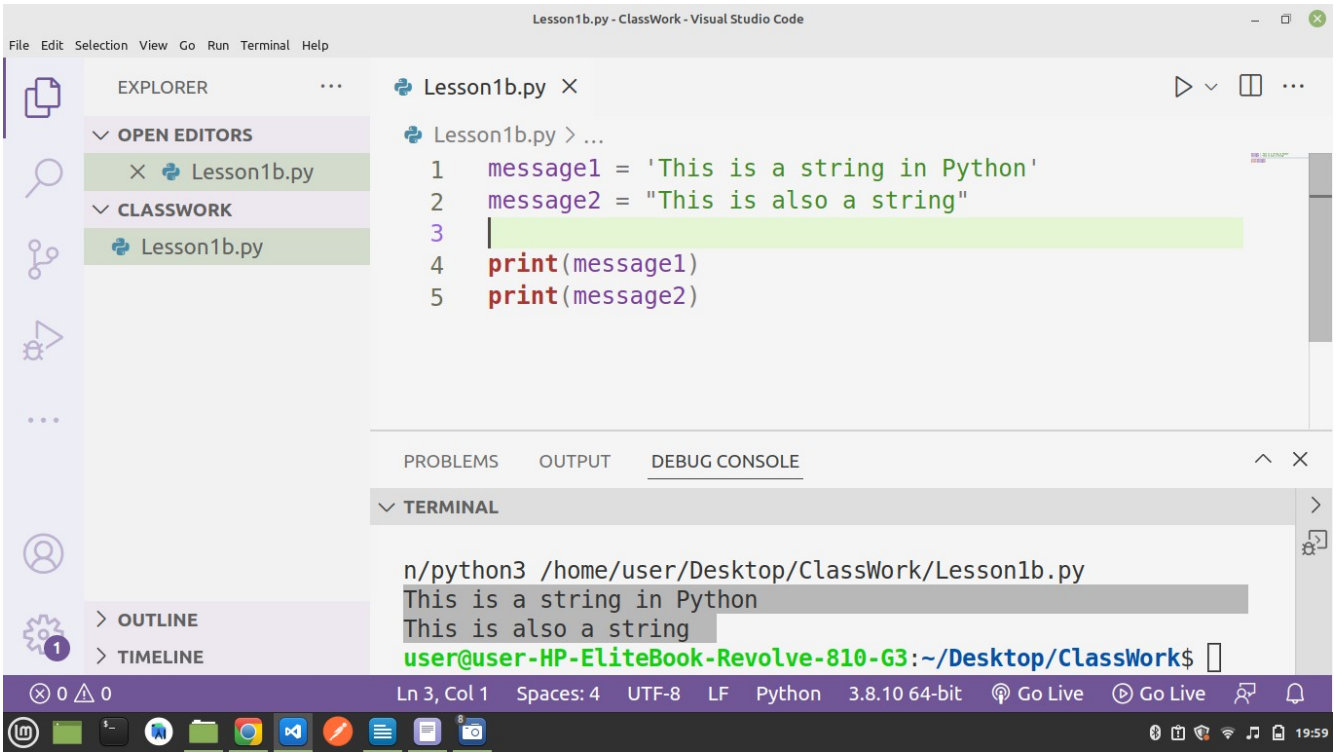
And when a string contains double quotes, you can use the single quotes:

```
message = '"Beautiful is better than ugly.". Said Tim Peters'
```

To escape the quotes, you use the backslash (`\`). For example:

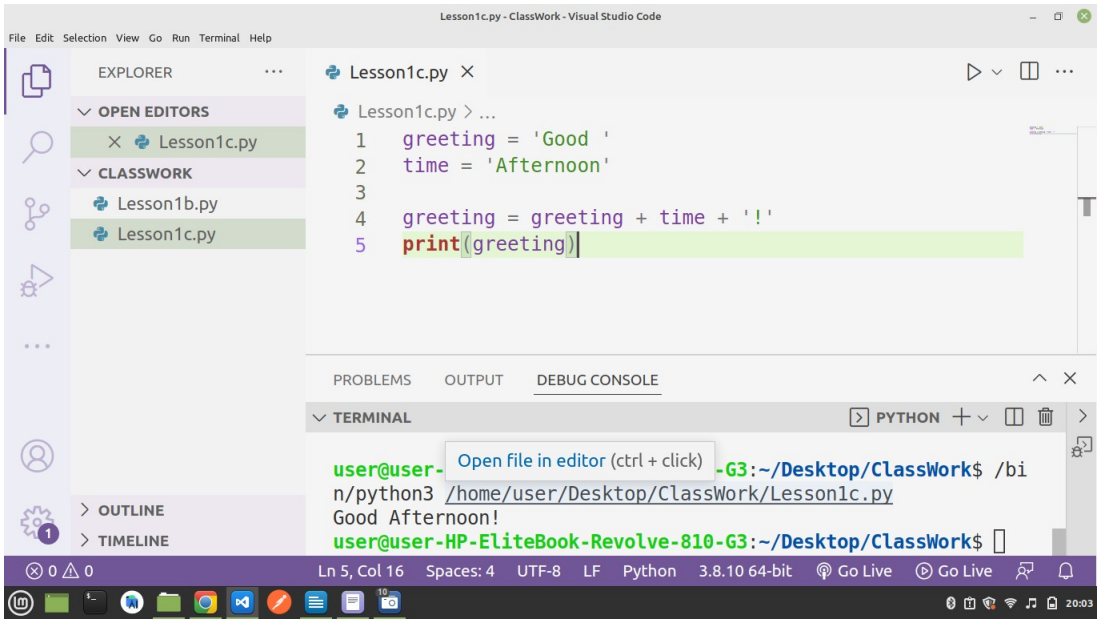
```
message = 'It\'s also a valid string'
```

Create a New File Named Lesson1b.py Put the code below



Concatenating/Combining two strings variable use +.

Create a new Python File Named Lesson1c.py, put code below



Summary

•In Python, a string is a series of characters. Also, Python strings are immutable.

•Use quotes, either single quotes or double quotes to create string literals.

•Use the backslash character \ to escape quotes in strings

•Use print() function to display your strings.

Integers

Python supports integers, floats, and complex numbers. This tutorial discusses only integers and floats.

Integers are whole numbers that include negative numbers, zero, and positive numbers such as -3, -2, -1, 0, 1, 2, 3.

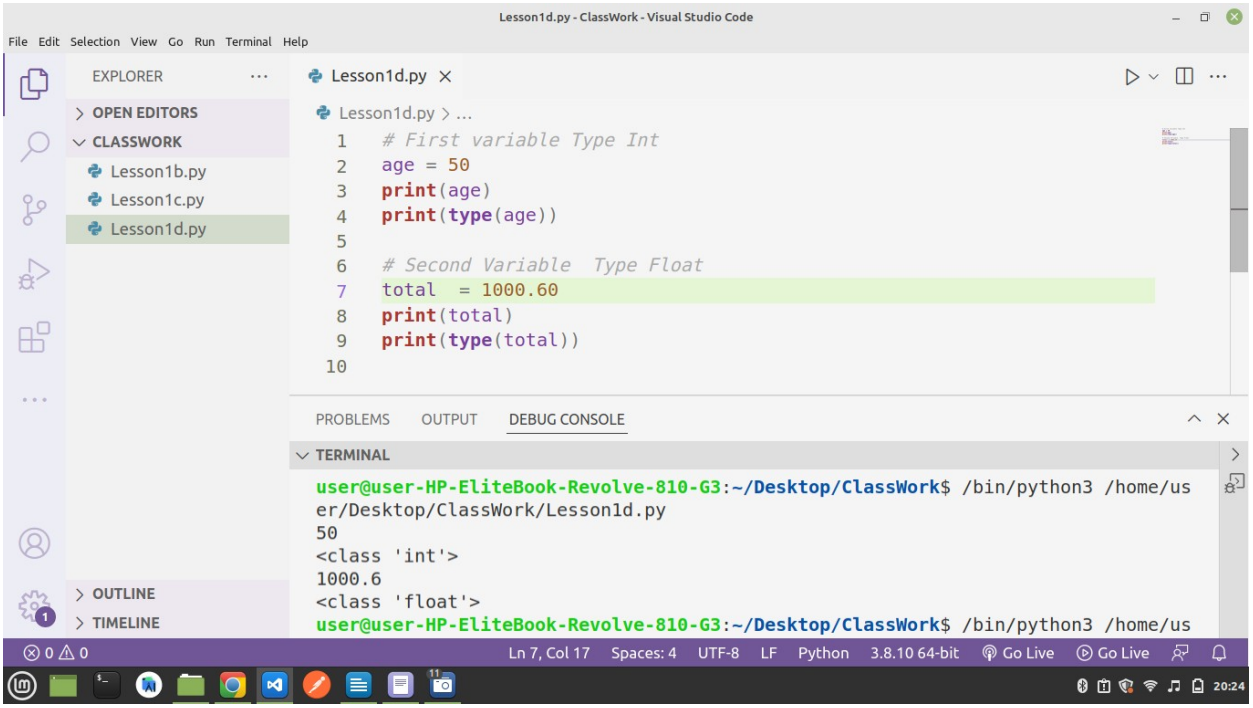
Python int type

The following defines a variable that references an integer and uses the type() function to get the class name of the integer

Floats

Python uses the float class to represent real numbers/Numbers with decimals, From Negative to Positive.

Create a new file named Lesson1d.py, type this code inside

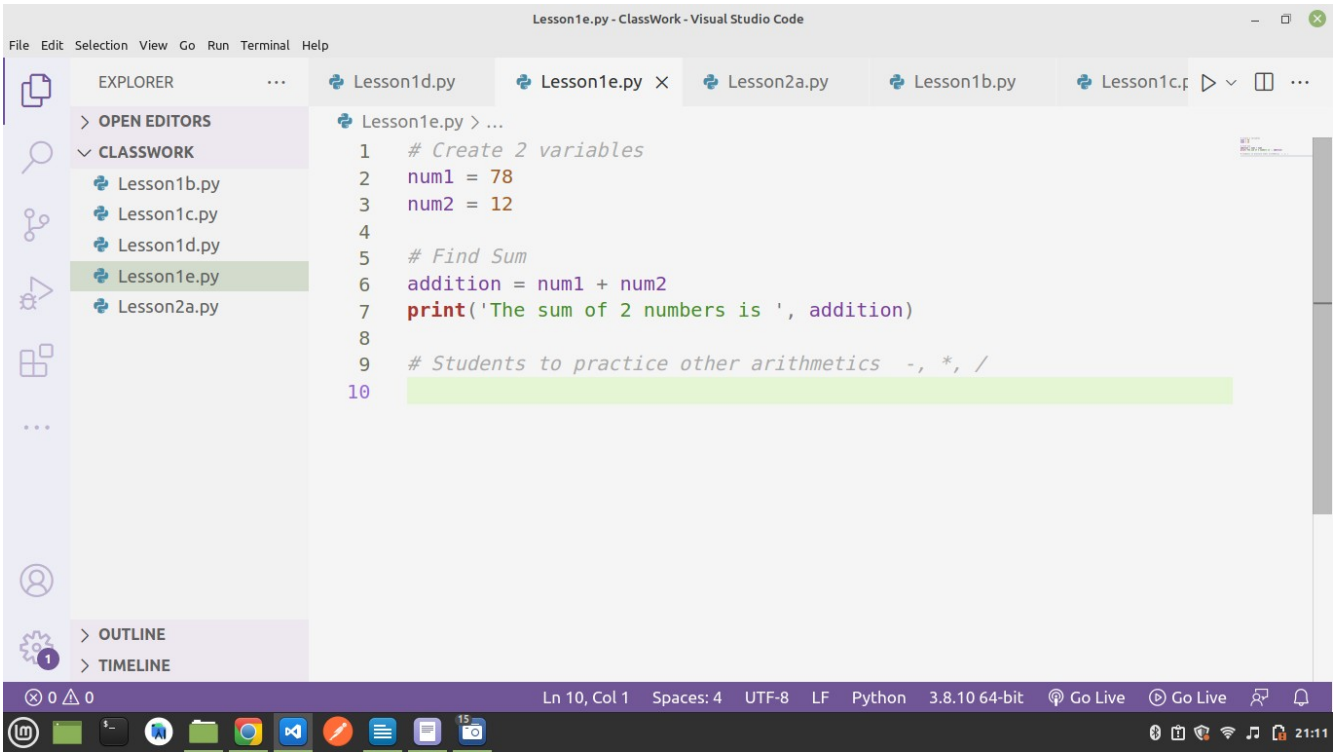


You can do Arithmetic Operators on variables.

Arithmetic Operators

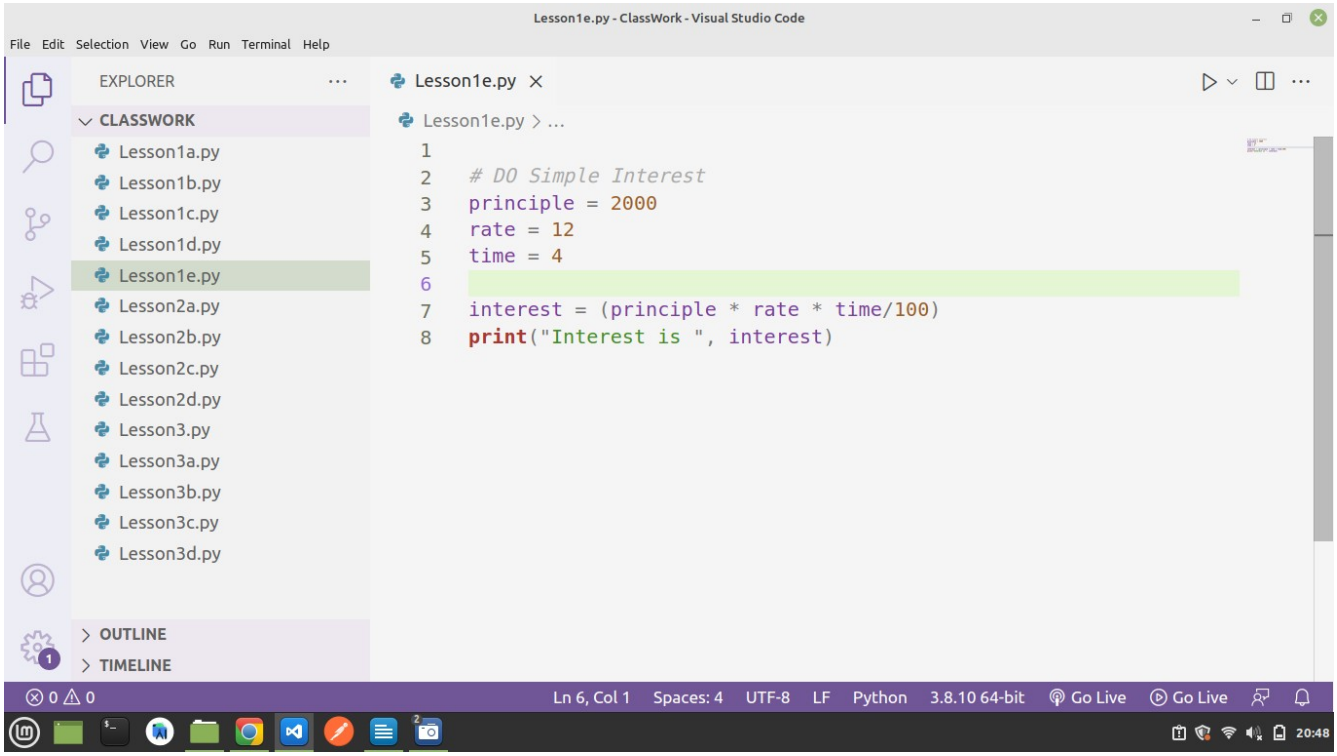
Operators	Meaning	Example	Result
+	Addition	4 + 2	6
–	Subtraction	4 – 2	2
*	Multiplication	4 * 2	8
/	Division	4 / 2	2
%	Modulus operator to get remainder in integer division	5 % 2	1
**	Exponent	5**2 = 5^2	25
//	Integer Division/ Floor Division	5//2 -5//2	2 -3

Create a File named Lesson1e.py, Write this code



Here is a simple example to Find Simple Interest

Create a File named Lesson1e.py , write below code



Assignment

Students to create a Program to find Area of a triangle

Formula: 1/2bh.

Python Boolean

Introduction to Python Boolean data type

In programming, you often want to check if a condition is true or not and perform some actions based on the result.

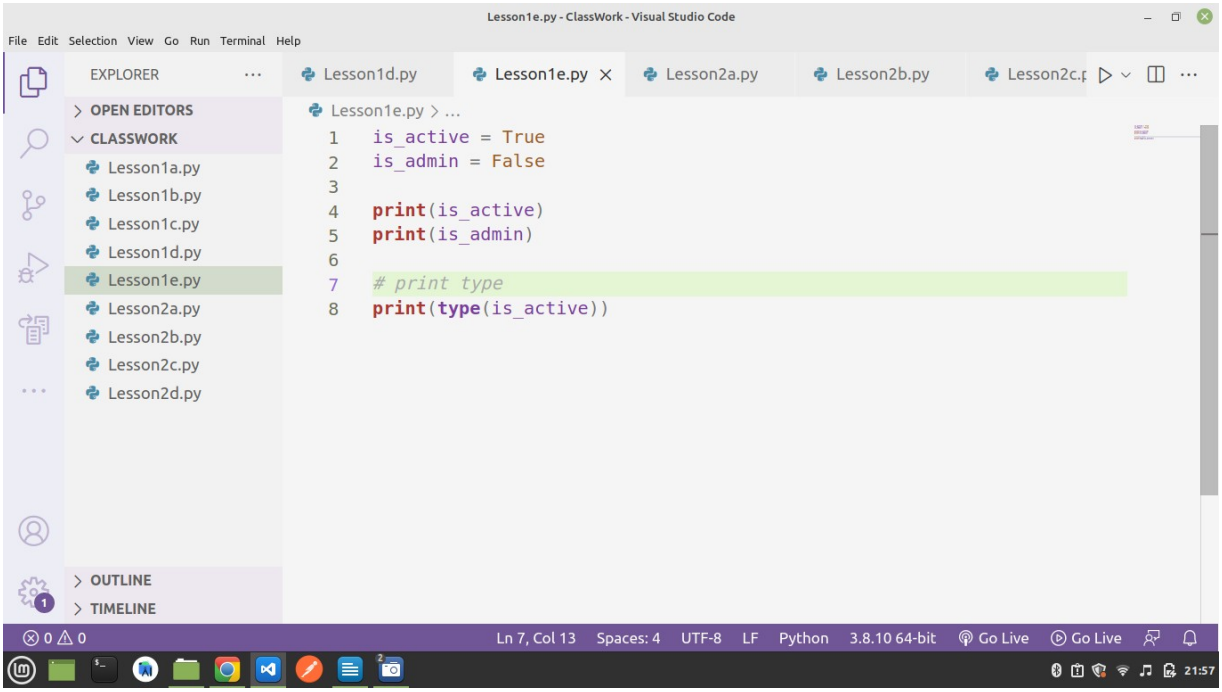
To represent true and false, Python provides you with the boolean data type. The boolean value has a technical name as bool.

The boolean data type has two values: True and False.

Note that the boolean values True and False start with the capital letters (T) and (F).

Example.

Create a File named Lesson1e.py, write below code.



Summary

•Python boolean data type has two values: True and False.

Python Lists

What is a List

A list is an ordered collection of items.

Python uses the square brackets ([]) to indicate a list. The following shows an empty list:

Typically, a list contains one or more items. To separate two items, you use a comma (,). For example:

The following example defines a list of six numbers:

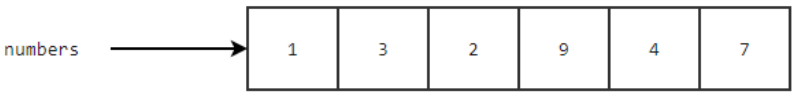
```
numbers = [1, 3, 2, 7, 9, 4]
```

If you print out the list, you'll see its representation including the square brackets. For example:

```
print(numbers)
```

Output:

```
[1, 3, 2, 7, 9, 4]
```



Accessing elements in a list

Since a list is an ordered collection, you can access its element by indexes like this:

list[index]

Example

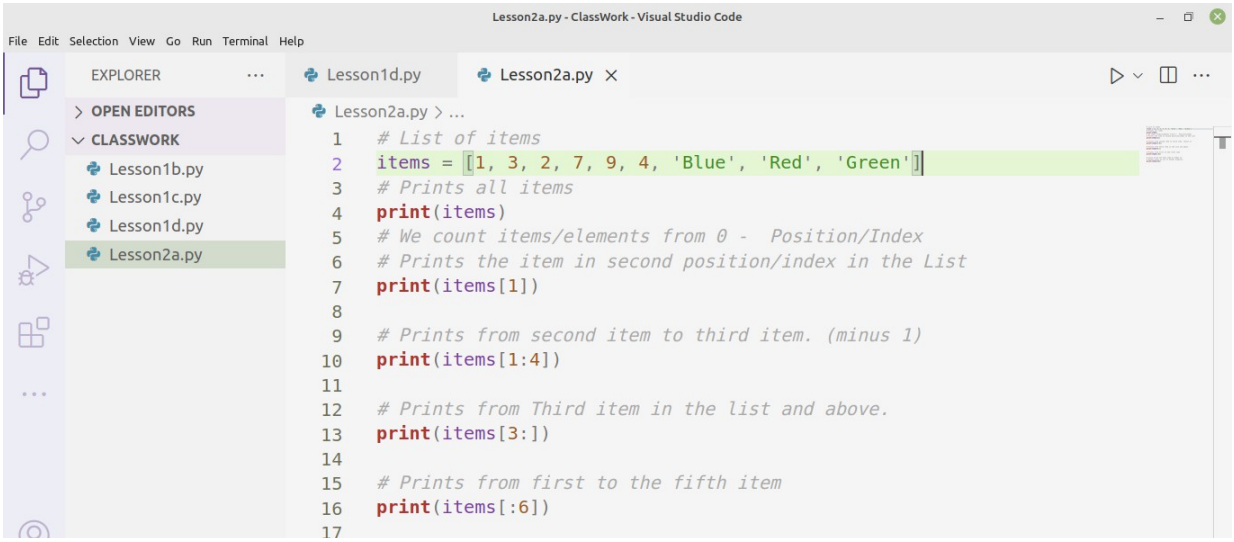
```
numbers = [1, 3, 2, 7, 9, 4]

print(numbers[0])
```

Output:

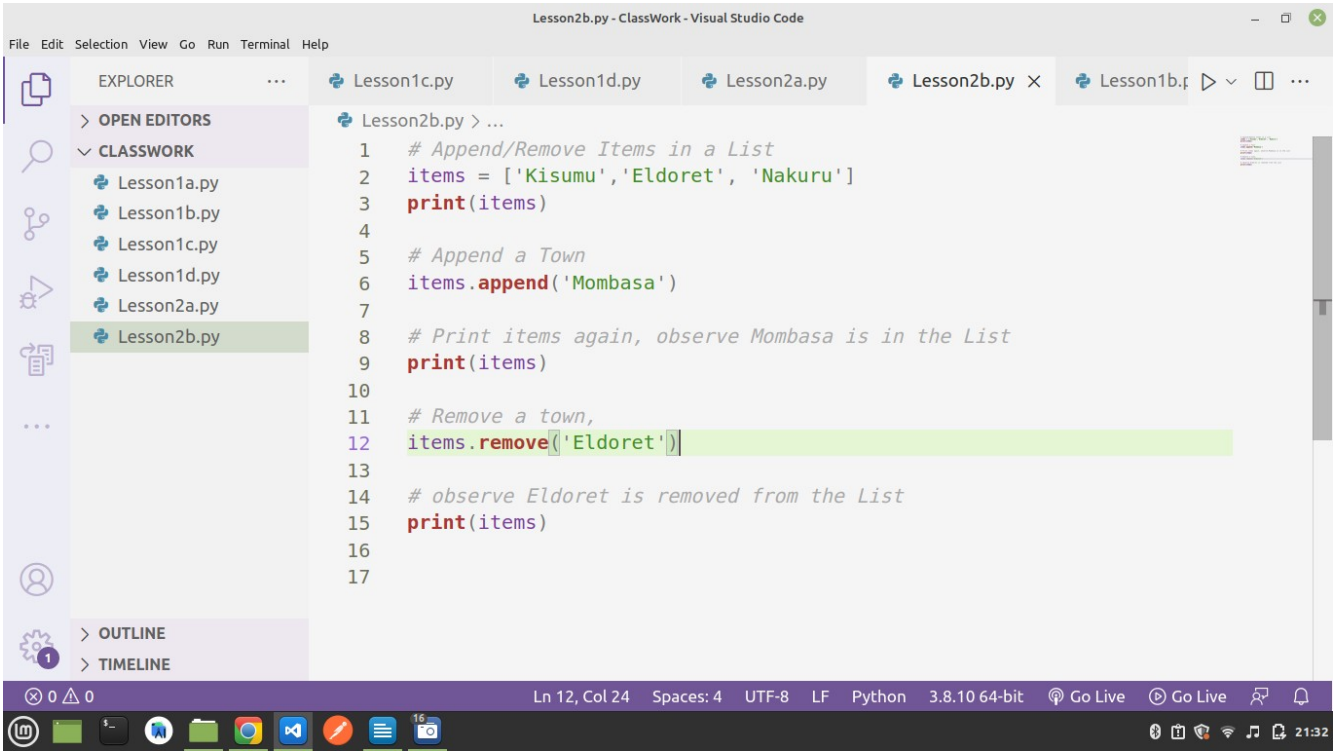
```
1
```

Create a File named Lesson2a.py, Write below code.



You can append/ remove items in a list, Lets do it practically

Create another file named Lesson2a.py, write below code.



Output

['Kisumu', 'Eldoret', 'Nakuru']

['Kisumu', 'Eldoret', 'Nakuru', 'Mombasa']

['Kisumu', 'Nakuru', 'Mombasa']

Summary

- Lists are mutable lists.
- Use lists when you want to define a list that can change.
- Use [] for defining Lists

Python Tuples

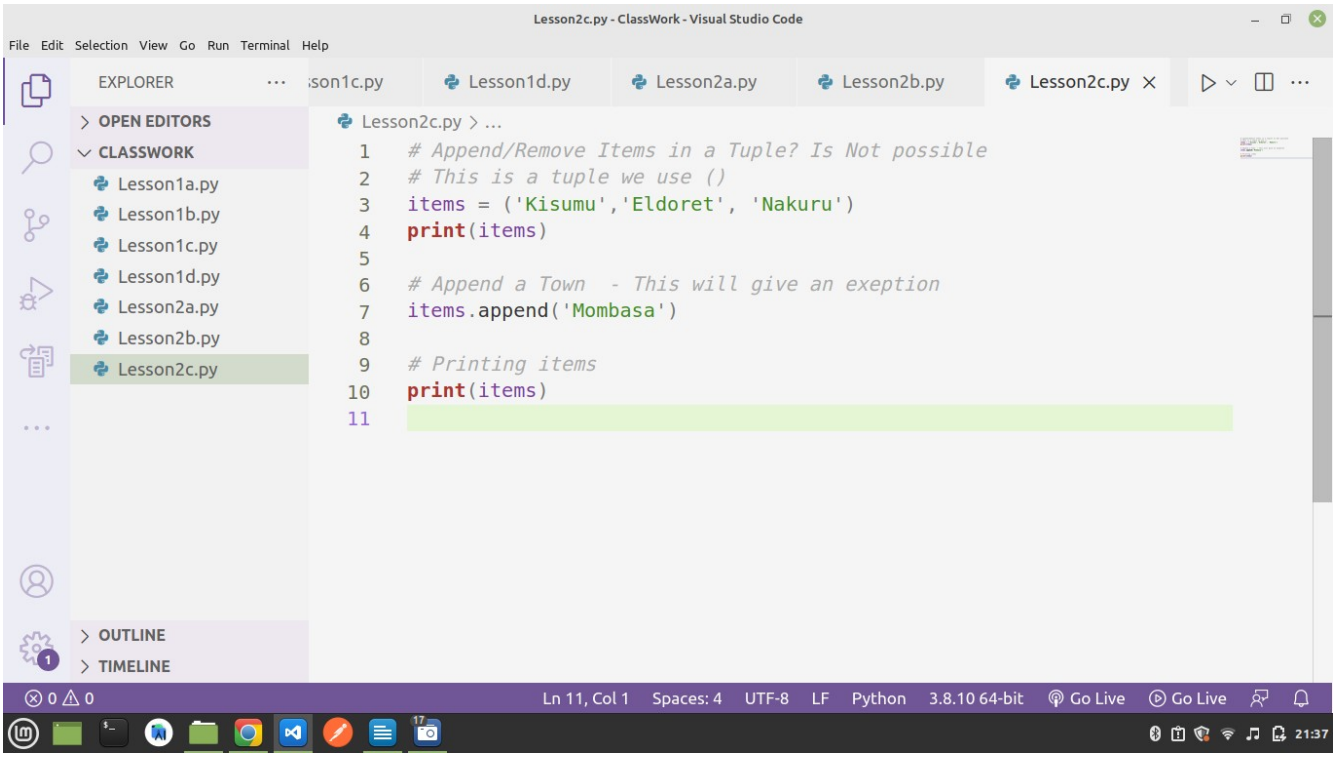
Introduction to Python tuples

Sometimes, you want to create a list of items that cannot be changed throughout the program. Tuples allow you to do that.

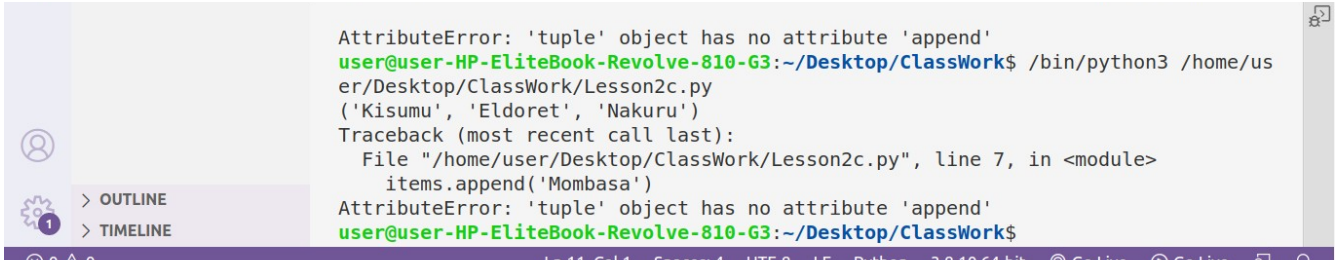
A tuple is a list that cannot change. Python refers to a value that cannot change as immutable. So by definition, a tuple is an immutable list

A tuple is like a list except that it uses parentheses () instead of square brackets [].

Create a new File named Lesson2c.py, write the following code.



Output



Above output shows that we cannot append a Town Mombasa in a Tuple.

Summary

- Tuples are immutable lists.
- Use tuples when you want to define a list that cannot change.
- Use () for defining tuples

Python Dictionary

Introduction to the Python Dictionary type

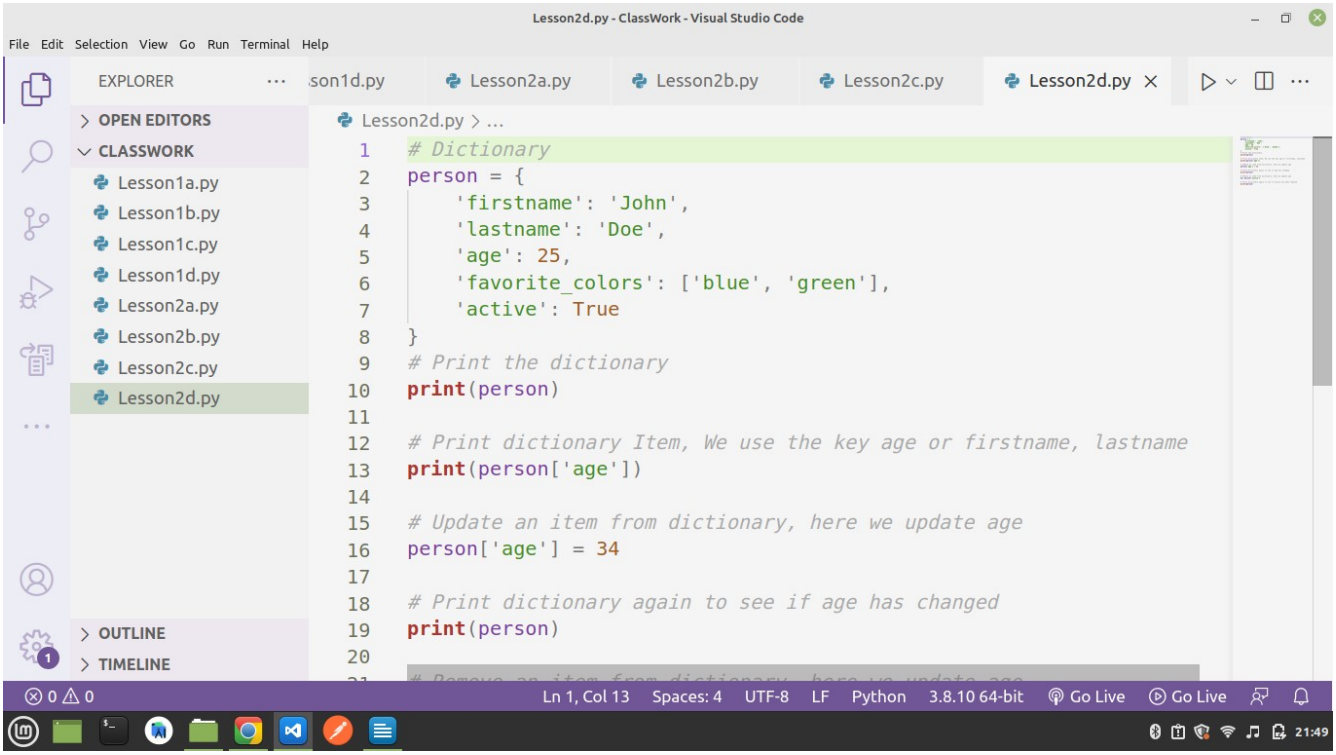
A Python dictionary is a collection of key-value pairs where each key is associated with a value.

A value in the key-value pair can be a number, a string, a list, a tuple, or even another dictionary. In fact, you can use a value of any valid type in Python as the value in the key-value pair.

A key in the key-value pair must be immutable. In other words, the key cannot be changed, for example, a number, a string, a tuple, etc.

Python uses curly braces {} to define a dictionary. Inside the curly braces, you can place zero, one, or many key-value pairs.

Create a file named Lesson2d.py, write below code



Summary

- A Python dictionary is a collection of key-value pairs, where each key has an associated value.
- Use square brackets or get() method to access a value by its key.
- Use the del statement to remove a key-value pair by the key from the dictionary.

Control flow

Python has two types of control statements. **Conditional** and **Iterative** (Looping) with their particular uses. We will learn conditional statements like IF, IF-ELSE, ELIF, along with their respective syntaxes. And for Iterative Statements, we will learn about WHILE and FOR along with their syntaxes.

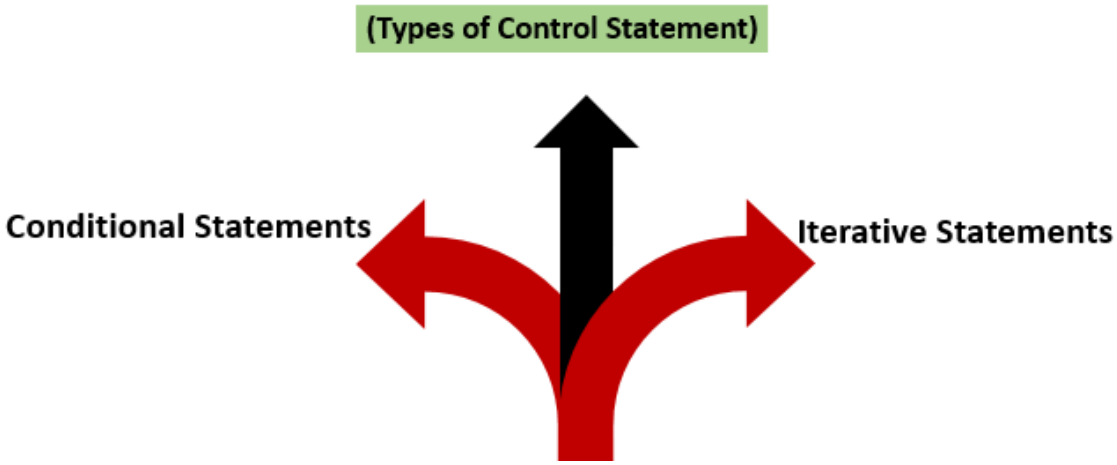
Before we get into Control Flow lets look at Comparison and Logical Operators.

Relational/Comparison Operators

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

Logical Operators

Operator	Meaning	Example	Result
and	Logical and	(5<2) and (5>3)	False
or	Logical or	(5<2) or (5>3)	True
not	Logical not	not (5<2)	True



We start with Conditional Statements

In computer programming, we use the `if` statement to run a block code only when a certain condition is met.

For example, assigning grades **(A, B, C)** based on marks obtained by a student.

1. if the percentage is above **90**, assign grade **A**
2. if the percentage is above **75**, assign grade **B**
3. if the percentage is above **65**, assign grade **C**

In Python, there are three forms of the `if...else` statement.

1. `if` statement
2. `if...else` statement
3. `if...elif...else` statement

1. Python if statement

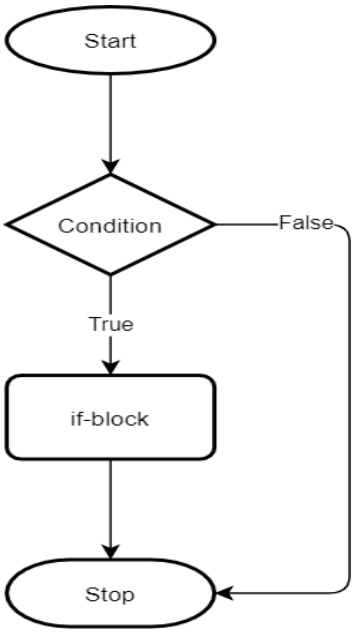
The syntax of `if` statement in Python is:

```
if condition:
    # body of if statement
```

The `if` statement evaluates `condition`.

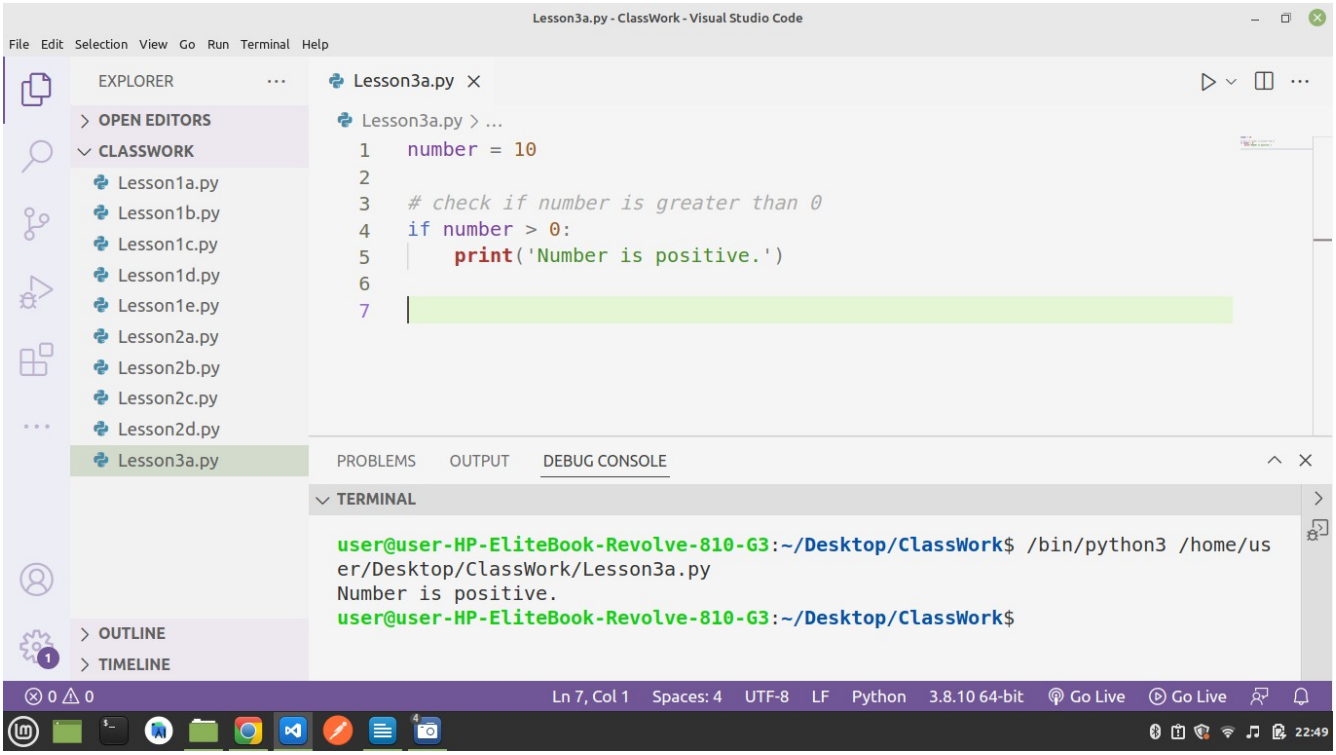
1. If `condition` is evaluated to `True`, the code inside the body of `if` is executed.
2. If `condition` is evaluated to `False`, the code inside the body of `if` is skipped.

How it works.



Example 1: Python if Statement

Create a File Named Lesson3a.py, write below code.



2. Python if...else Statement

An `if` statement can have an optional `else` clause.

The syntax of `if...else` statement is:

```
if condition:
    # block of code if condition is True

else:
    # block of code if condition is False
```

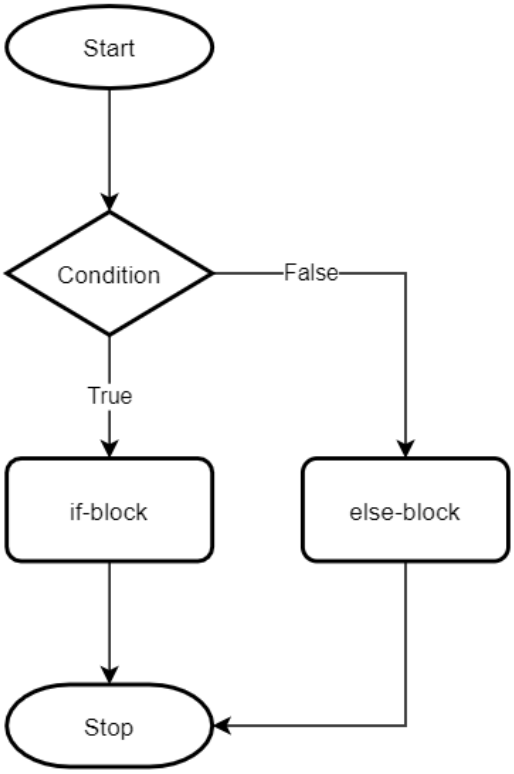
If the condition evaluates to True,

- the code inside `if` is executed
- the code inside `else` is skipped

If the condition evaluates to False,

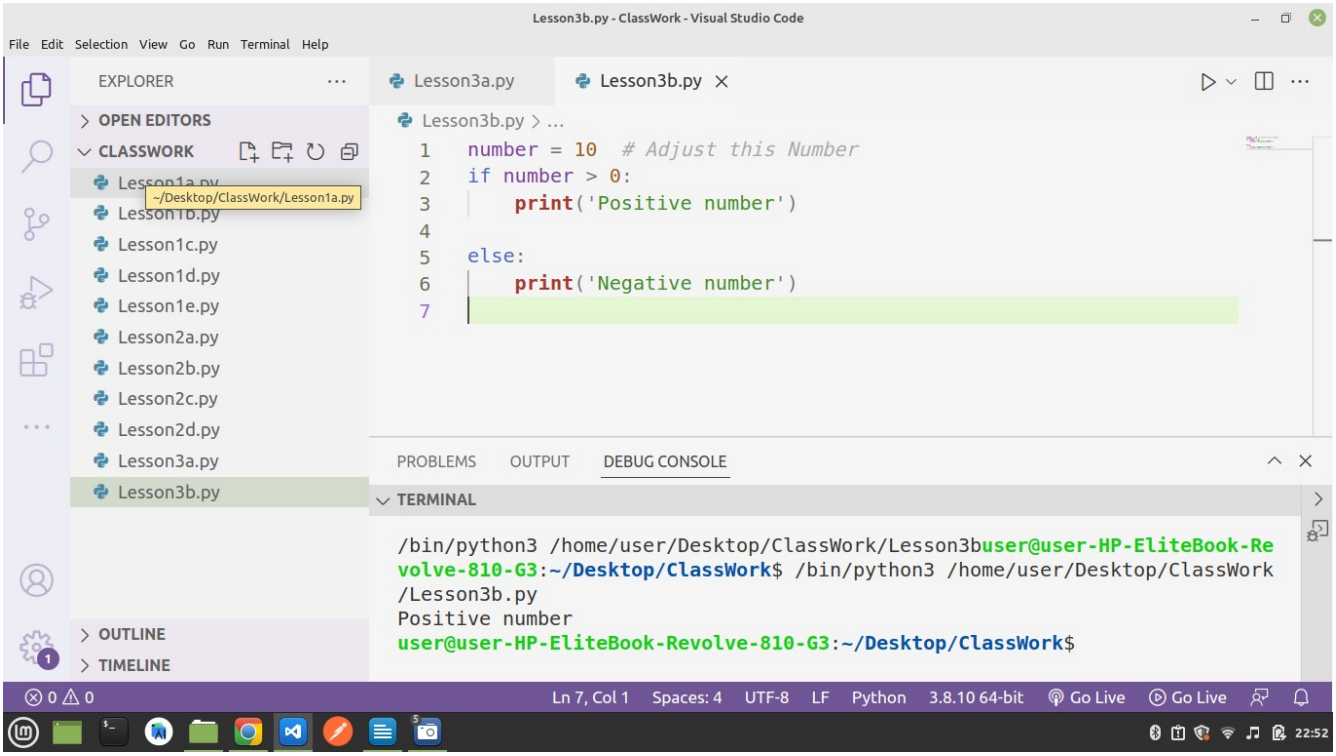
- the code inside `else` is executed
- the code inside `if` is skipped

How its Works



Example 2. Python if...else Statement

Create a File named Lesson3b.py, write below code.



3. Python if...elif...else Statement

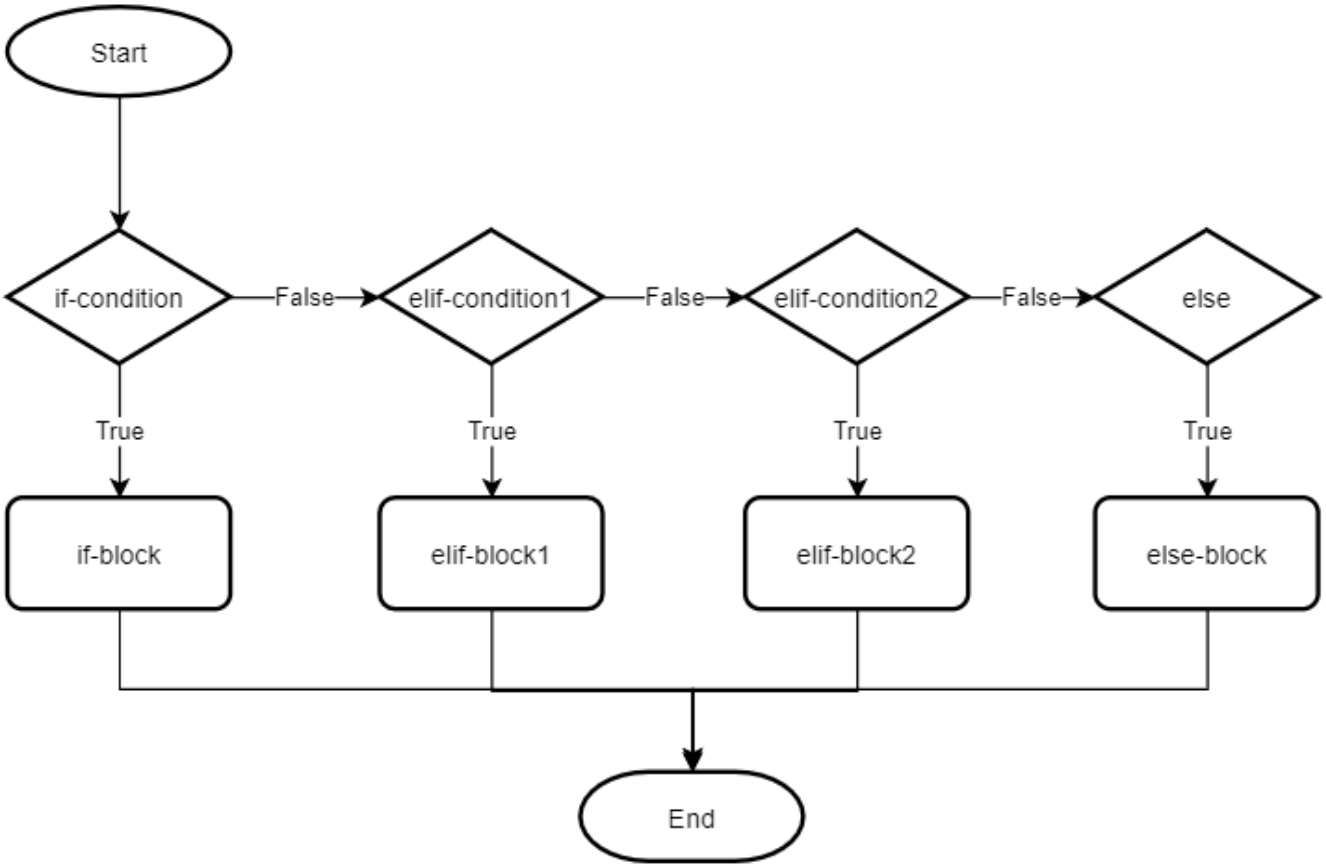
The `if...else` statement is used to execute a block of code among two alternatives. However, if we need to make a choice between more than two alternatives, then we use the `if...elif...else` statement.

The syntax of the `if...elif...else` statement is:

```
if condition1:
    # code block 1

elif condition2:
    # code block 2

else:
    # code block 3
```

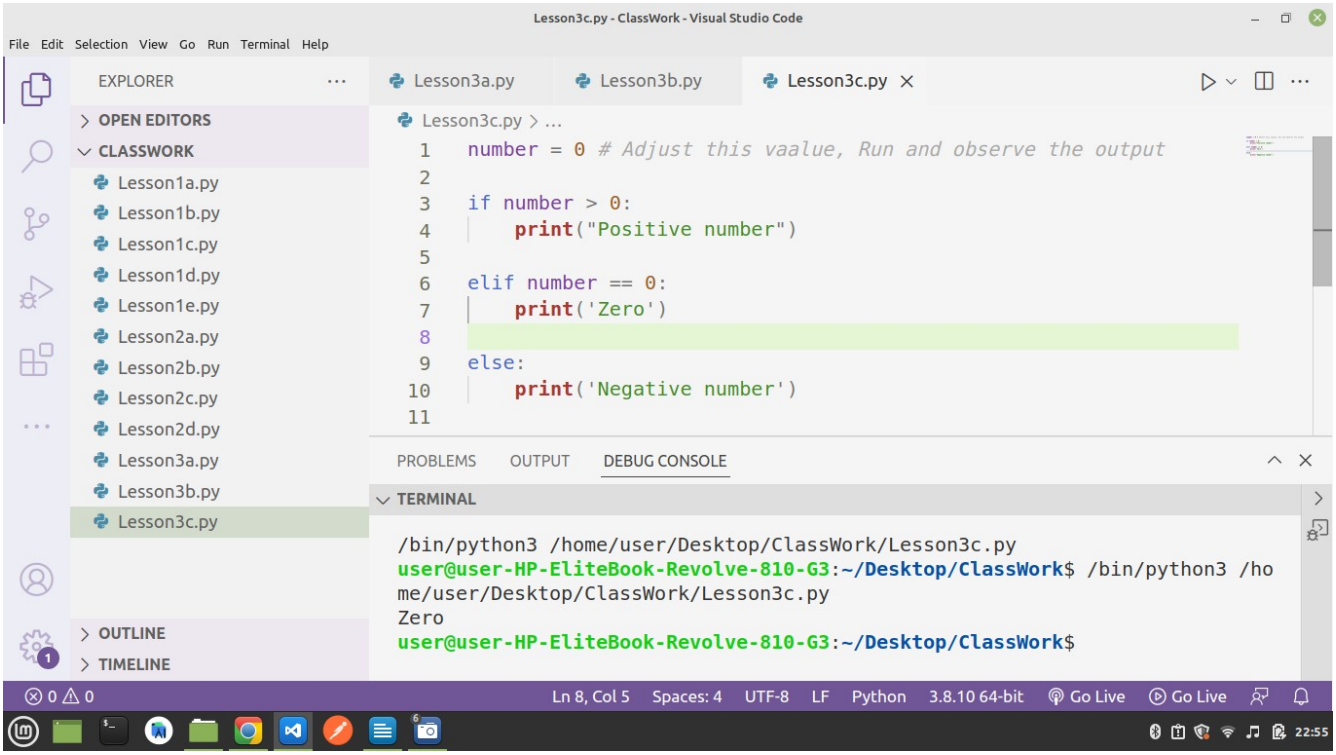


The if...elif...else statement checks each condition (if-condition, elif-condition1, elif-condition2, ...) in the order that they appear in the statement until it finds the one that evaluates to True.

When the if...elif...else statement finds one, it executes the statement that follows the condition and skips testing the remaining conditions.

If no condition evaluates to True, the if...elif...else statement executes the statement in the else branch.

Example 3: Python if...elif...else Statement



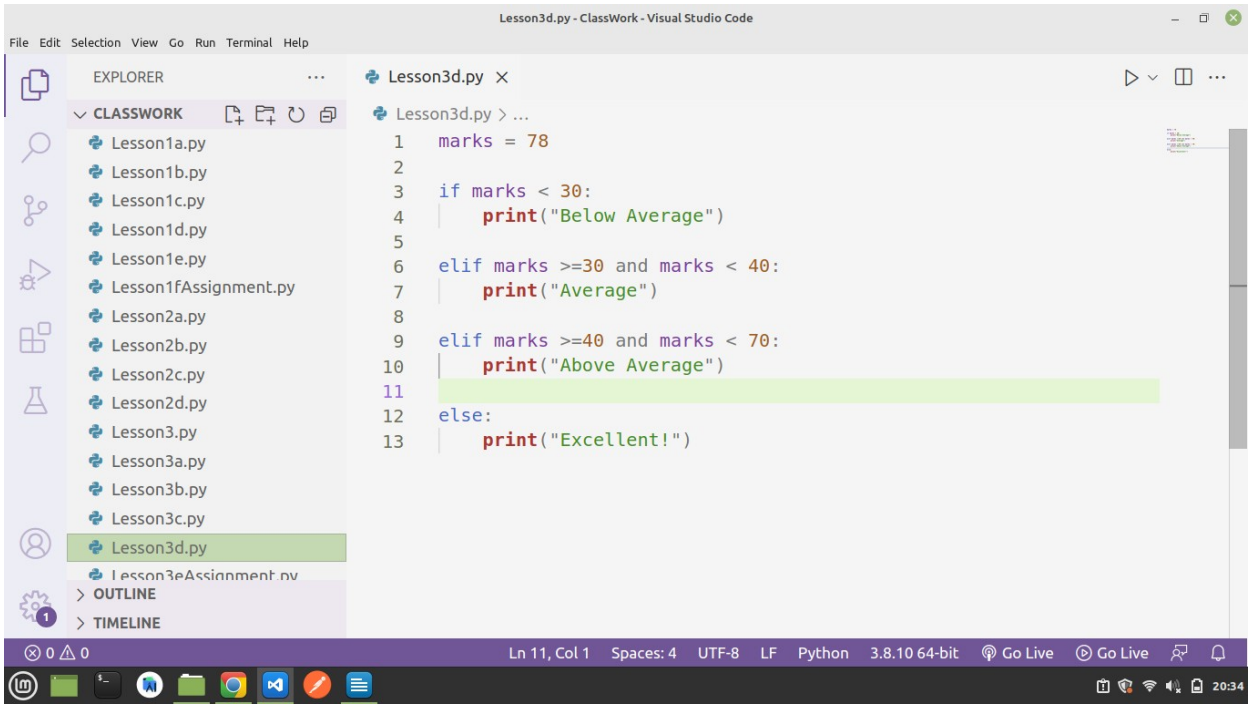
Students marks Example

In this example we have a variable marks

We check the conditions using conditional operators based on marks variable and give respective messages.

See below code

Create a File named Lesson3d.py , write below code



Assignment

Given a a variable age = 20.

Create a Python program to check;

when age is less than 10 the program should print “You are in Primary Classes”

when age more 12 and less than 15 the program should print “You are in Junior Secondary”

when age > 15 and less than 19 the program should print “You are in Senior Secondary”

when age > 19 the program should print “You are in College”

Python Nested if statements

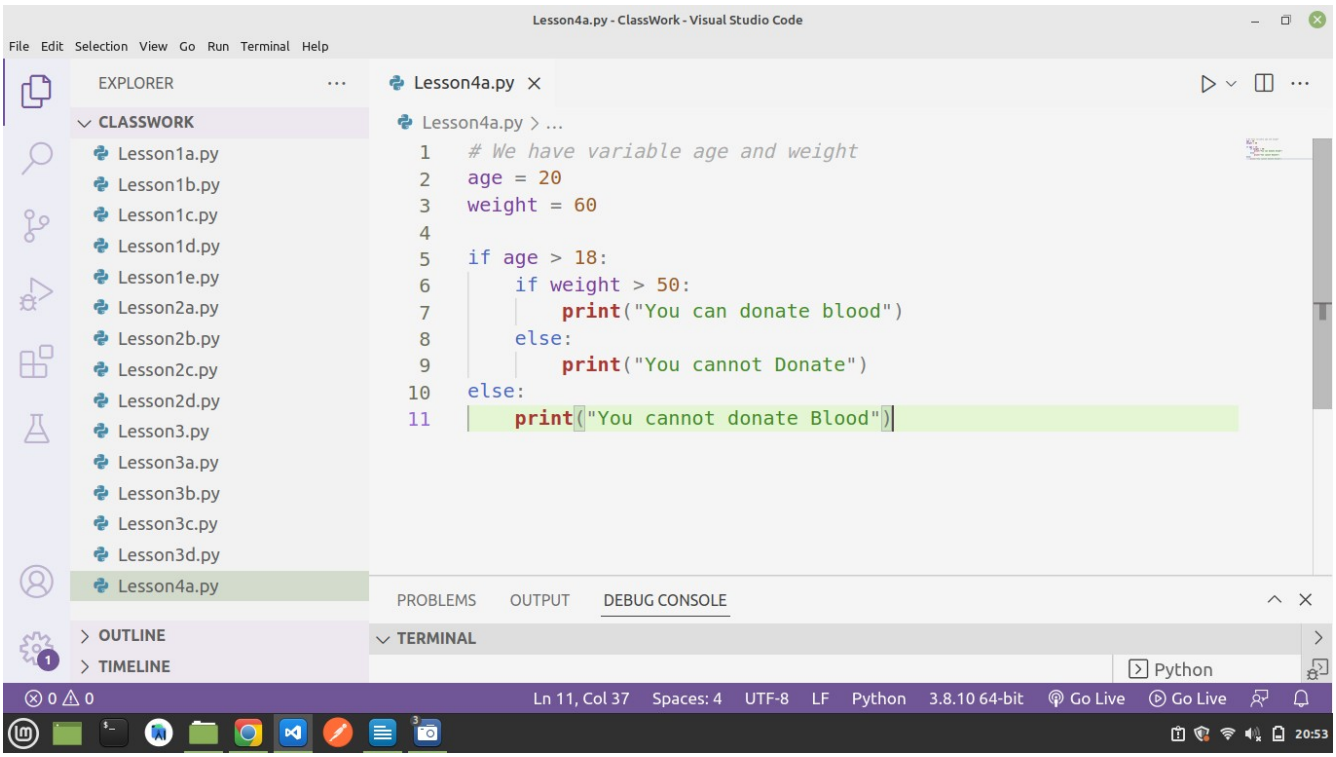
We can also use an `if` statement inside of an `if` statement. This is known as a **nested if** statement.

```
# outer if statement
if condition1:
    # statement(s)

    # inner if statement
    if condition2:
        # statement(s)
```

Nested if Statements

Example Create a File named Lesson4a.py, write below program



Code Explanation

On Line 5, we check if age > 18, If its true, on Line6 we check if weight > 50, if True we print You can donate blood.

On Line 8, This will execute if weight is less than 50.
On Line 10, This will execute if age is less than 18.

Python Loop

In computer programming, loops are used to repeat a block of code.

For example, if we want to show a message **100** times, then we can use a loop. It's just a simple example; you can achieve much more with loops.
There are 2 types of loops in Python:

1. **for loop**
2. **while loop**

Python for Loop

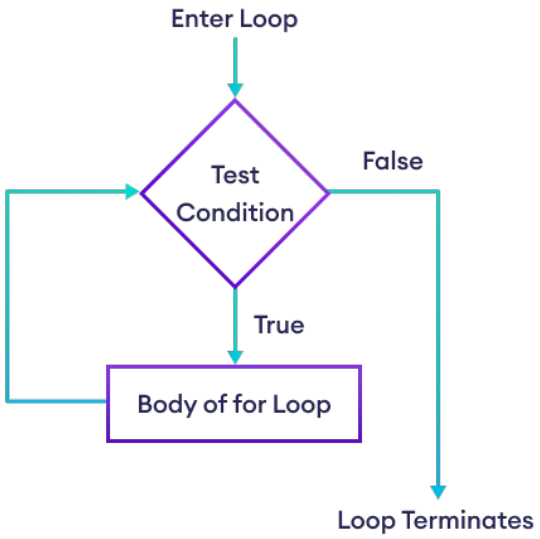
In Python, the `for` loop is used to run a block of code for a certain number of times. It is used to iterate over any sequences such as list, tuple, string, etc.
The syntax of the `for` loop is:

```
for val in range(n):
    # statement(s)
```

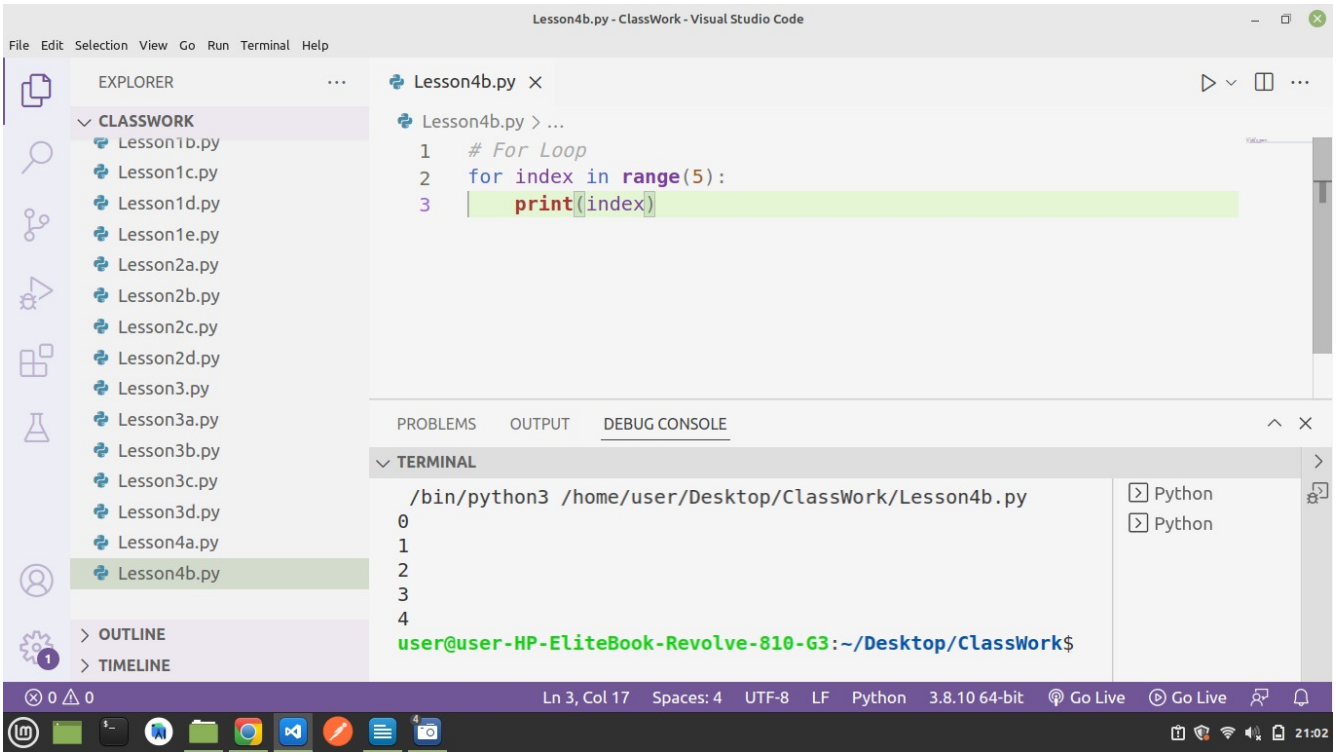
Or

```
for val in sequence:
    # statement(s)
```

Flowchart of Python for Loop



Example using range(n).



Code Explanation

In this syntax, the index is called a **loop counter**. And n is the number of times that the loop will execute the statement.

The name of the loop counter doesn't have to be index, you can use whatever name you want.

The range() is a built-in function in Python. It's like the print() function in the sense that it's always available in the program.

The range(n) generates a sequence of n integers starting at zero. It increases the value by one until it reaches n.

So the range(n) generates a sequence of numbers: 0,1, 2, ...n-1. Note that it's always short of the final number (n).

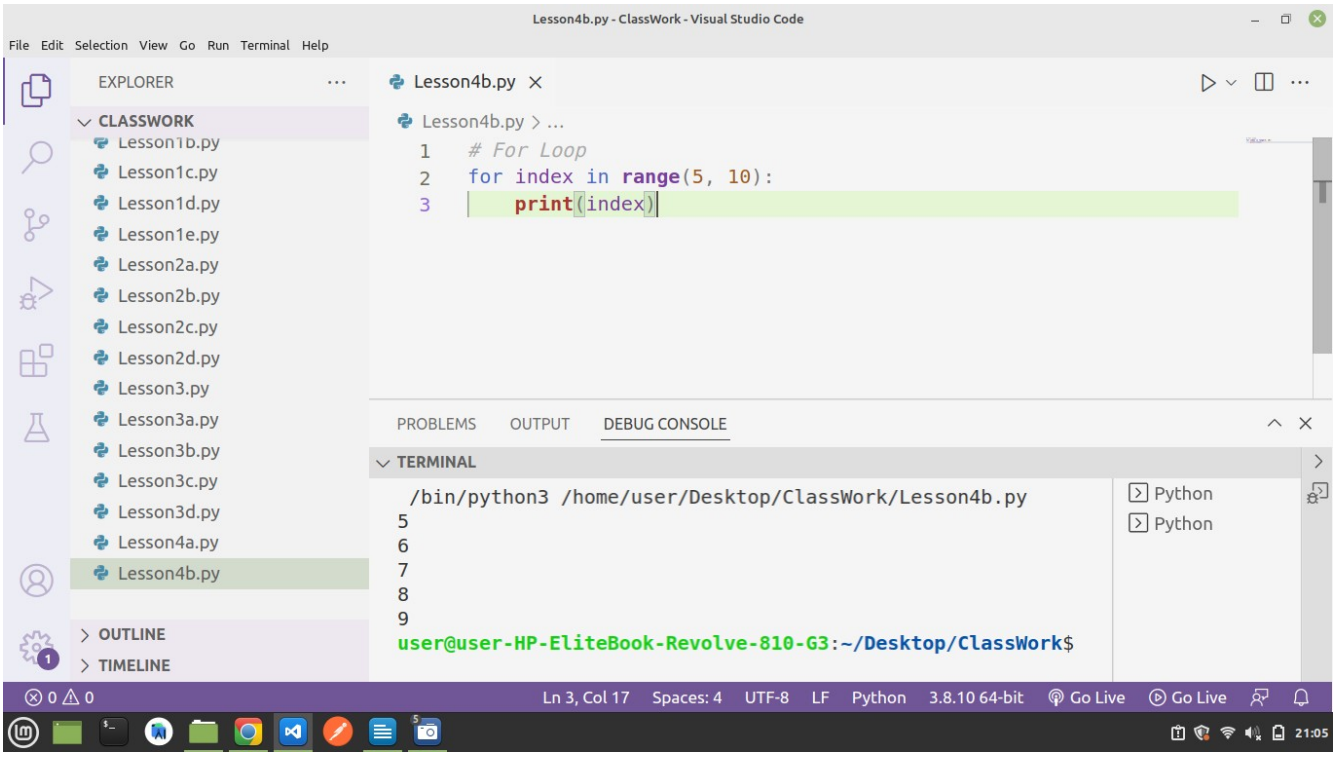
Specifying the starting value for the sequence

By default, the range() function uses zero as the starting number for the sequence.

In addition, the range() function allows you to specify the starting number like this:

range(start, stop)

In this syntax, the range() function increases the start value by one until it reaches the stop value.



Code Explanation

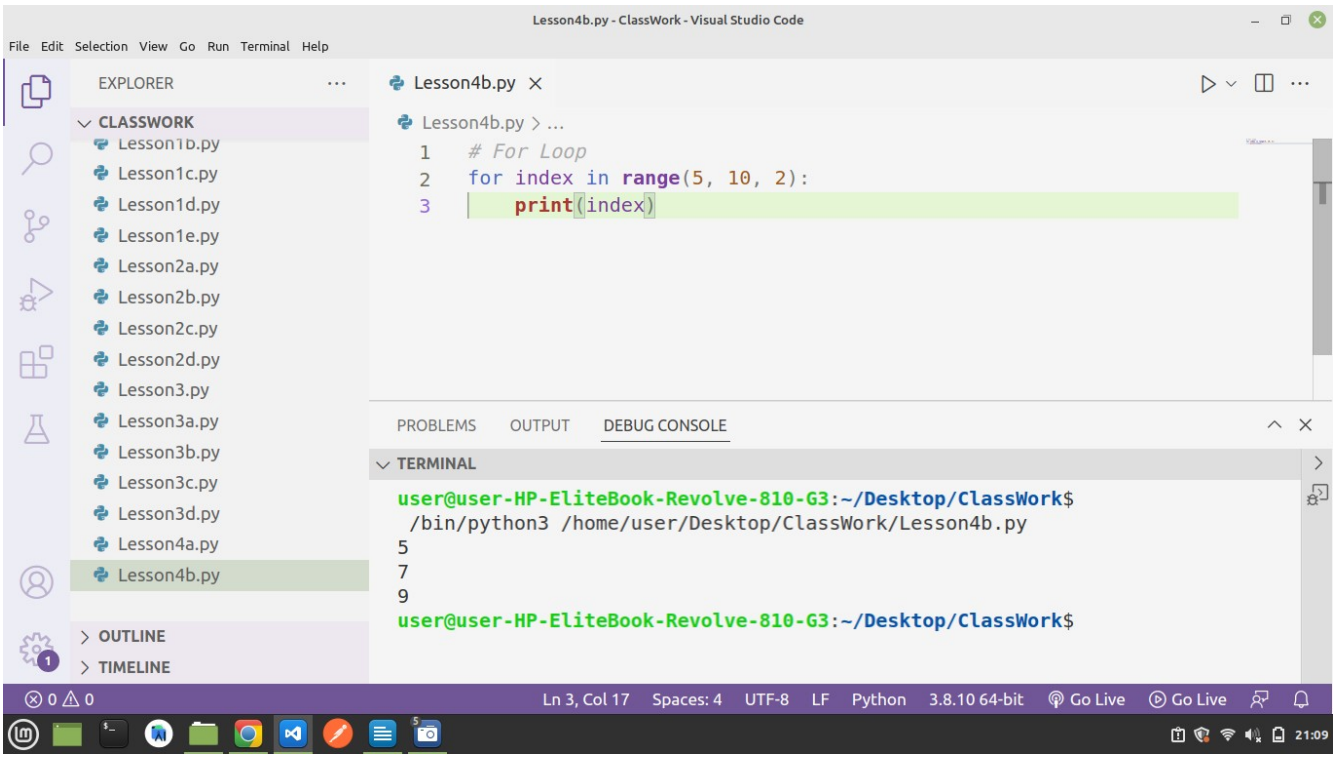
Above code Prints from 5 to 9.

Specifying the increment for the sequence

By default, the range(start, stop) increases the start value by one in each loop iteration.

To increase the start value by a different number, you use the following form of the range() function, using the step.

range(start, stop, step)

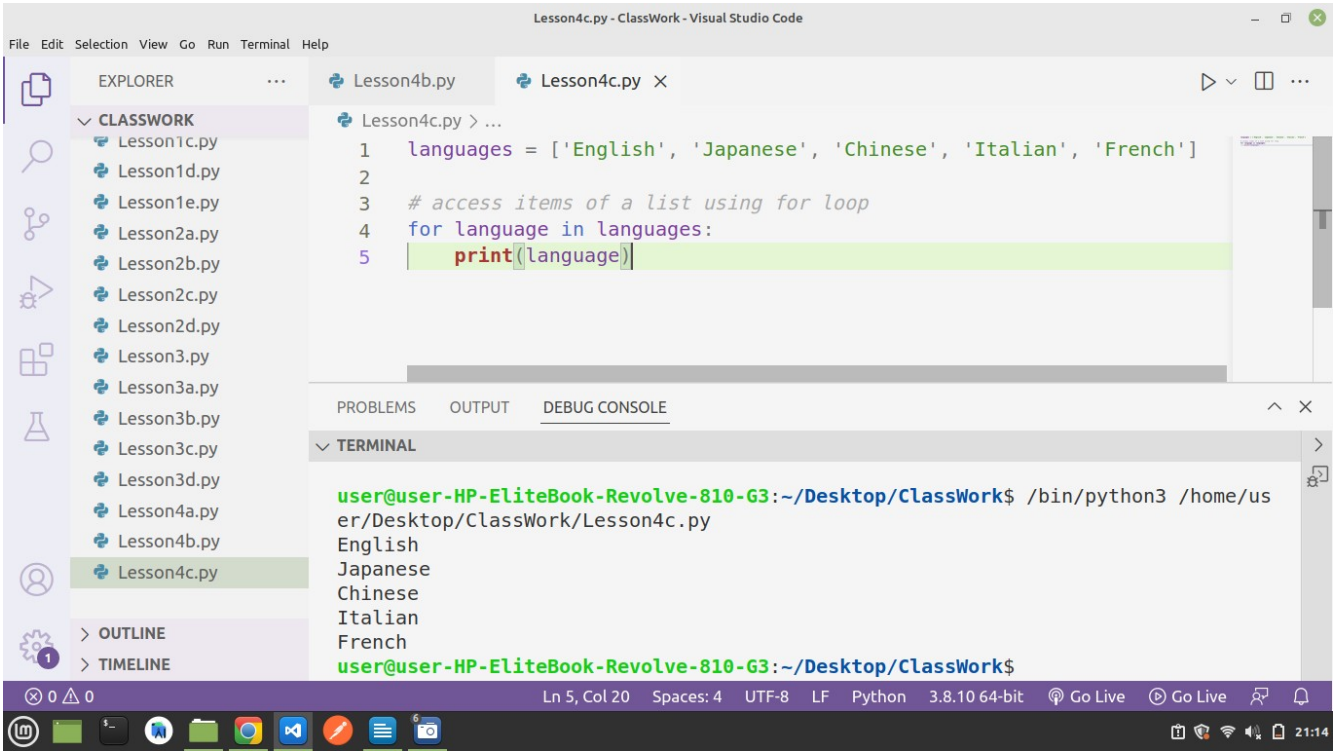


Code Explanation

Above code prints from 5 to 10 but with an increment of 2.

So the answer is 5, 7, 9.

Example: Loop Over Python List



Code Explanation

On Line 1, we create a List of languages.

On Line 4, we loop each language in the languages list collection.

This prints all languages one by one.

Summary

- Use the for loop statement to run a code block a fixed number of times.
- Use the range(start, stop, step) to customize the loop
- Use for value in collection when looping a List.

Python while Loop

In programming, loops are used to repeat a block of code. For example, if we want to show a message **100** times, then we can use a loop. It's just a simple example, we can achieve much more with loops.

In the previous tutorial, we learned about Python for loop. Now we will learn about the `while` loop.

Python while Loop

Python `while` loop is used to run a block code until a certain condition is met.

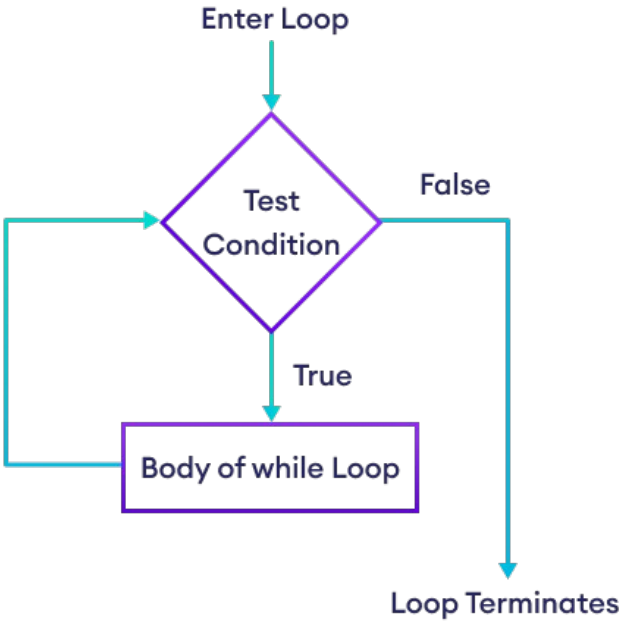
The syntax of `while` loop is:

```
while condition:
    # body of while loop
```

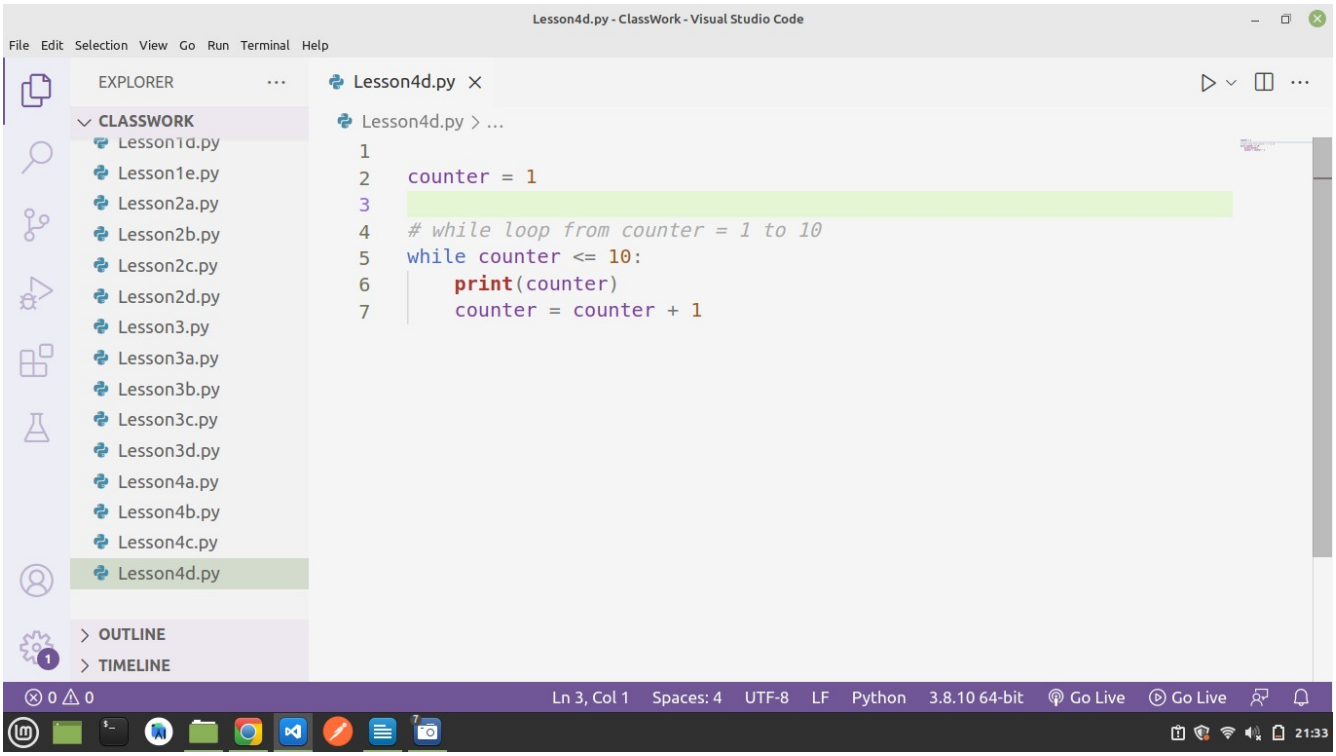
Here,

1. A `while` loop evaluates the `condition`
2. If the `condition` evaluates to `True`, the code inside the `while` loop is executed.
3. `condition` is evaluated again.
4. This process continues until the condition is `False`.
5. When `condition` evaluates to `False`, the loop stops.

Flowchart of Python while Loop



Example: Python while Loop



Code Explanation

On Line 2, We define a variable name counter = 1, this defines the start of the loop.

On Line 5, we check if counter(counter=1)is less than or equal to 10, if its true, On Line 6 we print the counter.

On Line 7, we add 1 to the counter which makes counter to 2.

We repeat On Line 5, we check if counter(counter=2)is less than or equal to 10, if its true, On Line 6 we print the counter.

On Line 7, we add 1 to the counter which makes counter to 3

We repeat On Line 5, we check if counter(counter=3)is less than or equal to 10, if its true, On Line 6 we print the counter.

This is repeated until counter increments to 11 which will make the condition False and the Loop stops

Python Functions

What is a function

A function is a named code block that performs a job or returns a value.

Why do you need functions in Python

Sometimes, you need to perform a task multiple times in a program. And you don’t want to copy the code for that same task all over places.

To do so, you wrap the code in a function and use this function to perform the task whenever you need it.

For example, whenever you want to display a value on the screen, you need to call the print() function. Behind the scene, Python runs the code inside the print() function to display a value on the screen.

In practice, you use functions to divide a large program into smaller and more manageable parts. The functions will make your program easier to develop, read, test, and maintain.

The print() function is one of many built-in functions in Python. It means that these functions are available everywhere in the program.

Defining a Python function

Here’s a simple function that shows a greeting:

```
def greet():
    """ Display a greeting to users """
    print('Hi')
```

This example shows the simplest structure of a function. A function has two main parts: a function definition and body.

1) Function definition

A function definition starts with the def keyword and the name of the function (greet).

If the function needs some information to do its job, you need to specify it inside the parentheses (). The greet function in this example doesn’t need any information, so its parentheses are empty.

The function definition always ends in a colon (:).

2) Function body

All the indented lines that follow the function definition make up the function’s body.

The text string surrounded by triple quotes is called a docstring. It describes what the function does. Python uses the docstring to generate documentation for the function automatically.

The line print('Hi') is the only line of actual code in the function body.
The greet() function does one task: print('Hi').

3) Calling a function

When you want to use a function, you need to call it. A function call instructs Python to execute the code inside the function.

To call a function, you write the function's name, followed by the information that the function needs in parentheses.

The following example calls the greet() function. Since the greet() function doesn't need any information, you need to specify empty parentheses like this:

greet()

If you run the program, it'll show a greeting on the screen:

Hi

Passing information to Python functions

Suppose that you want to greet users by their names. To do it, you need to specify a name in parentheses of the function definition as follows:

```
def greet(name):
```

The name is called a function parameter or simply a parameter.

When you add a parameter to the function definition, you can use it as a variable inside the function body:

The basic syntax for doing this looks as shown below:

```
def functionName(arg1, arg2):
    # What to do with function
```

Below we create a function with one parameter name.

```
def greet(name):
    print("Hi ", name)
```

And you can access the name parameter only within the body of the greet() function, not the outside.

When you call a function with a parameter, you need to pass the information. For example:

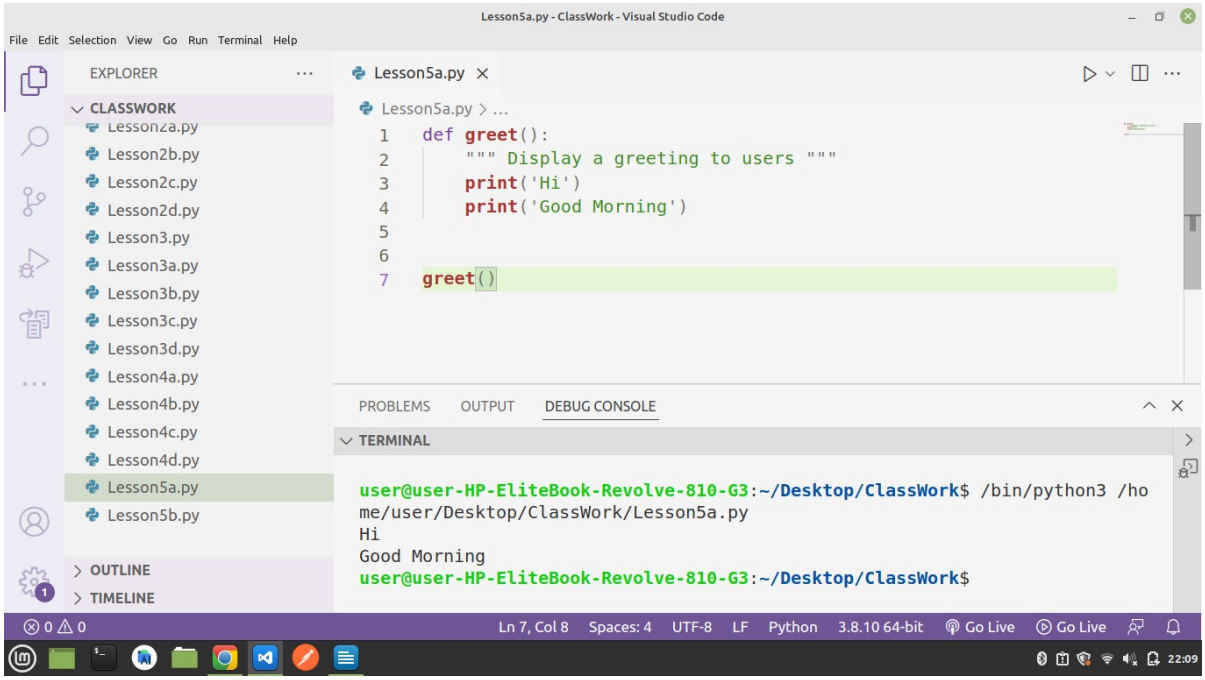
```
greet('John')
```

Output:

```
Hi John
```

Try out this example on VS code

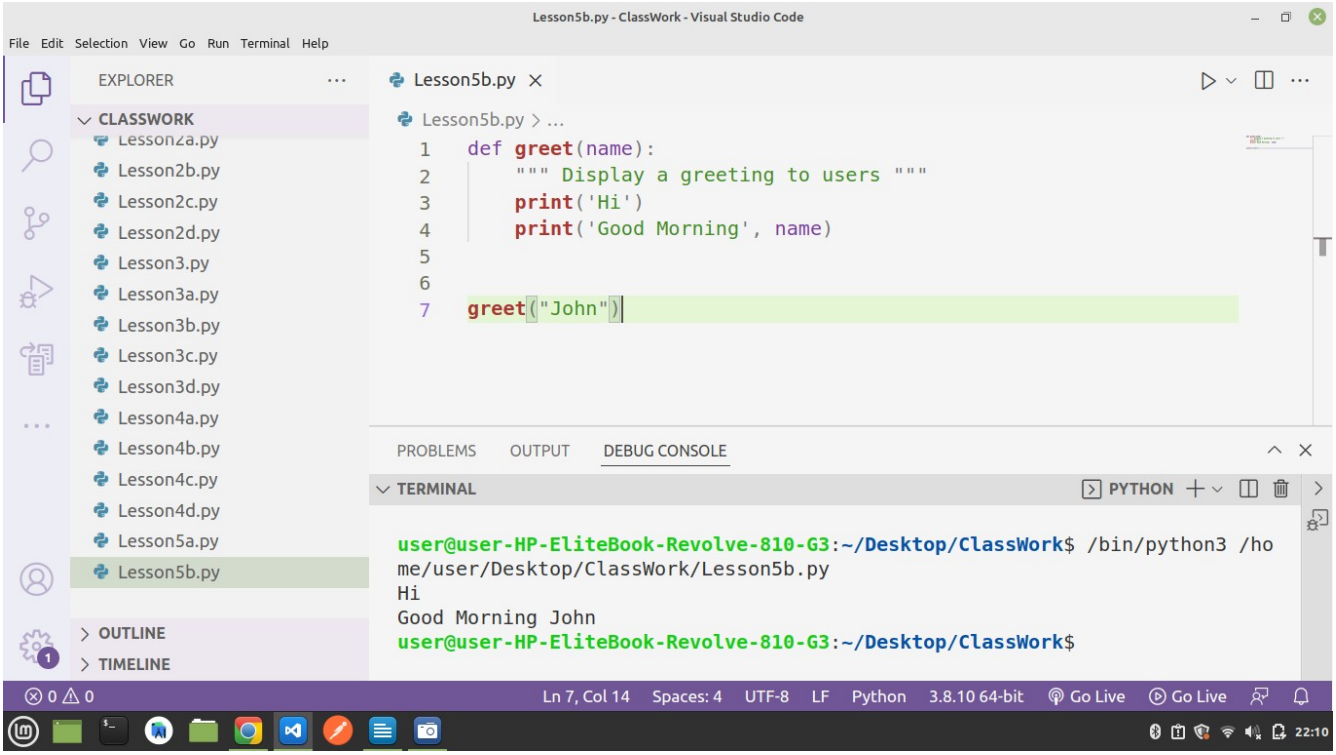
Example: Lesson5a.py



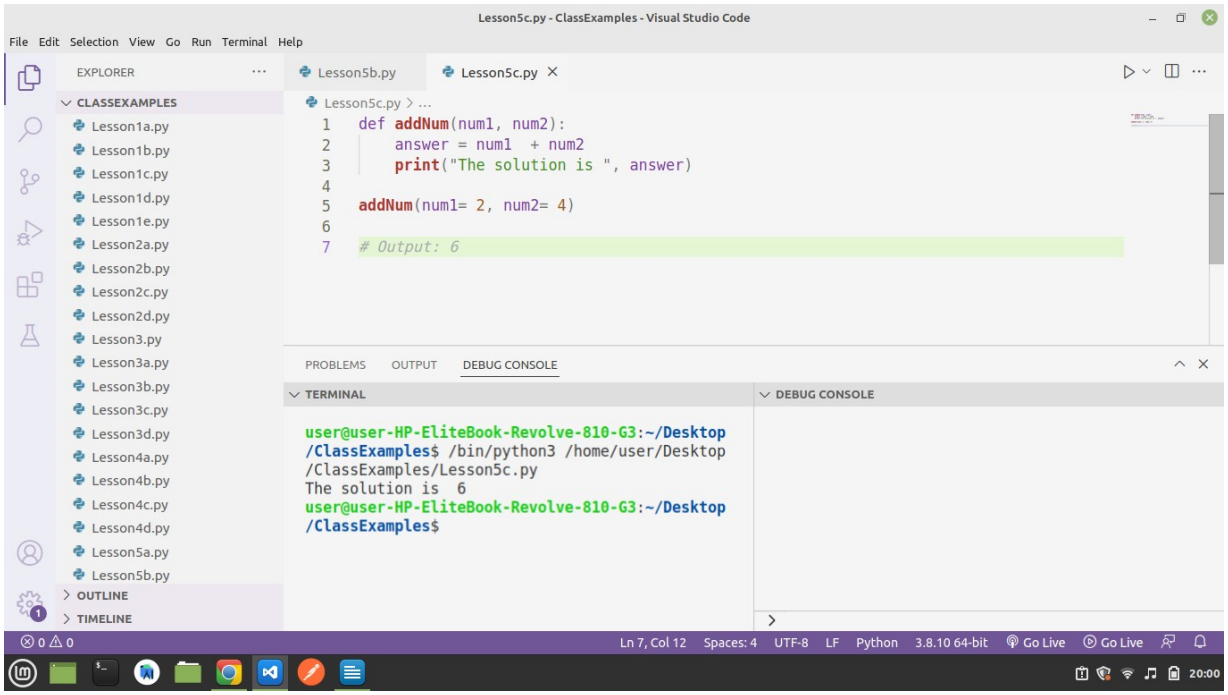
Try out this example on VS code

Example: Lesson5b.py

Below function has parameter **name**



Here's another example of arguments in a Python function:



Python Modules

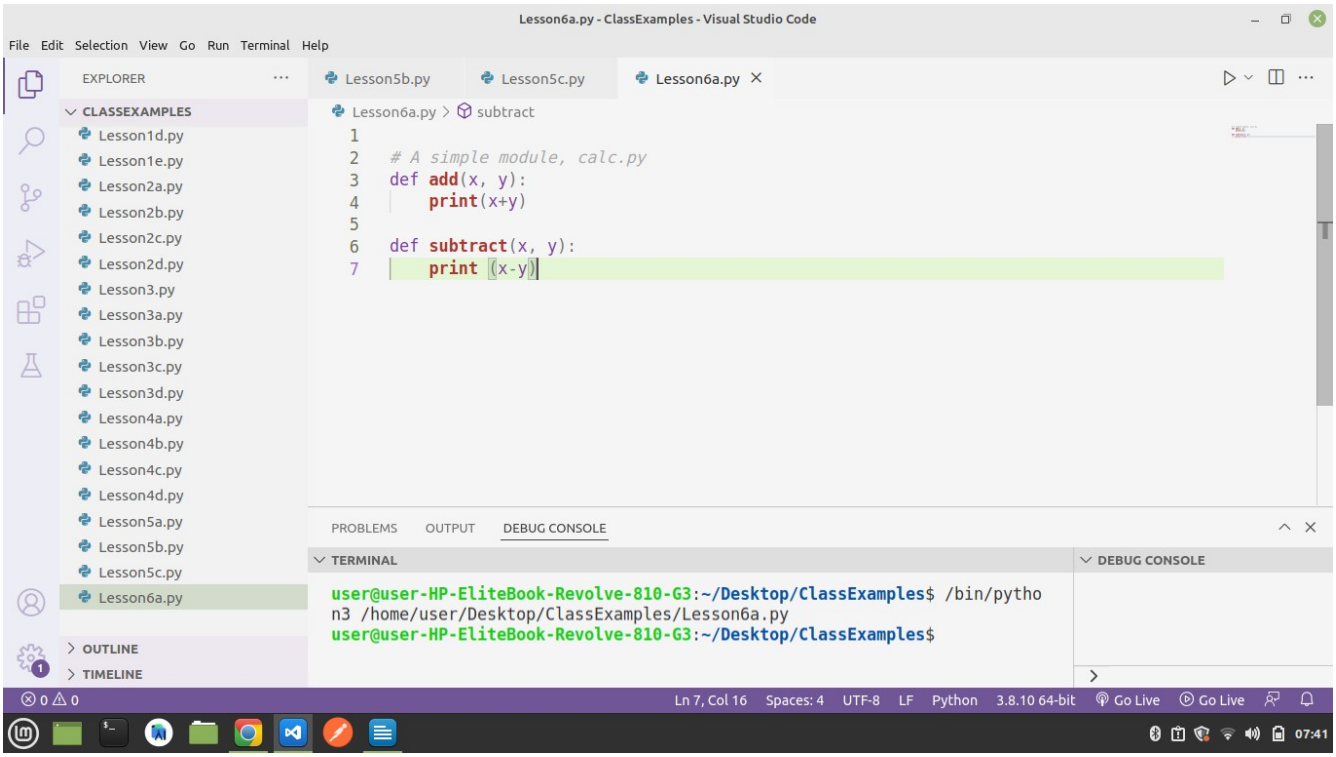
What is Python Module

A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code. Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized.

Create a simple Python module

Let’s create a simple calc.py in which we define two functions, one add and another subtract.

Create a file named **Lesson6a.py**



Import Module in Python

We can import the functions, and classes defined in a module to another module using the **import statement** in some other Python source file.

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches for importing a module. For example, to import the module calc.py, we need to put the following command at the top of the script.

Syntax of Python Import

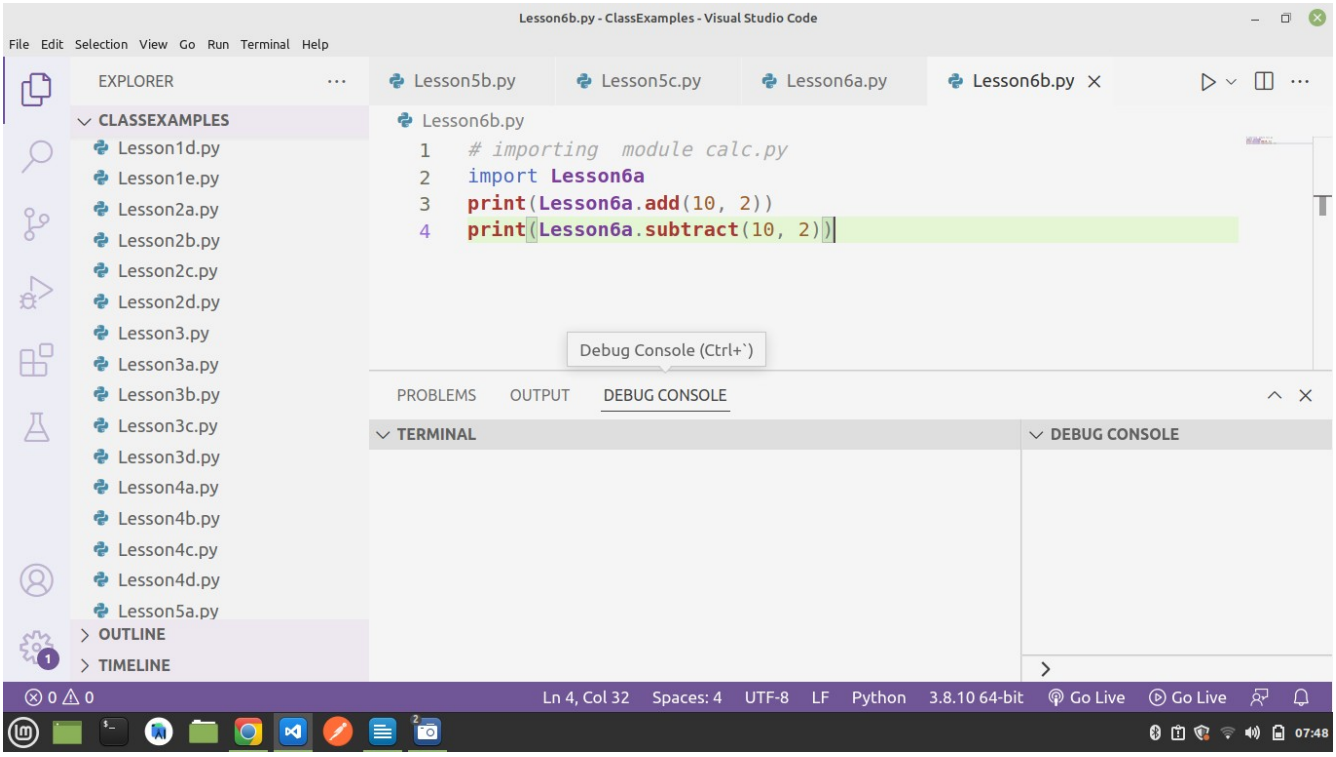
```
import module
```

Note: This does not import the functions or classes directly instead imports the module only. To access the functions inside the module the dot(.) operator is used.

Importing modules in Python

Now, we are importing the Lesson6a that we created earlier to perform add operation.

Create another file named Lesson6b.py



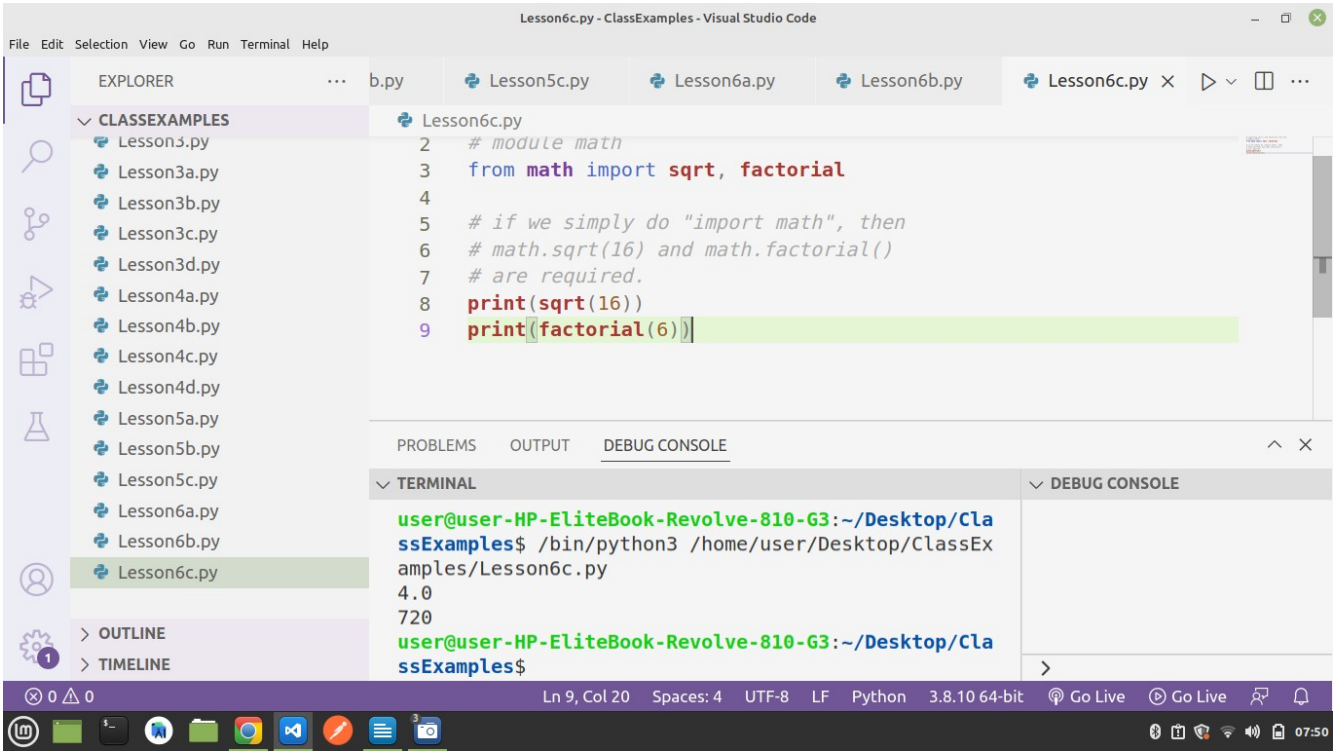
The from-import Statement in Python

Python's *from* statement lets you import specific attributes from a module without importing the module as a whole.

Importing specific attributes from the module

Here, we are importing specific sqrt and factorial attributes from the math module.

Create a File Lesson6c.py, Write below codes



Summary

- A module is a Python source code file with the .py extension. The module name is the Python file name without the extension.
- To use objects from a module, you import them using the import statement.

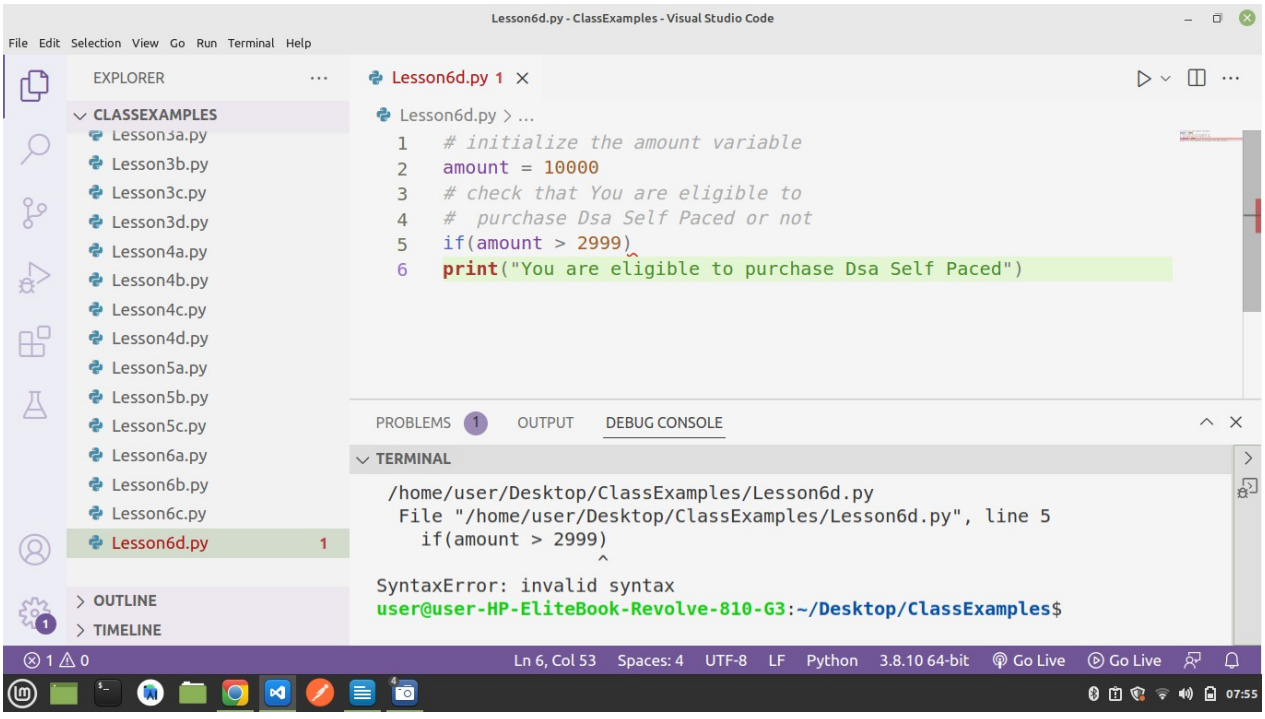
Python Exception Handling

In this lesson, we will discuss how to handle exceptions in Python using try. except, and finally statement with the help of proper examples.

Error in Python can be of two types i.e. **Syntax errors and Exceptions**. Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur which changes the normal flow of the program.

Difference between Syntax Error and Exceptions
Syntax Error: As the name suggests this error is caused by the wrong syntax in the code. It leads to the termination of the program.

Example

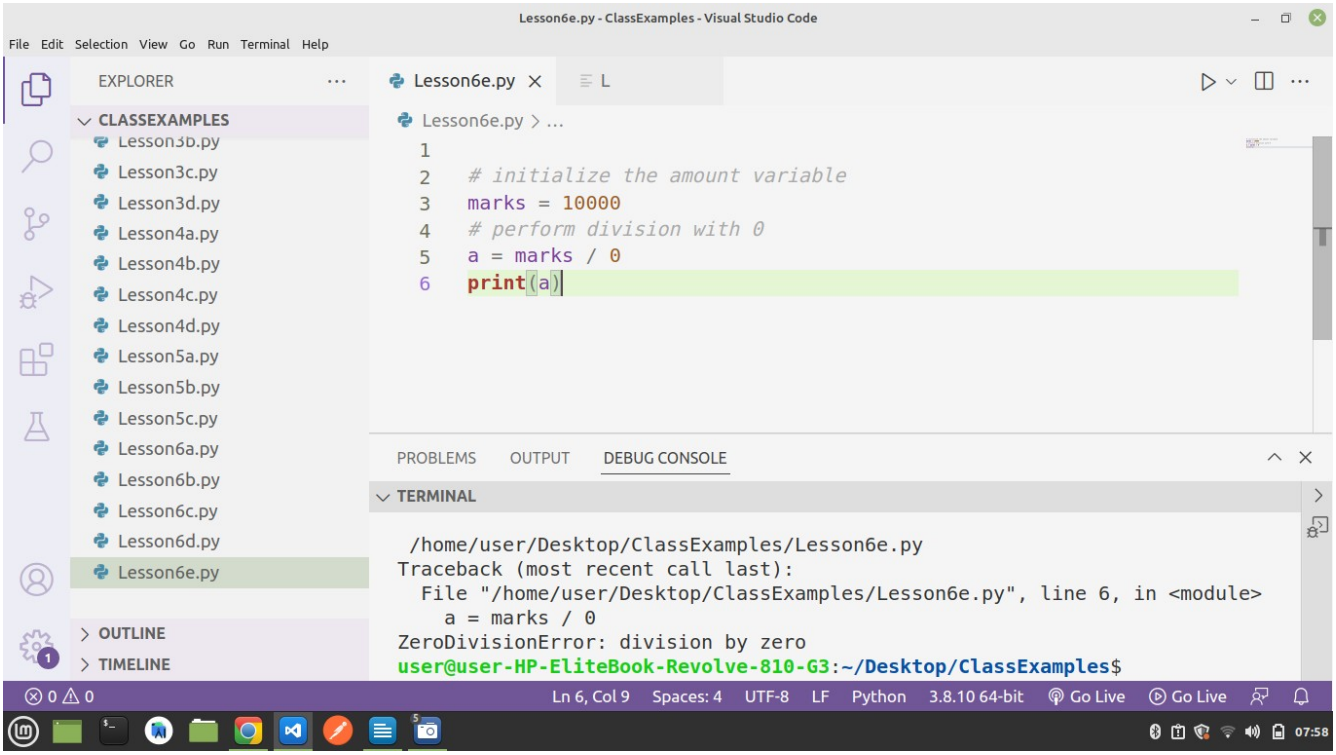


Above program has a syntax error, its missing a Colon at the end of line 5,

It also lacks indentation on Line 6.

Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program, however, it changes the normal flow of the program.

Example

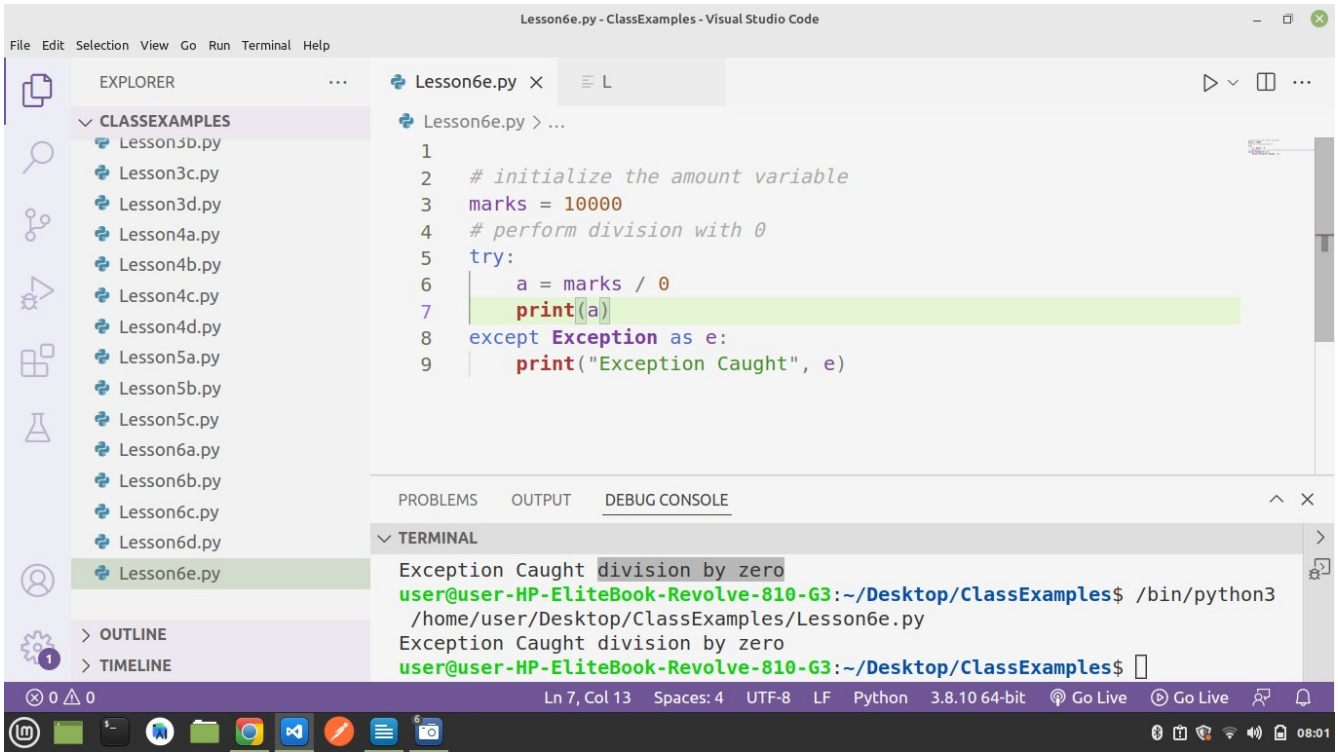


In the above example raised the ZeroDivisionError as we are trying to divide a number by 0.

Try and Except Statement – Catching Exceptions

Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

Example



Above we put a Try on Line 5, to protect an exception being triggered on Line 6, If an exception is thrown its caught on Line 9. If an exception is not thrown then Line 7 Runs.

Summary

- Use Python **try...except** statement to handle exceptions gracefully.
- Use specific exceptions in the **except** block as much as possible.
- Use the except Exception statement to catch other exceptions.

Appendix 1

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

Questions

1. Create a Program to Find Simple Interest

2. Create a Program to Find Body Mass Index

3. Create a Program to Find Area of a Triangle.

4. Create a Program to Find Area of a Circle

5. **Given below dictionary**

```
shopping = {
    'sugar': 120,
    'rice': 200,
    'milk': 60,
    'bread': 60
}
```

```
Do the following
1. Print this dictionary
2. Find the sum of all items in above dictionary.
```

6. Please attempt the following question:
   Suppose we have a dictionary:

person = {'john': 40, 'peter': 45}

What happens when we try to retrieve a value using the expression print(person['susan']).

Please show your working on how you arrived to your choice below, give one choice below
a) Since “susan” is not a value in the set, Python raises a KeyError exception
b) It is executed fine and no exception is raised, and it returns None
c) Since “susan” is not a key in the set, Python raises a KeyError exception
d) Since “susan” is not a key in the set, Python raises a syntax error

7. Create a List of town and reverse it.

8. Create a Loop to Print from 10 to 40

9. Create a Loop to Print from -10 to -50

10. Create a Function to Check if Year is Leap or Not

11. Create a Function to Find Body Mass Index.

12. Using if statements, write a Python program to check how much a traveler would pay based of distance.

Distance	Cost
0 through 100	5.00
More than 100 but not more than 500	8.00
More than 500 but less than 1,000	10.00
1,000 or more	12.00

13. Create a Python Program to find if number is Odd or Even

References

https://www.geeksforgeeks.org/python-programming-language/

https://www.w3schools.com/python/

https://www.tutorialspoint.com/python/index.htm

https://www.guru99.com/python-tutorials.html