**VAISHALI BOKADIYA  
DAY 7**

**PYTHON BASICS ASSESSMENT**

**Python and its Features:**

Python is a high-level, general-purpose programming language. It is an interpreted and dynamically typed language.

It is used in:

* web development
* software development
* data science
* system scripting

Features of python:

* Easy to Code
* Interpreted Language
* Free and Open Source
* Object-Oriented Language
* Cross-Platform Language
* Extensive Feature
* High-Level Language

**Variables & Operators:**

**Variables:**

Python is a dynamically typed language i.e. we don’t specify the data type of the variable while declaration.

Example:

x=10

name=’Vaishali Bokadiya’

If we want to change a variable into a particular data type then we can use type casting.

Example:

x=str(10)

y=float(3)

Variable naming rules in python:

* Variable names must start with a letter or the underscore.
* Variable names can’t start with a number.
* Variable names can only contain alpha-numeric characters and underscores.
* Variable names are case-sensitive.
* Variable names cannot be any of the Python keywords.

**Operators:**

Python has the following types of operators:

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

**Arithmetic operators:** Arithmetic operators are used with numeric values to perform mathematical operations such as addition, multiplication, subtraction, division etc.

Examples of arithmetic operators are +, -, \*, /, %, \*\*, //.

**Assignment operators:** Assignment operators are used to assign values to variables. Examples of assignment operators are =, +=, -=, \*=, /=, \*\*=, //=, %= etc.

**Comparison operators:** Comparison operators are used to compare two values. Examples of comparison operators are >, <, ==, >=, <=, !=.

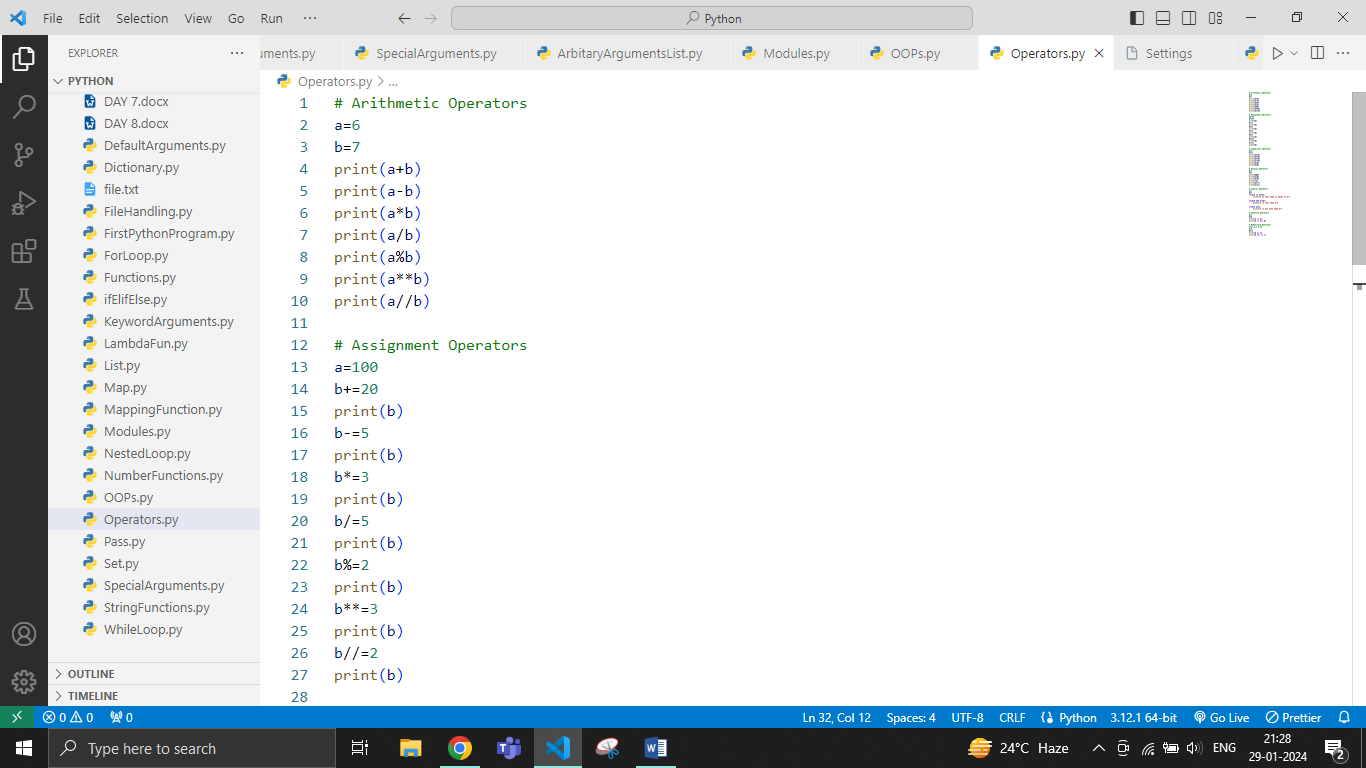
**Logical operators:** Logical operators are used to combine conditional statements. Examples of logical operators are ‘and’, ‘or, and ‘not’.

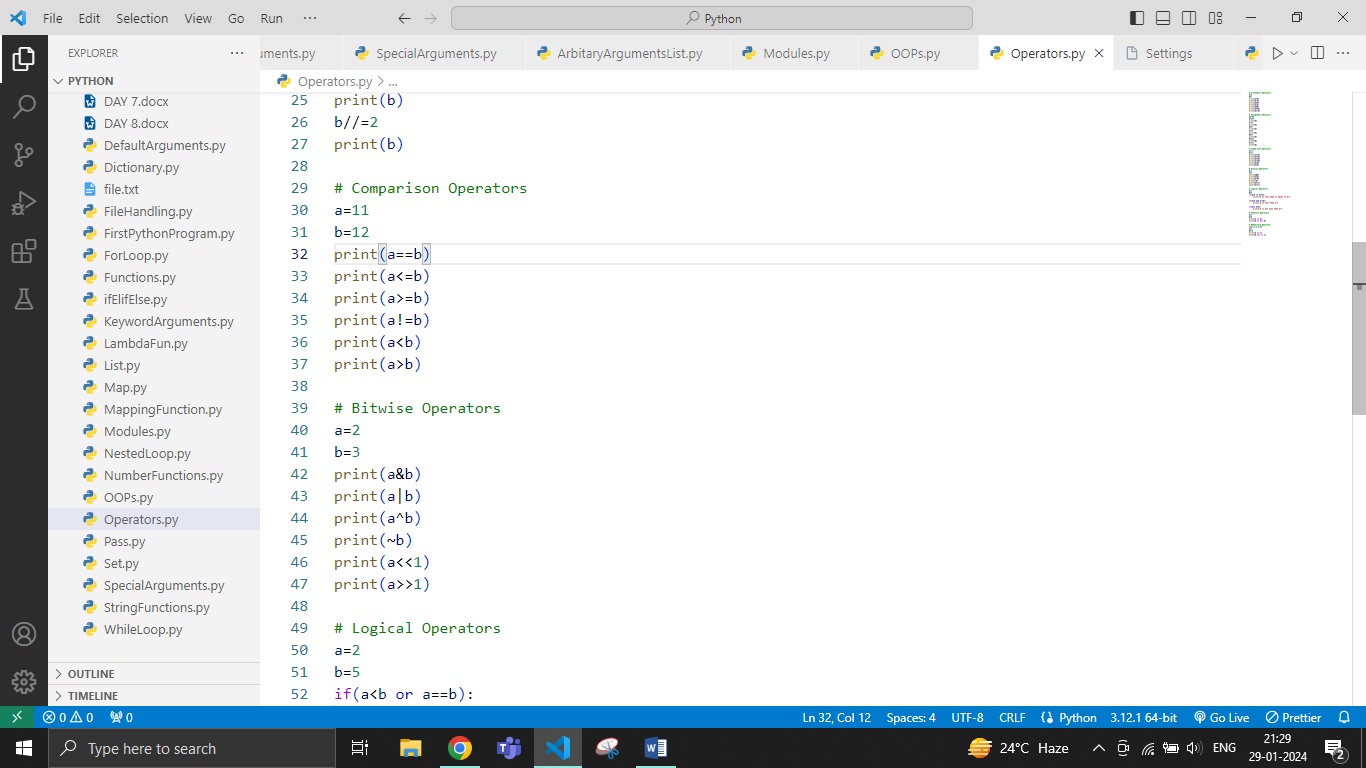
**Identity operators:** Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location. Examples of identity operators are ‘is’ and ‘is not’.

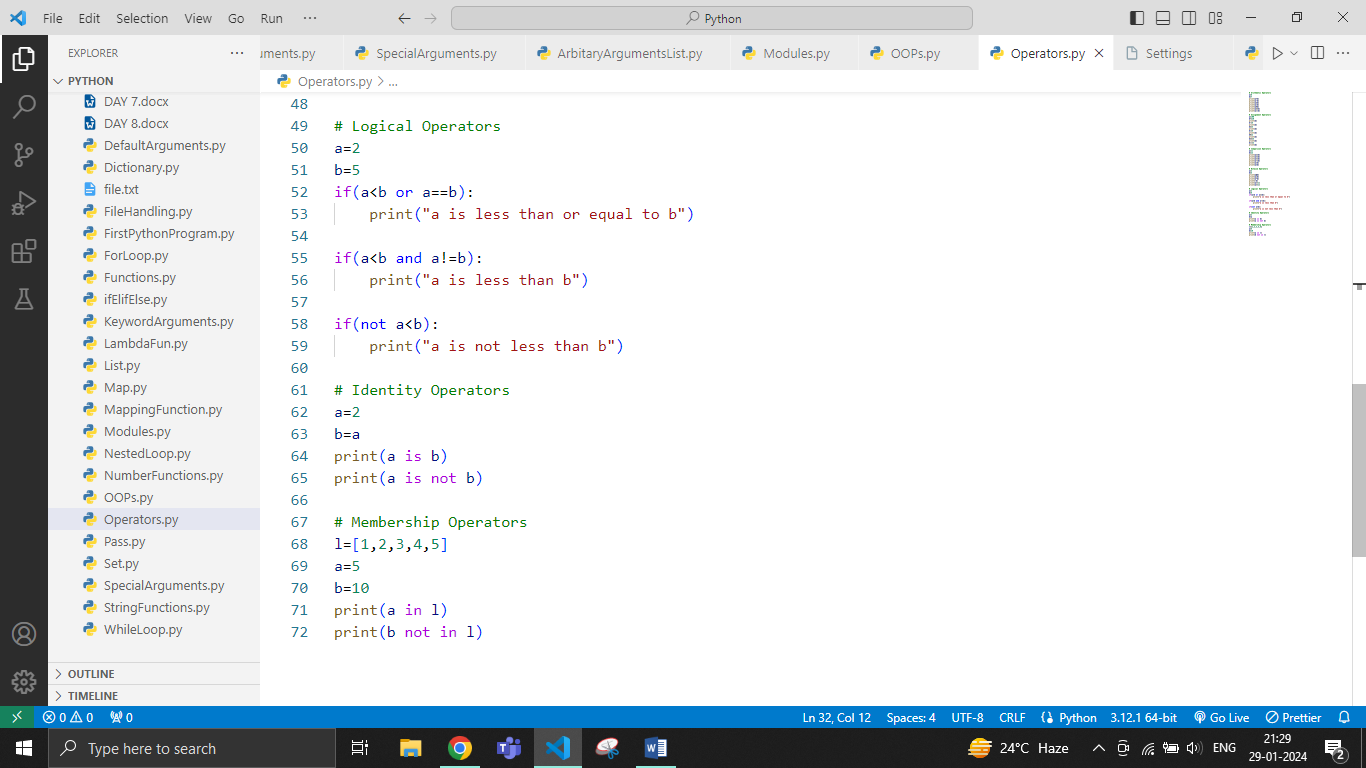
**Membership operators:** Membership operators are used to test if a sequence is presented in an object. Examples of membership operators are ‘in’ and ‘not in’.

**Bitwise operators:** Bitwise operators are used to compare binary numbers. Examples of bitwise operators are &, |, ^, ~, << and >>.

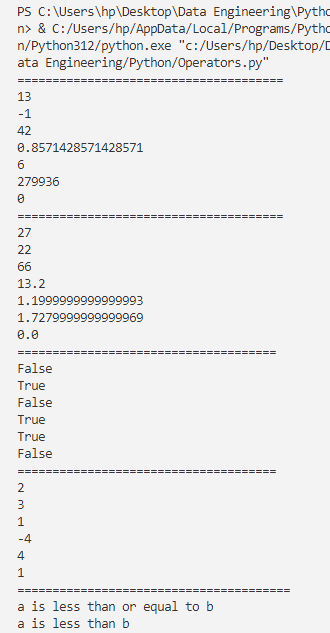
Code:

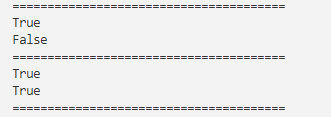






Output:





**Data Types:**

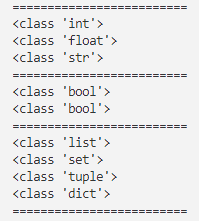
Python has the following datatypes:

* Numeric
  + int: integer type
  + float: with decimal point
* String
* Boolean
  + bool: true or false
* Sequence
  + list: collection of data
  + set: collection of unique elements
  + tuple: collection of data
  + dict: with key-value pair

Code:



Output:



**If statement:**

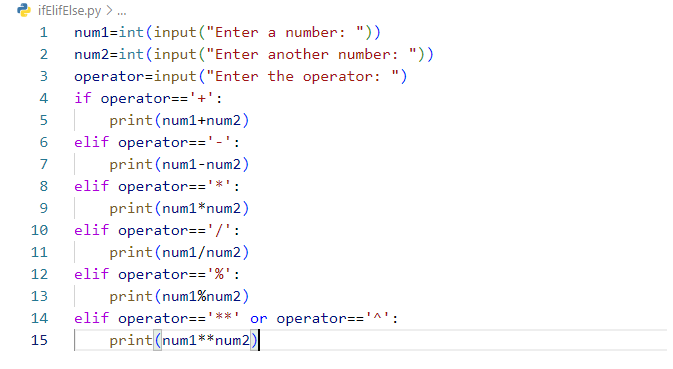
Conditional statement that executes a block of code if a specified condition is true.

**If-else statement:**

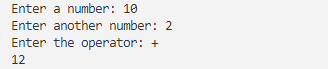
Conditional execution of code, where a specified block is executed if a condition is true, and an alternative block is executed if the condition is false.

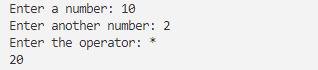
**If-elif-else statement:**

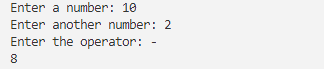
Conditional execution by checking multiple conditions in order, and executing the corresponding block of code associated with the first true condition, or the else block if none of the conditions are true.

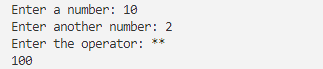
Code:

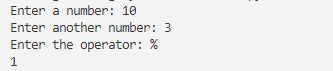
Output:

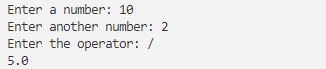








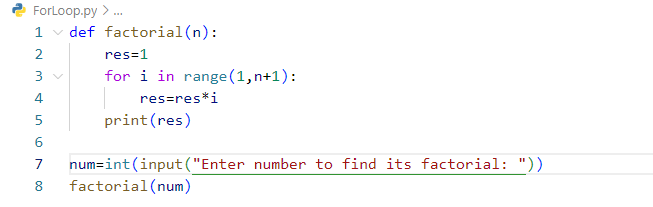




**For Loop:**

For loop in Python iterates over a sequence (such as a list, tuple, or string) and executes a block of code for each element in the sequence.

Code:



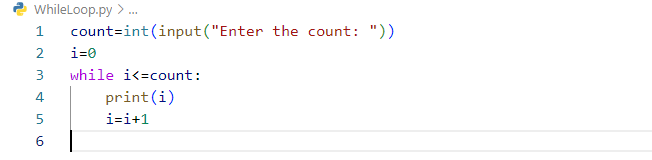
Output:



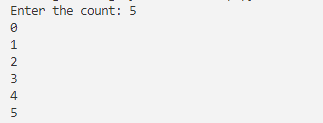
**While Loop:**

A while loop in Python repeatedly executes a block of code as long as a specified condition is true.

Code:



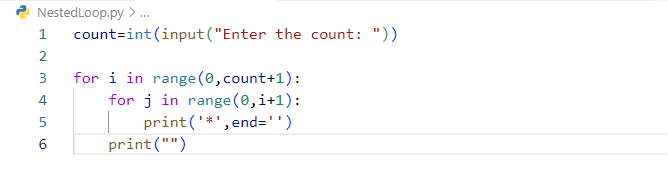
Output:



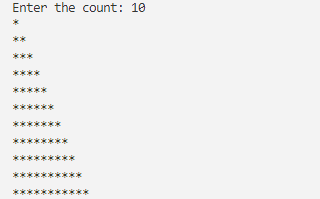
**Nested Loop:**

Nested loops in Python are loops within loops, where the inner loop runs for each iteration of the outer loop, allowing for more complex iteration structures.

Code:



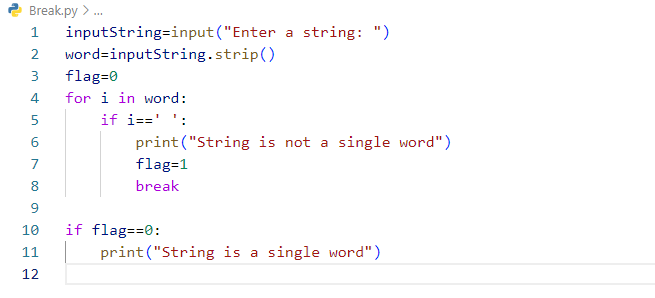
Output:



**Break, Continue and Pass:**

**Break:**

Break statement is used to terminate the loop's execution before it reaches its natural end.

Code:

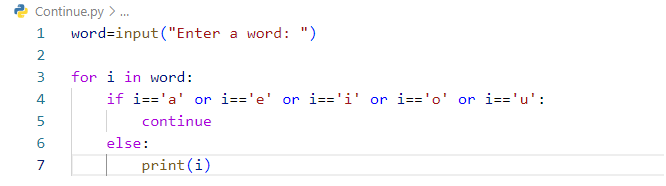
Output:



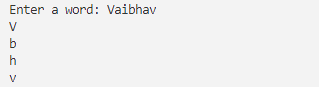
**Continue:**

Continue in Python is a keyword that is used to skip the rest of the code inside a loop and move to the next iteration.

Code:

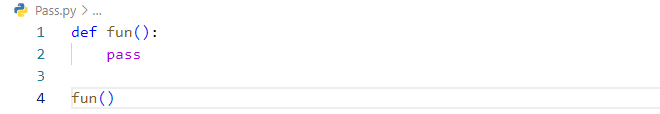


Output:



**Pass:**

Pass statement is a null statement that serves as a placeholder for syntactically required code blocks with no actual functionality.



**String Functions:**

**1. len()**: Returns the length of the string.

**2. lower()**: Converts all characters in a string to lowercase.

**3. upper():** Converts all characters in a string to uppercase.

**4. capitalize():** Capitalizes the first character of a string.

**5. count():** Returns the number of occurrences of a substring in the string.

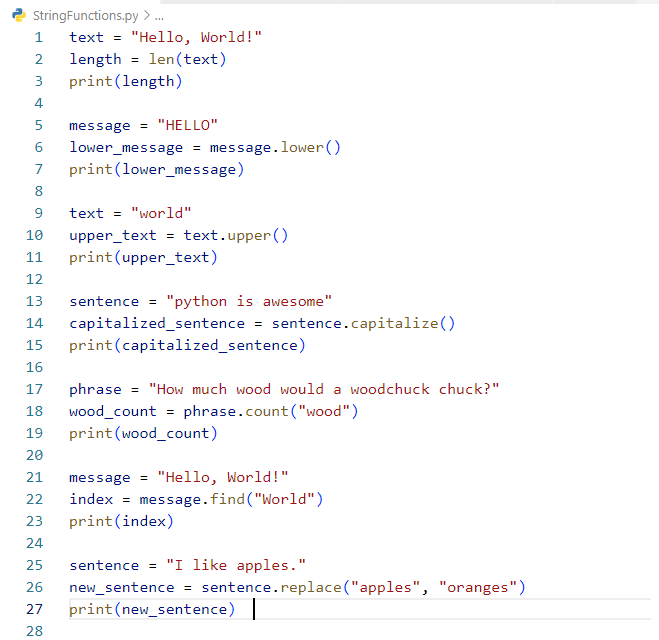
**6. find():** Returns the lowest index of a substring in the string. Returns -1 if the substring is not found.

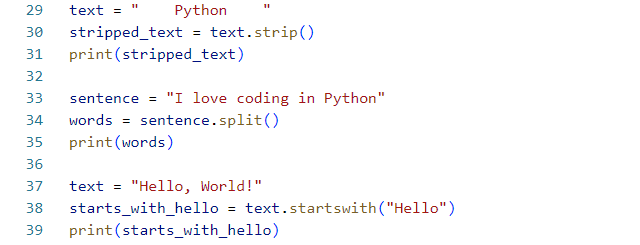
**7. replace():** Replaces a substring with another substring.

**8. strip():** Removes leading and trailing whitespaces from a string.

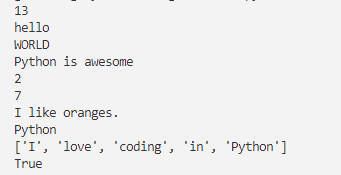
**9. split():** Splits a string into a list of substrings based on a specified delimiter.

**10. startswith():** Checks if a string starts with a specified prefix.

Code:



Output:



**Number Functions:**

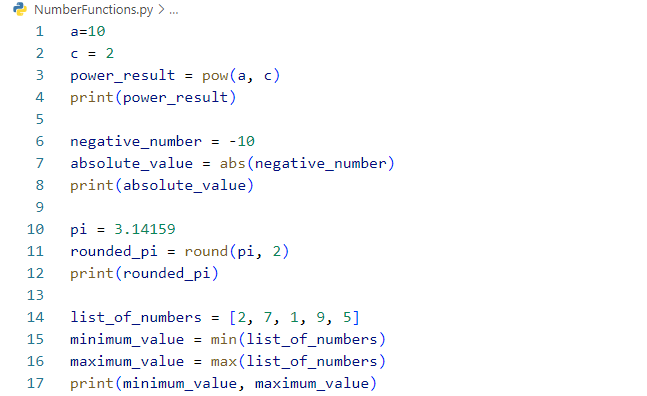
**1. pow():** Power or exponential

**2. abs():** Absolute value

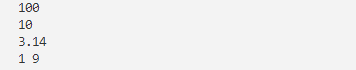
**3. round():** Round to a specific number of decimals

**4. min() and max():** Minimum and Maximum

Code:



Output:



**Date functions:**

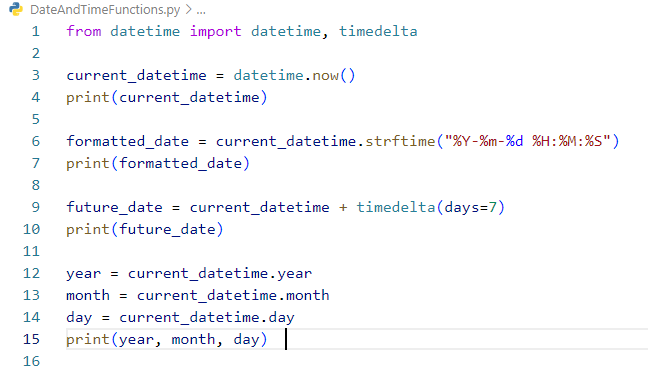
**1.datetime.now():** Get the current date and time

**2.strftime():** Formatting a date as a string

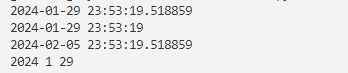
**3.timedelta():** Adding days to a date

**4. .year, .month, .day:** Get components of a date (year, month, day)

Code:



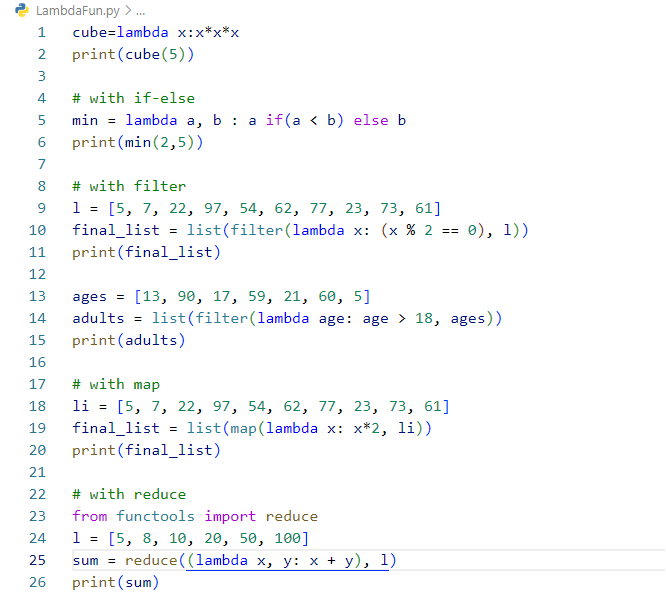
Output:



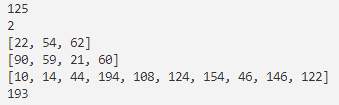
**Lambda Expression:**

A lambda expression in Python is an anonymous function defined using the lambda keyword, typically used for short, simple operations.

Code:



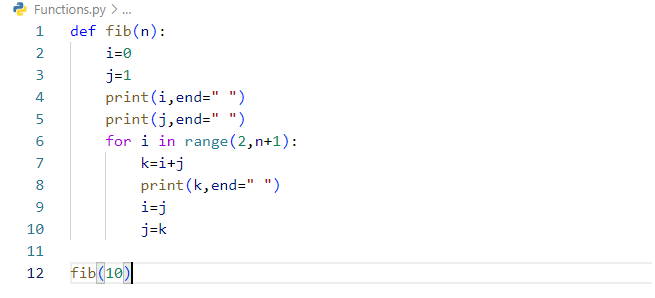
Output:



**Functions:**

Functions in Python are like mini-programs that perform a specific task, making it easy to reuse code by allowing you to call them with a single line of code.

Code:



Output:

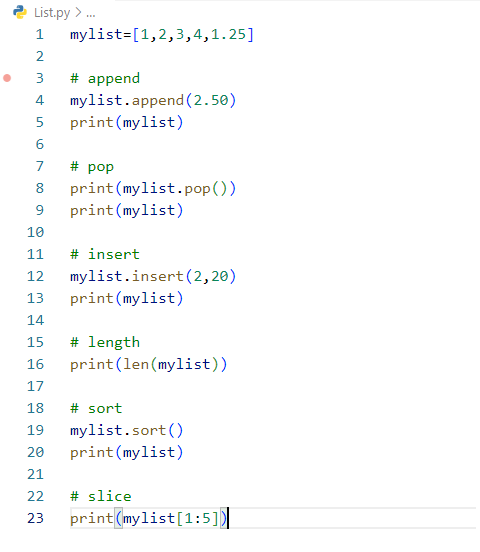


**List and list methods:**

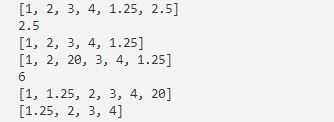
A list in Python is a collection of ordered elements, allowing you to store and manipulate multiple values in a single variable.

* Lists are ordered.
* Lists are heterogeneous.
* Lists are mutable.

Code:



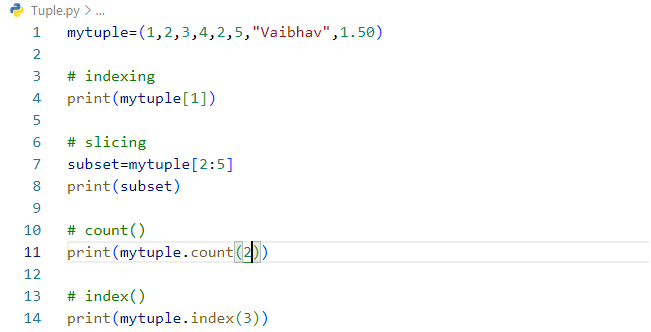
Output:



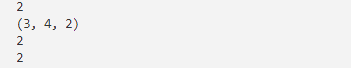
**Tuple:**

A tuple in Python is an ordered collection of elements, similar to a list, but immutable, meaning its values cannot be changed after creation.

* Tuples are ordered.
* Tuples are heterogeneous.
* Tuples are immutable.

Code: 

Output:



**Dictionary:**

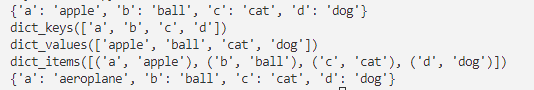
A dictionary in Python is a collection of key-value pairs, where each key must be unique and is used to access its corresponding value.

* Dictionary is unordered.
* Dictionary is mutable.

Code:



Output:

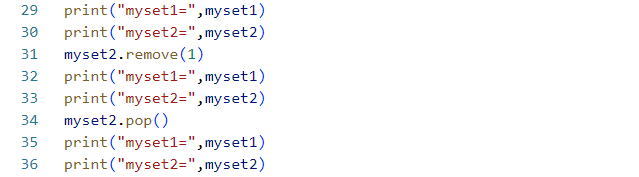


**Sets:**

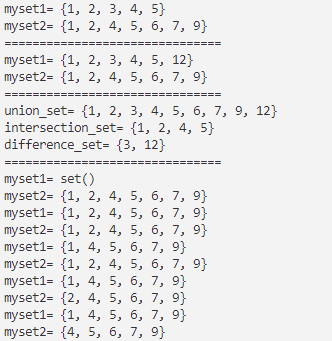
A set in Python is an unordered collection of unique elements.

* Sets are unordered.
* Sets are heterogeneous.
* Sets are mutable.

Code:



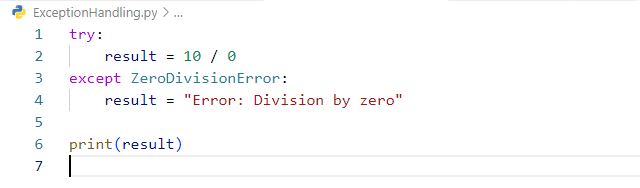
Output:



**Exception Handling:**

Exception handling in Python is a way to manage and respond to errors in your code, preventing it from crashing by using try, except, and optionally, finally blocks.

Code:



Output:



**OOP:** Object-oriented Programming is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.

* **Class:** A class is a blueprint of objects.
* **Object:** The object is an entity that has a state and behaviour associated with it.
* **Inheritance:** Inheritance is the capability of one class to derive or inherit the properties from another class.
* **Polymorphism:** Polymorphism allows functions and operators to have same name but different signatures.
* **Encapsulation:** Encapsulation describes the idea of wrapping data and the methods that work on data within one unit.

Code:

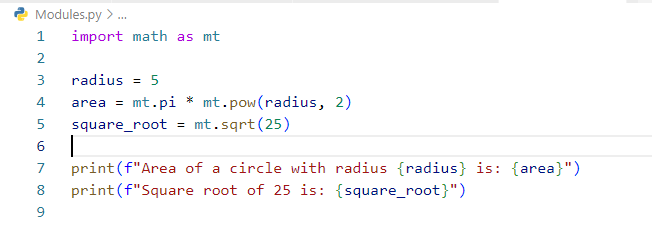


Output:



**Modules and packages:**

Modules are individual files containing Python code, while packages are collections of modules organized in directories.

Code:

Output:

