



Functions and Modules



Pre-requisites

Hope you have gone through the self-learning content for this session on the PRISM portal.



By the End of this Session:

- Learn how to perform repetitive tasks using functions in Python.
- Understand the various components in the function signature.
- Understand the different types of function arguments.
- Create and use Python modules to store several functions in a single file.

What have You Learned So Far?

- Complex data types – Dictionary and Set
- Operators in Python – Arithmetic, Comparison, Logical, and Assignment
- Control flow statements in Python
- IF-ELIF-ELSE statements
- Looping constructs in Python – FOR and WHILE loop

Poll Time

Q. Consider the below Python code snippet. What will be the output of the code?

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
total = 0
for num in numbers:
    if num % 2 == 0:
        total += num
print(total)
```

- a. 30
- b. 20
- c. 25
- d. 15



Poll Time

Q. Consider the below Python code snippet. What will be the output of the code?

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
total = 0
for num in numbers:
    if num % 2 == 0:
        total += num
print(total)
```

- a. 30
- b. 20
- c. 25
- d. 15

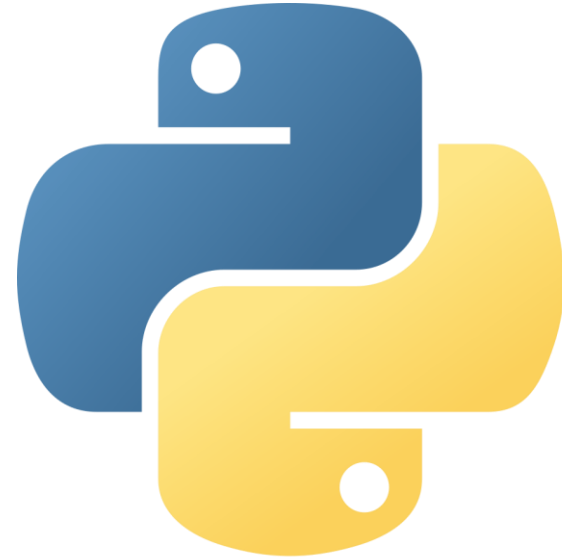
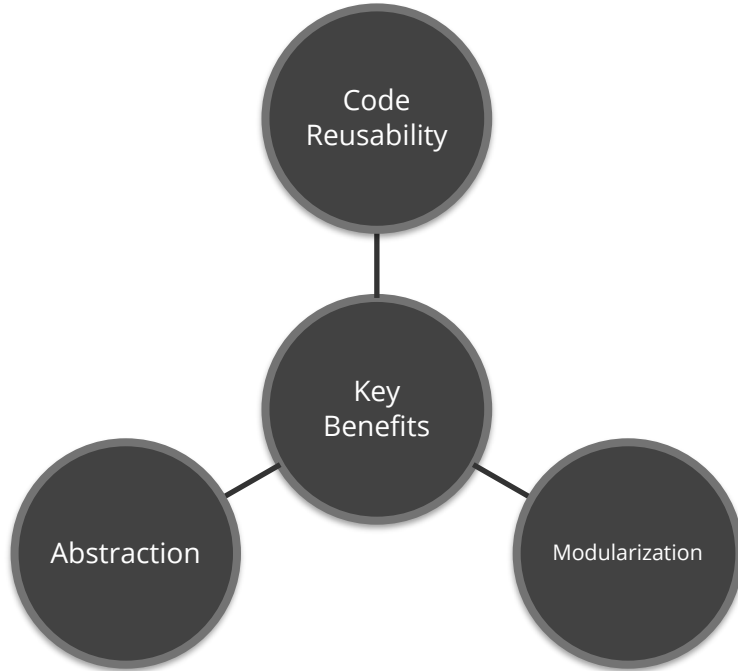




Functions

Introduction to Functions in Python

A function is a reusable block of code that performs a specific task. They help organize code and promote reusability.



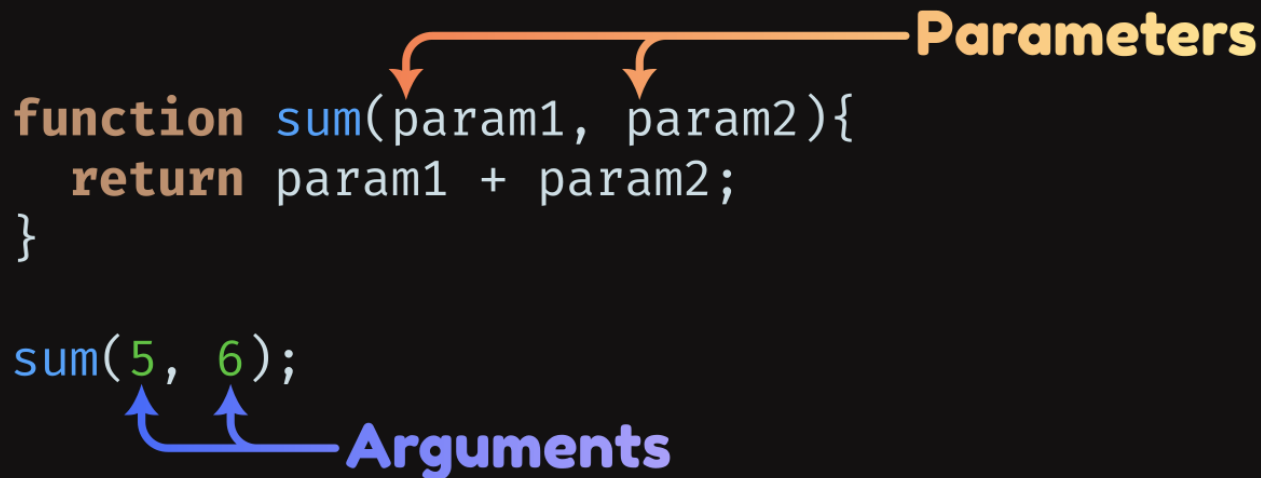
Defining and Calling Functions

```
def function_name(parameters):  
    # Function code block  
    return result
```

Function Components

- **def**: Keyword to define a function.
- **function_name**: Name of the function, follow naming conventions.
- **parameters**: Optional input data for the function.
- **::**: Colon indicates the start of the function's code block.
- **return**: Keyword to specify the function's output (optional).

Parameters and Arguments in Functions



The diagram illustrates the relationship between function parameters and arguments. It features two code snippets on a dark background. The first snippet is a function definition: `function sum(param1, param2){` followed by an indented `return param1 + param2;` and a closing brace. The second snippet is a function call: `sum(5, 6);`. An orange arrow originates from the word **Parameters** on the right and points to the `param1` and `param2` identifiers in the function signature. A blue arrow originates from the word **Arguments** at the bottom and points to the values `5` and `6` in the function call.

```
function sum(param1, param2){  
    return param1 + param2;  
}
```

sum(5, 6);

Parameters

Arguments

Pop Quiz

Q. What is the purpose of using functions in Python?

- a. To define a sequence of actions that are executed only once
- b. To create a reusable block of code that performs a specific task
- c. To handle errors and exceptions in Python programs
- d. To make decisions based on specific conditions



Pop Quiz

Q. What is the purpose of using functions in Python?

- a. To define a sequence of actions that are executed only once
- b. To create a reusable block of code that performs a specific task**
- c. To handle errors and exceptions in Python programs
- d. To make decisions based on specific conditions



Returning Values from Functions

Functions can return a value to the caller. You use **return** statement to specify the value to be returned. The returned value can be assigned to a variable.

```
def add_numbers(a, b):  
    return a + b  
  
result = add_numbers(10, 5)
```

Positional and Keyword Arguments

Positional Arguments

- Arguments passed based on their position.
- The order of arguments matters.
- Matched to parameters in the order they appear.

Keyword Arguments

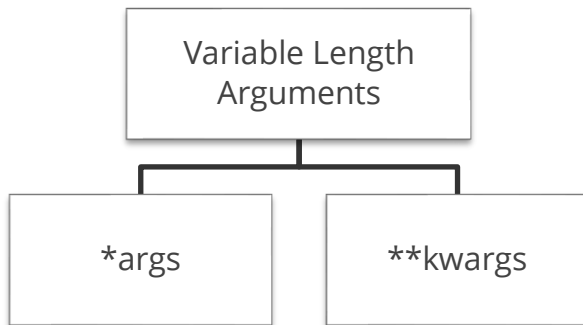
- Arguments passed with parameter names.
- Order does not matter.
- Easier to identify which value corresponds to which parameter.

```
def greet(first_name, last_name):  
    return "Hello, " + first_name + " " + last_name
```

```
message = greet("John", "Doe")
```

```
message = greet(first_name="John", last_name="Doe")
```

Variable-length Arguments



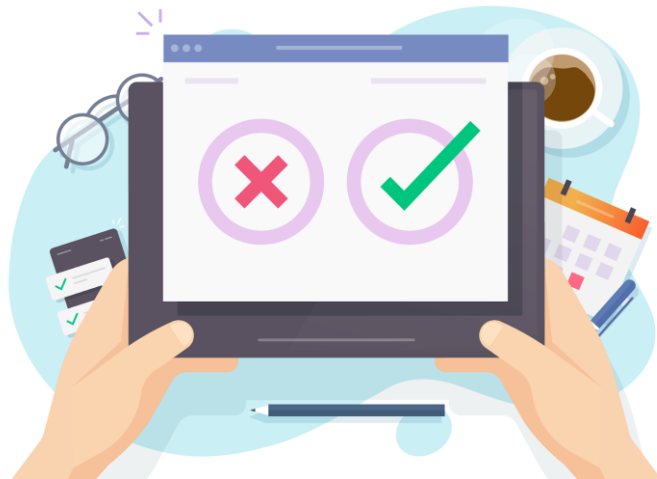
```
def calculate_sum(*args):  
    sum = 0  
    for num in args:  
        sum += num  
    return sum  
  
result = calculate_sum(1, 2, 3, 4, 5)
```

```
def print_info(**kwargs):  
    for key, value in kwargs.items():  
        print(key, ":", value)  
  
print_info(name="John", age=30, city="New York")
```

Poll Time

Q. Which type of function arguments in Python allows passing multiple positional arguments without specifying parameter names?

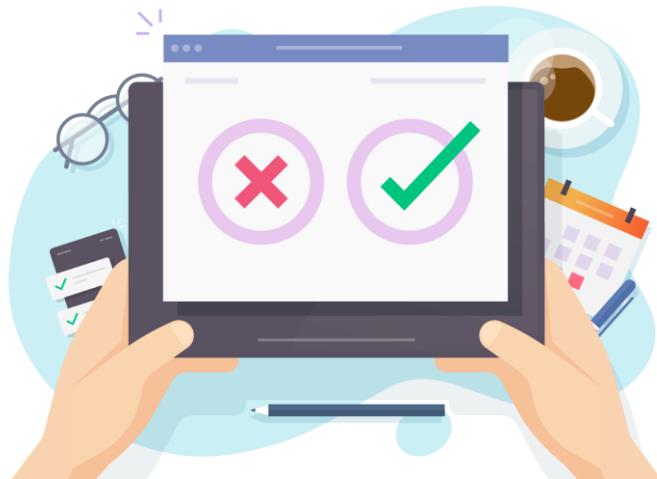
- a. Regular Arguments
- b. Keyword Arguments
- c. Variable-length Arguments (*args)
- d. Default Arguments



Poll Time

Q. Which type of function arguments in Python allows passing multiple positional arguments without specifying parameter names?

- a. Regular Arguments
- b. Keyword Arguments
- c. Variable-length Arguments (*args)**
- d. Default Arguments








Modules

Introduction to Modules in Python

Modules are Python files containing reusable code. They allow organizing functions, classes, and variables.

Module: greet.py

python

 Copy code

```
def say_hello(name):  
    return "Hello, " + name + "!"  
  
def say_goodbye(name):  
    return "Goodbye, " + name + "!"
```

User-defined and Built-in Modules

Step 1

- Open a Text Editor

Step 2

- Write Your Python Code

Step 3

- Save the File with a .py Extension

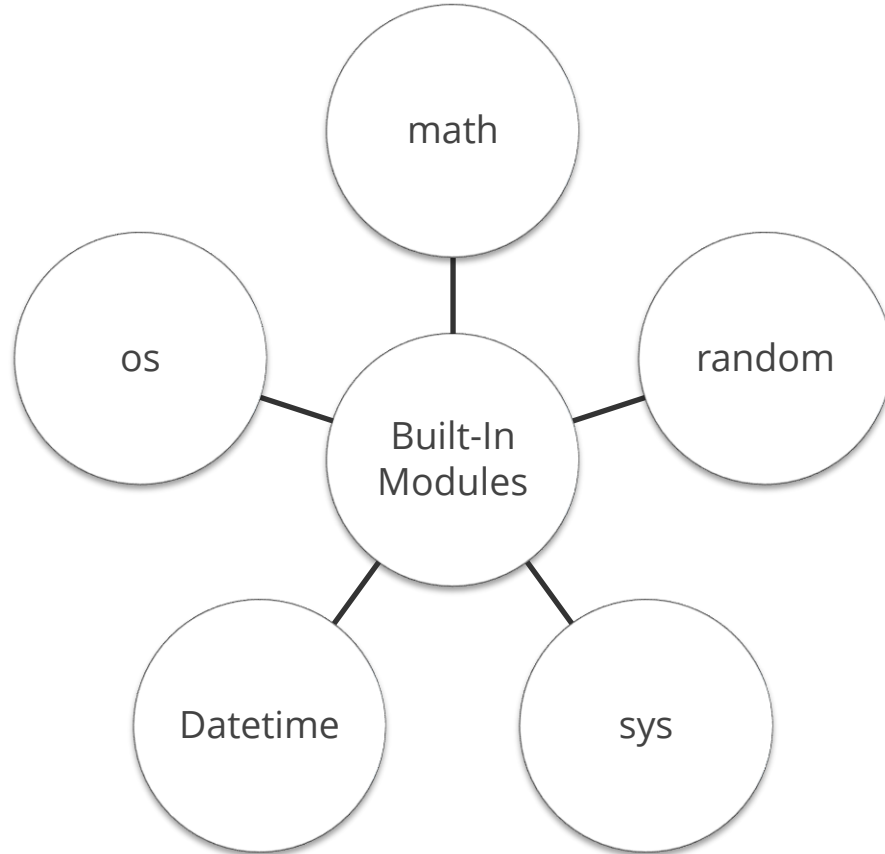
Step 4

- Define Your Module Functions and Code

Step 5

- Import Your Module in Other Python Scripts


User-defined and Built-in Modules



Aliasing and Renaming Modules

Module aliasing involves giving a shorter or alternate name (alias) to a module. Useful when module names are long or to avoid naming conflicts. Use **as** keyword to create alias.

python

 Copy code

```
import math as m  
result = m.sqrt(25)
```

Pop Quiz

Q. Which of the following statements is correct when importing a module in Python?

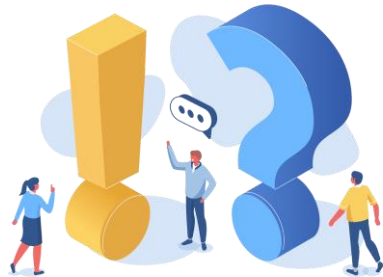
- a. Modules can only be imported with their full names
- b. Module names cannot be changed while importing
- c. Modules cannot be imported from external sources
- d. Aliasing allows giving shorter names to modules while importing



Pop Quiz

Q. Which of the following statements is correct when importing a module in Python?

- a. Modules can only be imported with their full names
- b. Module names cannot be changed while importing
- c. Modules cannot be imported from external sources
- d. Aliasing allows giving shorter names to modules while importing**





Summary

- ✓ A function is a reusable block of code that performs a specific task.
- ✓ In Python, functions support both positional and keyword arguments.
- ✓ Modules are Python files containing reusable code.
- ✓ Aliases help import modules with long names and avoid naming conflicts.

Activity 1

Pre-requisites:

- Python 3.x – preferably Python 3.8
- Jupyter Notebook

Scenario:

Continue practicing the basics of Python. Perform the below operations:

- Write a Python function to reverse the contents in a list.
- Write a Python function that returns the count of each item in a list. The output of the function should be a dictionary with format {item: count}
- Create a python module called **comparison.py**. Use this module to define a function for running comparison operations such as less_than(), greater_than(), equal_to() etc.

Next Session:

Lambda Functions, File Handling, and Error Handling

THANK YOU!

Please complete your assessments and review the self-learning content for this session on the **PRISM** portal.



knowledgehut
upGrad



File Handling, Lambda Functions, and Error Handling



Pre-requisites

Hope you have gone through the self-learning content for this session on the PRISM portal.



By the End of this Session, You Will:

- Use File Handling to manipulate text files.
- Read data from text files into Python strings.
- Write and Append data to text files.
- Use Lambda functions to create Python functions.
- Learn TRY and EXCEPT statements to perform error handling.



Recap

Pop Quiz

Q. What are modules in Python?

- a. A module is a collection of built-in functions in python
- b. A module is a type of variable used in Python programs
- c. A module is a file containing Python code that can be reused in other Python programs
- d. A module is a type of loop used for repetitive tasks in Python



Pop Quiz

Q. What are modules in Python?

- a. A module is a collection of built-in functions in python
- b. A module is a type of variable used in Python programs
- c. A module is a file containing Python code that can be reused in other Python programs**
- d. A module is a type of loop used for repetitive tasks in Python

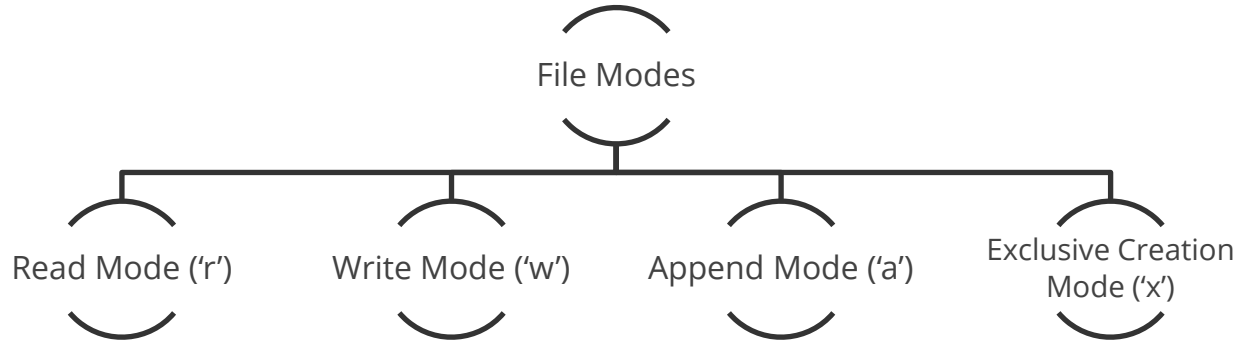




File Handling

Introduction to File Handling in Python


- File handling involves working with files on the computer.
- Reading, writing, and manipulating data in files.



Opening and Closing Files

- Use the **open()** function to open a file
- Syntax: file = **open("filename", "mode")**
- Use the **close()** method to close the file

python


 Copy code

```
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()
```

Reading Data from Files: read() and readlines()


- read() - Reads the entire content of the file as a single string.
- readlines() - Reads the file line by line and returns a list of lines.

python

 Copy code

```
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

python


 Copy code

```
with open("example.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        print(line.strip()) # Use strip() to remove newline characters.
```

Writing Data to Files: write() and writelines()


- write() - Writes a string to the file.
- writelines() - Writes a list of strings to the file.

python

 Copy code

```
with open("example.txt", "w") as file:  
    file.write("Hello, World!\n")  
    file.write("This is a new line.")
```

python


 Copy code

```
lines = ["Line 1\n", "Line 2\n", "Line 3\n"]  
with open("example.txt", "w") as file:  
    file.writelines(lines)
```


Appending Data to Files

- Syntax: **file = open("filename", "a")**
- Opens the file for writing at the end (appending)

python

 Copy code

```
with open("example.txt", "a") as file:  
    file.write("This is appended content.")
```

Pop Quiz

Q. Which file handling mode in Python is used to open a file for reading?

- a. 'r' - Read Mode
- b. 'w' - Write Mode
- c. 'a' - Append Mode
- d. 'x' - Exclusive Creation Mode



Pop Quiz

Q. Which file handling mode in Python is used to open a file for reading?

- a. 'r' - Read Mode**
- b. 'w' - Write Mode
- c. 'a' - Append Mode
- d. 'x' - Exclusive Creation Mode






Lambda Functions

Introduction to Lambda Functions

- Lambda functions, also known as anonymous functions, are short, one-line functions.
- They are defined using the lambda keyword.
- Syntax: **lambda arguments: expression**

python

 Copy code

```
# Regular Function
```

```
def add(a, b):
```

```
    return a + b
```

```
# Lambda Function
```

```
sum = lambda a, b: a + b
```

Anonymous vs. Named Functions

Named Functions

They are defined using the keyword followed by a function name, parameter list, and a block of code.

They have a name, which is used to call the function.

They can contain multiple lines of code and statements.

They are used for complex operations and when the function needs to be reused in different parts of the code.

Anonymous Functions (Lambda Functions)

They are defined using the keyword followed by a parameter list and a single expression.

They have no name and are known as "anonymous" or "lambda" functions.


Lambda functions are concise and are typically used for simple, one-line operations.

Lambda functions are commonly used with higher-order functions like map, filter and reduce.

Using Lambda Functions as Function Arguments

- In Python, functions that accept other functions as arguments are known as higher-order functions.
- Lambda functions are commonly used as anonymous functions in higher-order functions.

python

 Copy code

```
# List of numbers
numbers = [1, 2, 3, 4, 5]

# Using map() with a lambda function to square each number
squared = list(map(lambda x: x ** 2, numbers))
```

Poll Time

Q. Which of the following statements about Lambda functions in Python is correct?

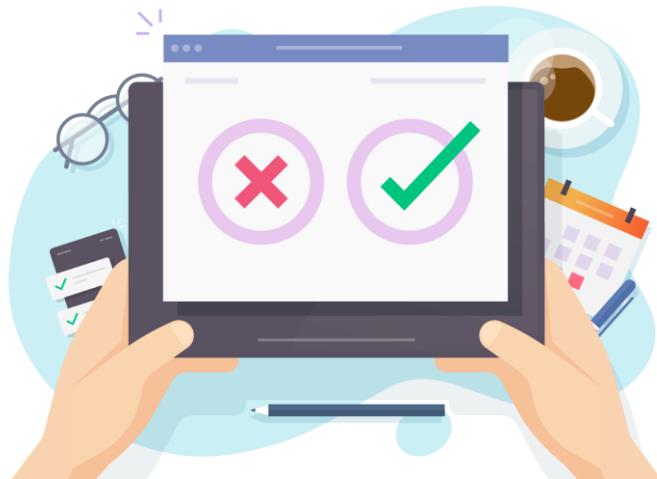
- a. Lambda functions are defined using the def keyword
- b. Lambda functions are named functions with a specific function name
- c. Lambda functions are used for complex operations and multiple lines of code
- d. Lambda functions are anonymous functions defined using the lambda keyword



Poll Time

Q. Which of the following statements about Lambda functions in Python is correct?


- a. Lambda functions are defined using the def keyword
- b. Lambda functions are named functions with a specific function name
- c. Lambda functions are used for complex operations and multiple lines of code
- d. Lambda functions are anonymous functions defined using the lambda keyword**



Filtering Data with Lambda and Filter()

- The filter() function is a higher-order function in Python.
- It filters elements from an iterable based on a given function's output.

python

 Copy code

```
# Regular function to check if a number is even
def is_even(num):
    return num % 2 == 0


# List of numbers
numbers = [1, 2, 3, 4, 5]

# Using filter() with the is_even function
even_numbers = list(filter(is_even, numbers))
```

Mapping Data with Lambda and Map()

- The map() function is a higher-order function in Python.
- It applies a given function to each item of an iterable and returns an iterator.

python

 Copy code

```
# Regular function to square a number
def square(num):
    return num ** 2


# List of numbers
numbers = [1, 2, 3, 4, 5]

# Using map() with the square function
squared_numbers = list(map(square, numbers))
```

Reducing Data with Lambda and Reduce()

- The reduce() function is part of the functools module in Python.
- It performs a cumulative computation on an iterable.

python

 Copy code

```
from functools import reduce

# Regular function to calculate the product of two numbers
def multiply(a, b):
    return a * b

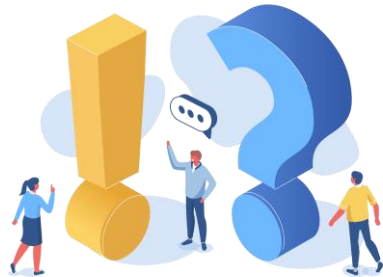
# List of numbers
numbers = [1, 2, 3, 4, 5]

# Using reduce() with the multiply function
product = reduce(multiply, numbers)
```

Pop Quiz

Q. Which of the following statements about using Lambda functions with the map() function in Python is correct?

- a. Lambda functions cannot be used with the map() function
- b. Lambda functions can only be used with the map() function for single-line operations
- c. The map() function applies a lambda function to each item of an iterable
- d. Lambda functions with map() require a specific function name



Pop Quiz

Q. Which of the following statements about using Lambda functions with the map() function in Python is correct?

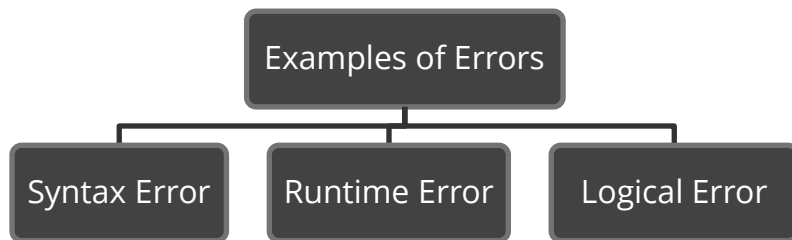
- a. Lambda functions cannot be used with the map() function
- b. Lambda functions can only be used with the map() function for single-line operations
- c. The map() function applies a lambda function to each item of an iterable**
- d. Lambda functions with map() require a specific function name





Introduction to Error Handling in Python

- Error handling is the process of dealing with unexpected errors or exceptions that may occur during program execution.
- It allows your program to gracefully handle errors and prevent unexpected crashes.




Error and Exception Handling

Using Try-except Blocks for Error Handling

- The try-except block is used for error handling in Python.
- It allows you to catch and handle exceptions gracefully.

python

 Copy code

```
try:
    # Code that might raise an exception
except SomeException:
    # Code to handle the exception
```

Poll Time

Q. What is the purpose of error handling in Python?

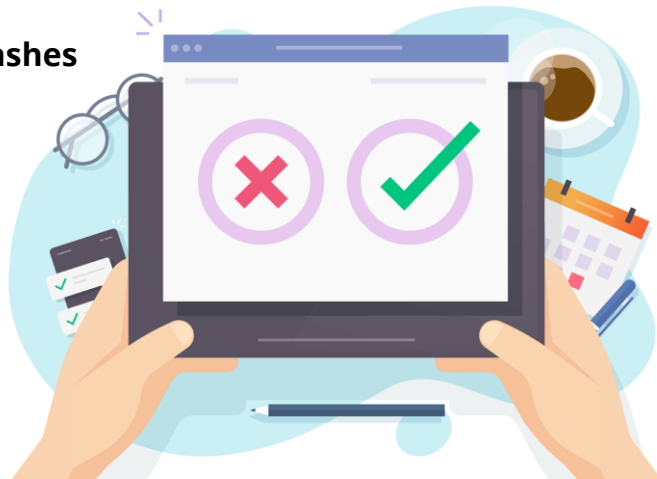
- a. To hide all errors and prevent them from occurring
- b. To allow the program to crash when errors occur
- c. To gracefully handle unexpected errors and prevent program crashes
- d. To introduce intentional errors in the code



Poll Time

Q. What is the purpose of error handling in Python?


- a. To hide all errors and prevent them from occurring
- b. To allow the program to crash when errors occur
- c. To gracefully handle unexpected errors and prevent program crashes**
- d. To introduce intentional errors in the code



Using Multiple Except Blocks

Multiple except blocks in a single try statement is used when you want to handle different types of exceptions separately and provide specific error-handling actions based on the type of exception that occurs during program execution.

python


 Copy code

```
try:
    result = 10 / 0 # Raises a ZeroDivisionError
    # Code that might raise other exceptions
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid value!")
except Exception as e:
    print("An unexpected error occurred:", str(e))
```

Handling Multiple Exceptions in a Single Except Block

This functionality allows you to handle different exceptions with the same set of statements and perform common error-handling actions for multiple exception types.

python

 Copy code

```
try:
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    result = num1 / num2
except (ValueError, ZeroDivisionError):
    print("Invalid input or division by zero. Please enter valid numbers.")
```

Pop Quiz

Q. Which block of code is used to handle exceptions in Python?

- a. **catch** block
- b. **try** block
- c. **except** block
- d. **finally** block



Pop Quiz

Q. Which block of code is used to handle exceptions in Python?

- a. **catch** block
- b. **try** block
- ☒ c. **except block**
- d. **finally** block





Summary

- ✓ File handling can be used to read, write and append data to text files.
- ✓ Lambda functions are used to create anonymous functions in Python.
- ✓ Map, Filter, and Reduce are higher order functions that can be used with Lambda functions.
- ✓ TRY EXCEPT statements are used to gracefully handle Python errors.

Activity 1

Pre-requisites:

- Python 3.x – preferably Python 3.8
- Jupyter Notebook

Scenario:

Continue practicing the basics of Python. Perform the below operations:

- Use Python file handling to create a text file and store your personal details like name, contact, email and address.
- Create a Python list to store first 100 natural numbers. Perform the below operations on the same.
 1. Square each element in the list using `map()`.
 2. Filtering even numbers from the list using `filter()`.
 3. Calculating the product of elements in the list using `reduce()`.

Session Feedback



Next Session:

Deep Dive into Python

THANK YOU!

Please complete your assessments and review the self-learning content for this session on the **PRISM** portal.

