

**Manual for Lab Component
of
Introduction to Python Programming
(BPLCK105B/205B)
1st / 2nd Sem
As per VTU Scheme: 2022 to 2023**

Table of Contents

Sl.NO	Contents	Page. No
1	Syllabus	3
2	Program 1(A): Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages.	5
3	Program 1(B): Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.	7
4	Program 2(A): Develop a program to generate Fibonacci sequence of length (N). Read N from the console.	8
5	Program 2(B): Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N and R).	9
6	Program 3 : Read N numbers from the console and create a list. Develop a program to print mean, variance and standard deviation with suitable messages.	10
7	Program 4: Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.	12
8	Program 5: Develop a program to print 10 most frequently appearing words in a text file. [Hint: Use dictionary with distinct words and their frequency of occurrences. Sort the dictionary in the reverse order of frequency and display dictionary slice of first 10 items]	13
9	Program 6: Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Hint: Use string methods strip(), len(), list methods sort(), append(), and file methods open(), readlines(), and write()].	16
10	Program 7 : Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.	18
11	Program 8: Write a function named DivExp which takes TWO parameters a, b and returns a value c ($c=a/b$). Write suitable assertion for $a>0$ in function DivExp and raise an exception for when $b=0$. Develop a suitable program which reads two values from the console and calls a function DivExp.	20
12	Program 9: Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers. Define a suitable class 'Complex' to represent the complex number. Develop a program to read N ($N \geq 2$) complex numbers and to compute the addition of N complex numbers.	22
13	Program 10 : Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details. [Hint: Use list to store the marks in three subjects and total marks. Use <code>__init__()</code> method to initialize name, USN and the lists to store marks and total, Use <code>getMarks()</code> method to read marks into the list, and <code>display()</code> method to display the score card details.]	24
14	Sample Viva Questions	26

Introduction to Python Programming- Lab Component Syllabus			
Course Code	BPLCK105B/205B	CIE Marks	20
Teaching Hours/Weeks (L: T: P: S)	0: 0: 2: 0		
Course objectives			
<ul style="list-style-type: none"> Learn the syntax and semantics of the Python programming language. Illustrate the process of structuring the data using lists, tuples Appraise the need for working with various documents like Excel, PDF, Word, and Others. Demonstrate the use of built-in functions to navigate the file system. Implement the Object Oriented Programming concepts in Python. 			
Note: two hours tutorial is suggested for each laboratory sessions.			
Prerequisite			
<ul style="list-style-type: none"> Students should be familiarized about Python installation and setting Python environment Usage of IDLE or IDE like PyCharm should be introduced Python Installation: https://www.youtube.com/watch?v=Kn1HF3oD19c PyCharm Installation: https://www.youtube.com/watch?v=SZUNUB6nz3g 			
Sl. No.			
1	a. Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages. b. Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.		
2	a. Develop a program to generate Fibonacci sequence of length (N). Read N from the console. b. Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N and R).		
3	Read N numbers from the console and create a list. Develop a program to print mean, variance and standard deviation with suitable messages.		
4	Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.		
5	Develop a program to print 10 most frequently appearing words in a text file. [Hint: Use dictionary with distinct words and their frequency of occurrences. Sort the dictionary in the reverse order of frequency and display dictionary slice of first 10 items]		
6	Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Hint: Use string methods strip(), len(), list methods sort(), append(), and file methods open(), readlines(), and write()].		
7	Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.		
8	Write a function named DivExp which takes TWO parameters a, b and returns a value c ($c=a/b$). Write suitable assertion for $a>0$ in function DivExp and raise an exception for when $b=0$. Develop a suitable program which reads two values from the console and calls a function DivExp.		
9	Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers. Define a suitable class 'Complex' to represent the complex number. Develop a program to read N ($N \geq 2$) complex numbers and to compute the addition of N complex numbers.		
10	Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details. [Hint: Use list to store the marks in three subjects and total marks. Use <code>__init__()</code> method to initialize name, USN and the lists to store marks and total, Use <code>getMarks()</code> method to read marks into the list, and <code>display()</code> method to display the score card details.]		

Course outcome (Course Skill Set)

1. At the end of the course the student will be able to:
2. Demonstrate proficiency in handling loops and creation of functions.
3. Identify the methods to create and manipulate lists, tuples, and dictionaries.
4. Develop programs for string processing and file organization Interpret the concepts of Object-Oriented Programming as used in Python.

Assessment Details (both CIE and SEE)

CIE for the practical component of the IC

- On completion of every experiment/program in the laboratory, the students shall be evaluated, and marks shall be awarded on the same day. The 15 marks are for conducting the experiment and preparation of the laboratory record, the other 05 marks shall be for the test conducted at the end of the semester.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (duration 03 hours) at the end of the 15th week of the semester /after completion of all the experiments (whichever is early) shall be conducted for 50 marks and scaled down to 05 marks.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IC/IPCC for 20 marks.
- The minimum marks to be secured in 08 (40% of maximum marks) in the practical component. The laboratory component of the IC/IPCC shall be for CIE only.

However, in SEE, the questions from the laboratory component shall be included. The maximum of 05 questions is to be set from the practical component of IC/IPCC, the total marks of all questions should not be more than 25 marks.

Text Books

1. Al Sweigart, “Automate the Boring Stuff with Python”, 1st Edition, No Starch Press, 2015. (Available under CC-BY-NC-SA license at <https://automatetheboringstuff.com/>) (Chapters 1 to 18, except 12) for lambda functions use this link: <https://www.learnbyexample.org/python-lambda-function/>
2. Allen B. Downey, “Think Python: How to Think Like a Computer Scientist”, 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at <http://greenteapress.com/thinkpython2/thinkpython2.pdf> (Chapters 13, 15, 16, 17, 18) (Download pdf/html files from the above link)

Web links and Video Lectures (e-Resources):

- <https://www.learnbyexample.org/python/>
- <https://www.learnpython.org/>
- <https://pythontutor.com/visualize.html#mode=edit>

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

- Quizzes for list, tuple, string dictionary slicing operations using below link https://github.com/sushantkhara/Data-Structures-And-Algorithms-with-Python/raw/main/Python%20%20_%20400%20exercises%20and%20solutions%20for%20beginners.pdf

Laboratory Program 1A: Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages.

Source Code:

```
def calculate_percentage(total_marks):
    return (total_marks / (3*max_marks)) * 100

def display_results(name, usn, marks):
    total_marks = sum(marks)
    percentage = calculate_percentage(total_marks)

    print("\nStudent Details:")
    print("Name:", name)
    print("USN:", usn)

    print("\nMarks:")
    print("Subject 1:", marks[0])
    print("Subject 2:", marks[1])
    print("Subject 3:", marks[2])

    print("\nTotal Marks:", total_marks)
    print("Percentage:", percentage, "%")

# Getting input from the user
name = input("Enter student name: ")
usn = input("Enter USN: ")
max_marks = int(input("Enter the max_marks (25/50/100/Any max):"))
marks = []

# Reading marks for three subjects
for i in range(3):
    subject_marks = float(input("Enter marks for subject {0} per
                                {1}: ".format(i+1,max_marks)))
    marks.append(subject_marks)

# Displaying the results
display_results(name, usn, marks)
```

Sample Output:

Enter student name: Palguni
Enter USN: SDM12345
Enter the max_marks (25/50/100/Any max):25
Enter marks for subject 1 per 25: 21
Enter marks for subject 2 per 25: 22
Enter marks for subject 3 per 25: 23

Student Details:
Name: Palguni
USN: SDM12345

Marks:
Subject 1: 21.0
Subject 2: 22.0
Subject 3: 23.0

Total Marks: 66.0
Percentage: 88.0 %

Laboratory Program 1B: Develop a program to read the name and year of birth of a person. Display whether the person is a senior citizen or not.

Source Code:

```
def is_senior_citizen(year_of_birth):
    current_year = 2023
    age = current_year - year_of_birth
    return age >= 60

# Getting input from the user
name = input("Enter your name: ")
year_of_birth = int(input("Enter your year of birth: "))

# Checking if the person is a senior citizen
if is_senior_citizen(year_of_birth):
    print(name, "is a senior citizen.")
else:
    print(name, "is not a senior citizen.")
```

Sample Output:

```
Enter your name: Thyagaraju
Enter your year of birth: 1975
Thyagaraju is not a senior citizen.
```

Laboratory Program 2A: Develop a program to generate Fibonacci sequence of length (N). Read N from the console.

Source Code:

```
# Program to display the Fibonacci sequence up to N
N = int(input("Enter the length of the sequence "))
# first two terms
t1, t2 = 0, 1
count = 1
# check if the number of terms is valid
if N <= 0:
    print("Please enter a positive integer")
# if there is only one term, return t1
elif N == 1:
    print("Fibonacci sequence upto ",N,": ",end=' ')
    print(t1)
# generate fibonacci sequence
else:
    print("Fibonacci sequence: ")
    while count <= N:
        print(t1,end=' ')
        nextterm = t1 + t2
        # update values
        t1 = t2
        t2 = nextterm
        count += 1
```

Sample Output:

```
Enter the length of the sequence 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
```


Laboratory Program 2B: Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N and R).

Source Code:

```
# definition
def fact(N):
    if N == 0:
        return 1
    res = 1
    for i in range(2, N+1):
        res = res * i
    return res

def NCR(N, R):
    return (fact(N)/(fact(R)*fact(N - R)))

print("Please note N must be greater than R")
N=int(input("Enter the value of N: "))
R=int(input("Enter the value of R: "))
print("The Binomial coefficient of {0}C{1} = {2} ".format(N,R,NCR(N,R)))
```

Sample Output:

```
Please note N must be greater than R
Enter the value of N: 7
Enter the value of R: 3
The Binomial coefficient of 7C3 = 35.0
```

Laboratory Program 3: Read N numbers from the console and create a list. Develop a program to print mean, variance, and standard deviation with suitable messages.

Source Code:

```
import math
list_num = list()
N = int(input(("Enter the value of N: ")))
print("Enter {0} numbers".format(N))
for i in range(1,N+1):
    num = int(input())
    list_num.append(num)
print("The entered list of numbers is : ",list_num)

# Finding sum of numbers in list
sum = 0
for num in list_num:
    sum = sum + num
print("The sum of numbers in the list is : ",sum)

# To find the mean of numbers in list
mean = sum/len(list_num)
print("Mean of numbers in the list is : ",mean)

# To find variance of numbers in list
sum_var = 0
for num in list_num:
    sum_var = sum_var + (num -mean)**2
variance = sum_var/N
print("Variance of numbers in the list is : ",variance)

# To find the standard deviation of numbers in list
std = math.sqrt(variance)
print("The standard deviation of numbers in list is : ",std)
```

Sample Output:

Enter the value of N: 10

Enter 10 numbers

2

3

7

4

6

5

3

8

9

7

The entered list of numbers is : [2, 3, 7, 4, 6, 5, 3, 8, 9, 7]

The sum of numbers in the list is : 54

Mean of numbers in the list is : 5.4

Variance of numbers in the list is : 5.0400000000000001

The standard deviation of numbers in list is : 2.244994432064365

Laboratory Program 4: Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit with suitable message.

Source Code:

```
number=int(input("Enter any Multidigit Number : "))
print("Digit\tFrequency")
for i in range(0,10):
    count=0;
    temp=number;
    while temp>0:
        digit=temp%10
        if digit==i:
            count=count+1
        temp=temp//10;
    if count>0:
        print(i,"\t",count)
```

Sample Output:

```
Enter any Multidigit Number : 223356789
Digit Frequency
2      2
3      2
5      1
6      1
7      1
8      1
9      1
```

Laboratory Program 5: Develop a program to print 10 most frequently appearing words in a text file. [Hint: Use dictionary with distinct words and their frequency of occurrences. Sort the dictionary in the reverse order of frequency and display dictionary slice of first 10 items]

Source Code:

```

from collections import OrderedDict
import numpy as np
import itertools
text = open("spamXY.txt")

def sort_dict_by_value(d, reverse = False):
    return dict(sorted(d.items(), key = lambda x: x[1], reverse = reverse))

# Create an empty dictionary
d = dict()

# Loop through each line of the file
for line in text:
    # Remove the leading spaces and newline character
    line = line.strip()

    # Convert the characters in line to
    # lowercase to avoid case mismatch
    line = line.lower()

    # Split the line into words
    words = line.split(" ")

    # To eliminate delimiters.
    for word in words:
        word = word.replace(".", "")
        word = word.replace(", ", "")
        word = word.replace(":", "")
        word = word.replace(";", "")
        word = word.replace("!", "")
        word = word.replace("*", "")

```

```

# Iterate over each word in line
for word in words:
    # Check if the word is already in dictionary
    if word in d:
        # Increment count of word by 1
        d[word] = d[word] + 1
    else:
        # Add the word to dictionary with count 1
        d[word] = 1
print("\n Sorted dictionary elements by frequency [Descending order]:\n")
d1 =sort_dict_by_value(d, True)
print(d1)
print("\n")

N= int(input("Enter the number of top frequency words to be displayed: \n"))
print("\nThe {0} most frequently appearing words are : \n".format(N))
out = dict(itertools.islice(d1.items(),N))
for i in out:
    print(i)

```

Input file

spamXY - Notepad
File Edit Format View Help

A quantum computer is a computer that exploits quantum mechanical phenomena. At small scales, physical matter exhibits properties of both particles and waves, and quantum computing leverages this behavior using specialized hardware. Classical physics cannot explain the operation of these quantum devices, and a scalable quantum computer could perform some calculations exponentially faster than any modern "classical" computer. In particular, a large-scale quantum computer could break widely-used encryption schemes and aid physicists in performing physical simulations; however, the current state of the art is still largely experimental and impractical.

Sample Output:

Sorted dictionary elements by frequency (Descending order):

```
{'quantum': 6, 'and': 5, 'a': 4, 'computer': 4, 'of': 3, 'the': 3, 'is': 2, 'physical': 2,
'could': 2, 'in': 2, 'that': 1, 'exploits': 1, 'mechanical': 1, 'phenomena.': 1, 'at': 1,
'small': 1, 'scales.': 1, 'matter': 1, 'exhibits': 1, 'properties': 1, 'both': 1, 'particles':
1, 'waves.': 1, 'computing': 1, 'leverages': 1, 'this': 1, 'behavior': 1, 'using': 1,
'specialized': 1, 'hardware.': 1, 'classical': 1, 'physics': 1, 'cannot': 1, 'explain': 1,
'operation': 1, 'these': 1, 'devices.': 1, 'scalable': 1, 'perform': 1, 'some': 1,
```

```
'calculations': 1, 'exponentially': 1, 'faster': 1, 'than': 1, 'any': 1, 'modern': 1,
'"classical"': 1, 'computer.': 1, 'particular.': 1, 'large-scale': 1, 'break': 1, 'widely-
used': 1, 'encryption': 1, 'schemes': 1, 'aid': 1, 'physicists': 1, 'performing': 1,
'simulations;': 1, 'however.': 1, 'current': 1, 'state': 1, 'art': 1, 'still': 1, 'largely': 1,
'experimental': 1, 'impractical.': 1}
```

Enter the number of top frequency words to be displayed:

10

The 10 most frequently appearing words are :

quantum
and
a
computer
of
the
is
physical
could
in

Laboratory Program 6: Develop a program to sort the contents of a text file and write the sorted contents into a separate text file. [Hint: Use string methods strip(), len(), list methods sort(), append(), and file methods open(), readlines(), and write()].

Source Code:

```
infile = open("spamXY.txt", "r")
words = []
for line in infile:
    line = line.strip()
    line = line.lower()
    temp = line.split()
    for word in temp:
        word = word.replace(".", "")
        word = word.replace(", ", "")
        word = word.replace(":", "")
        word = word.replace(";", "")
        word = word.replace("!", "")
        word = word.replace("*", "")
        word = word.replace("'", "")
    for i in temp:
        words.append(i)
infile.close()
words.sort()
print("Sorted words : ")
print(words)
# writing the sorted words into result.txt
outfile = open("result.txt", "w")
for i in words:
    outfile.write(i)
    outfile.write("\n")
outfile.close()
```

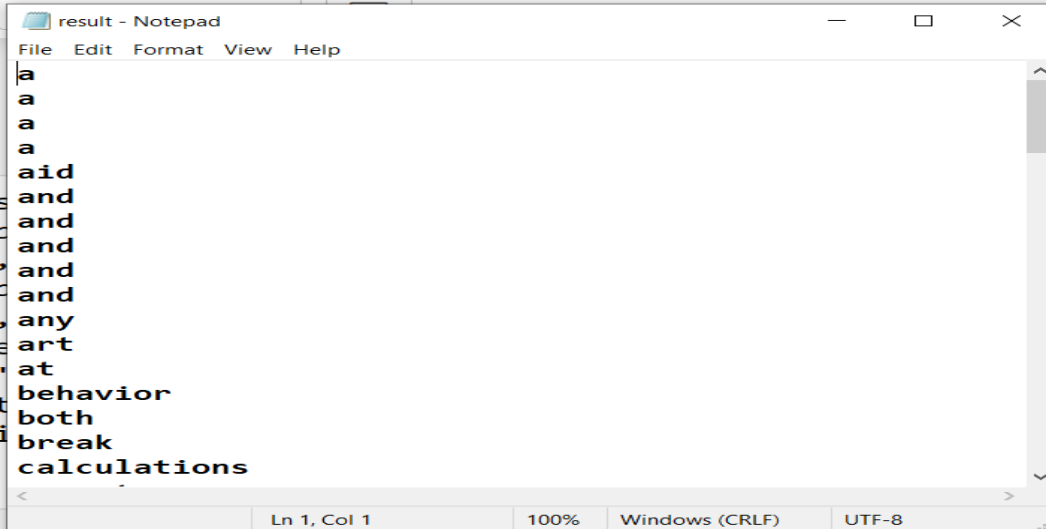
Sample Output:

```
Sorted words :
['a', 'a', 'a', 'a', 'aid', 'and', 'and', 'and',
```



```
'and', 'and', 'any', 'art', 'at', 'behavior', 'both',
'break', 'calculations', 'cannot', 'classical',
'classical', 'computer', 'computer', 'computer',
'computer', 'computer.', 'computing', 'could',
'could', 'current', 'devices,', 'encryption',
'exhibits', 'experimental', 'explain', 'exploits',
'exponentially', 'faster', 'hardware.', 'however,',
'impractical.', 'in', 'in', 'is', 'is', 'large-
scale', 'largely', 'leverages', 'matter',
'mechanical', 'modern', 'of', 'of', 'of',
'operation', 'particles', 'particular,', 'perform',
'performing', 'phenomena.', 'physical', 'physical',
'physicists', 'physics', 'properties', 'quantum',
'quantum', 'quantum', 'quantum', 'quantum',
'quantum', 'scalable', 'scales,', 'schemes',
'simulations;', 'small', 'some', 'specialized',
'state', 'still', 'than', 'that', 'the', 'the',
'the', 'these', 'this', 'using', 'waves,', 'widely-
used']
```

Output file :



```
result - Notepad
File Edit Format View Help
a
a
a
a
aid
and
and
and
and
and
any
art
at
behavior
both
break
calculations
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Laboratory Program 7: Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.

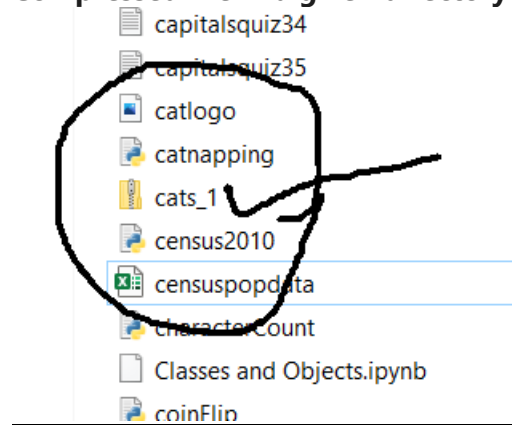
Source Code:

```
import zipfile, os
folder = input("Enter the folder name in the current working directory : ")
folder = os.path.abspath(folder) # make sure folder is absolute
number = 1
while True:
    zipFilename = os.path.basename(folder) + '_' + str(number) + '.zip'
    if not os.path.exists(zipFilename):
        break
    number = number + 1
# Create the zip file.
print('Creating %s...' % (zipFilename))
backupZip = zipfile.ZipFile(zipFilename, 'w')
# Walk the entire folder tree and compress the files in each folder.
for foldername, subfolders, filenames in os.walk(folder):
    print('Adding files in %s...' % (foldername))
    # Add the current folder to the ZIP file.
    backupZip.write(foldername)
    # Add all the files in this folder to the ZIP file.
    for filename in filenames:
        if filename.startswith(os.path.basename(folder) + '_') and
filename.endswith('.zip'):
            continue # don't backup the backup ZIP files
        backupZip.write(os.path.join(foldername, filename))
backupZip.close()
print('Done.')
```

Sample Output:

```
Enter the folder name in the current working directory : cats
Creating cats_1.zip...
Adding files in C:\Users\thyagu\18CS55\cats...
Done.
```

Compressed file in a given directory :



Laboratory Program 8: Write a function named DivExp which takes TWO parameters a, b and returns a value c ($c=a/b$). Write suitable assertion for $a>0$ in function DivExp and raise an exception for when $b=0$. Develop a suitable program which reads two values from the console and calls a function DivExp..

Source Code:

```
def DivExp(a,b):
    assert a>0,"a must be > 0"
    if b == 0:
        raise ZeroDivisionError;
    else: c = a/b
    return c

a = int(input("Enter the value of a :"))
b = int(input("Enter the value of b :"))
r = DivExp(a,b)
print("The value of {0}/{1} = {2}".format(a,b,r))
```

Sample Output 1:

```
Enter the value of a :3
Enter the value of b :2
The value of 3/2 = 1.5
```

Sample Output 2:

```
Enter the value of a :-2
Enter the value of b :3
```

```
-----
AssertionError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5336\2426602043.py in <module>
      8 a = int(input("Enter the value of a :"))
      9 b = int(input("Enter the value of b :"))
--> 10 r = DivExp(a,b)
     11 print("The value of {0}/{1} = {2}".format(a,b,r))

~\AppData\Local\Temp\ipykernel_5336\2426602043.py in DivExp(a, b)
      1 def DivExp(a,b):
----> 2     assert a>0,"a must be > 0"
      3     if b == 0:
      4         raise ZeroDivisionError;
      5     else: c = a/b

AssertionError: a must be > 0
```

Sample Output 3:

Enter the value of a :-2
 Enter the value of b :3

```
-----
AssertionError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5336\2426602043.py in <module>
      8 a = int(input("Enter the value of a :"))
      9 b = int(input("Enter the value of b :"))
----> 10 r = DivExp(a,b)
      11 print("The value of {0}/{1} = {2}".format(a,b,r))

~\AppData\Local\Temp\ipykernel_5336\2426602043.py in DivExp(a, b)
      1 def DivExp(a,b):
----> 2     assert a>0,"a must be > 0"
      3     if b == 0:
      4         raise ZeroDivisionError;
      5     else: c = a/b
```

AssertionError: a must be > 0

Sample Output 4:

Enter the value of a :3
 Enter the value of b :0

```
-----
ZeroDivisionError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5336\2426602043.py in <module>
      8 a = int(input("Enter the value of a :"))
      9 b = int(input("Enter the value of b :"))
----> 10 r = DivExp(a,b)
      11 print("The value of {0}/{1} = {2}".format(a,b,r))

~\AppData\Local\Temp\ipykernel_5336\2426602043.py in DivExp(a, b)
      2     assert a>0,"a must be > 0"
      3     if b == 0:
----> 4         raise ZeroDivisionError;
      5     else: c = a/b
      6     return c
```

ZeroDivisionError:

Laboratory Program 9: Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers. Define a suitable class 'Complex' to represent the complex number. Develop a program to read N ($N \geq 2$) complex numbers and to compute the addition of N complex numbers.

Source Code:

```
# Function to add two complex numbers
def addComp(C1, C2):
    # creating temporary variable
    temp=Complex(0, 0)
    # adding real part of complex numbers
    temp.real = C1.real + C2.real;
    # adding Imaginary part of complex numbers
    temp.imaginary = C1.imaginary + C2.imaginary;
    # returning the sum
    return temp;

# Class to represent a Complex Number
class Complex:
    def __init__(self, tempReal, tempImaginary):
        self.real = tempReal;
        self.imaginary = tempImaginary;

# variable csum for Storing the sum of complex numbers
csum = Complex(0, 0) # inital value of csum set to 0,0
n = int(input("Enter the value of n : "))
for i in range(1,n+1):
    realPart = int(input("Enter the Real Part of complex number{0}:"
    .format(i)))
    imgPart = int(input("Enter the Imaginary Part of complex number {0}:"
    .format(i)))
    c = Complex(realPart,imgPart)
    # calling addComp() method
    csum = addComp(csum,c);
    # printing the sum
    print("Sum of {0} complex numbers is :".format(n))
print(csum.real, "+i*" + str(csum.imaginary))
```

Sample Output:

```
Enter the value of n : 3
Enter the Real Part of complex number 1:2
Enter the Imaginary Part of complex number 1:3
Enter the Real Part of complex number 2:4
Enter the Imaginary Part of complex number 2:5
Enter the Real Part of complex number 3:6
Enter the Imaginary Part of complex number 3:7
Sum of 3 complex numbers is :
12 +i*15
```

Laboratory Program 10: Develop a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details. [Hint: Use list to store the marks in three subjects and total marks. Use `__init__()` method to initialize name, USN and the lists to store marks and total, Use `getMarks()` method to read marks into the list, and `display()` method to display the score card details.]

Source Code:

```
class Student:
    marks = []
    def getData(self, name, USN,max_marks, m1, m2, m3):
        Student.name = name
        Student.USN = USN
        Student.max_marks = max_marks
        Student.marks.append(m1)
        Student.marks.append(m2)
        Student.marks.append(m3)

    def displayData(self):
        print ("Name is: ", Student.name)
        print ("USN is: ", Student.USN)
        #print ("Marks in subject 1: ", Student.marks[0])
        #print ("Marks in subject 2: ", Student.marks[1])
        #print ("Marks in subject 3: ", Student.marks[2])
        print ("Marks are: ", Student.marks)
        print ("Total Marks is: ", self.total())
        print ("Average Marks is: ", self.average())
        print ("Percentage Marks is: ", self.percentage())

    def total(self):
        return (Student.marks[0] + Student.marks[1] +Student.marks[2])

    def average(self):
        return ((Student.marks[0] + Student.marks[1] +Student.marks[2])/3)

    def percentage(self):
        return ((self.average()/Student.max_marks)*100)
```



```
name = input("Enter the name: ")
usn = int (input("Enter the usn: "))
max_marks = int (input("Enter the max_marks (25/50/100/Any max):"))
m1 = int (input("Enter the marks in the first subject out of
{0}:".format(max_marks)))
m2 = int (input("Enter the marks in the second subject out of
{0}:".format(max_marks)))
m3 = int (input("Enter the marks in the third subject out of
{0}:".format(max_marks)))

s1 = Student()
s1.getData(name, usn,max_marks,m1, m2, m3)
s1.displayData()
```

Sample Output:

```
Enter the name: Thyagu
Enter the usn: 12345
Enter the max_marks (25/50/100/Any max):25
Enter the marks in the first subject out of 25:23
Enter the marks in the second subject out of 25:21
Enter the marks in the third subject out of 25:21
Name is: Thyagu
USN is: 12345
Marks are: [23, 21, 21]
Total Marks is: 65
Average Marks is: 21.666666666666668
Percentage Marks is: 86.66666666666667
```

Sample Viva Questions and Answers

Q: How can you enter expressions into the Python interactive shell?

A: You can enter expressions directly into the Python interactive shell by typing them and pressing Enter to see the result.

Q: What are the integer, floating-point, and string data types in Python?

A: The integer data type represents whole numbers, floating-point represents decimal numbers, and string represents a sequence of characters.

Q: How can you concatenate and replicate strings in Python?

A: Strings can be concatenated using the + operator and replicated using the * operator.

Q: What is the purpose of storing values in variables?

A: Storing values in variables allows us to reuse and manipulate those values in our programs.

Q: How do you define and execute your first Python program?

A: To define a Python program, you write your code in a text file with a .py extension. To execute it, you run the program using the python command followed by the filename.

Q: What is the role of flow control in Python?

A: Flow control allows you to dictate the execution path of your program based on certain conditions or criteria.

Q: What are Boolean values in Python?

A: Boolean values represent either true or false. They are often used in flow control and comparison operations.

Q: What are comparison operators in Python?

A: Comparison operators are used to compare values in Python. They include ==, !=, <, >, <=, and >=.

Q: How can you combine Boolean values using Boolean operators?

A: Boolean operators (and, or, and not) allow you to combine multiple Boolean values and perform logical operations.

Q: Can you mix Boolean and comparison operators in Python?

A: Yes, you can mix Boolean and comparison operators in Python to create complex conditions for flow control.

Q: What are the elements of flow control in Python?

A: The elements of flow control include if statements, else statements, elif statements, and loops (such as for loops and while loops).

Q: How is program execution carried out in Python?

A: Python executes programs line by line, from top to bottom, unless flow control statements or loops are encountered.

Q: How can you import modules in Python?

A: You can use the import statement to import modules in Python, making their functions and variables accessible in your program.

Q: How can you end a program early in Python?

A: You can end a program early using the `sys.exit()` function. It terminates the program immediately.

Q: How do you define functions in Python?

A: Functions are defined using the `def` keyword, followed by the function name and parentheses. They can take parameters and perform specific tasks.

Q: What are return values in Python functions?

A: Return values are the results that a function can send back to the caller. They are specified using the `return` statement.

Q: What is the `None` value in Python?

A: The `None` value represents the absence of a value. It is often used to indicate the absence of a return value in a function.

Q: How can you use keyword arguments with the `print()` function?

A: Keyword arguments in the `print()` function allow you to specify additional options such as the separator or the end character.

Q: What is the difference between local and global scope in Python?

A: Local scope refers to variables that are defined within a specific function or block and can only be accessed within that scope. Global scope refers to variables that are defined outside of any function and can be accessed throughout the entire program.

Q: How can you handle exceptions or errors in Python?

A: Python provides exception handling mechanisms to catch and handle errors that may occur during program execution. This can be done using `try-except` blocks, where potential exceptions are caught and appropriate actions are taken.

Q: Can you explain the concept of the global statement in Python?

A: The global statement in Python is used to indicate that a variable inside a function should be treated as a global variable rather than a local variable. This allows you to modify and access global variables within a function.

Q: How can you create a short program like "Guess the Number" in Python?

A: You can create a "Guess the Number" program by generating a random number, accepting user guesses, and providing feedback until the correct number is guessed. This program can utilize flow control, user input, and comparison operations.

Q: What is the purpose of comments in Python code?

A: Comments in Python are used to add explanatory notes within the code. They are ignored by the interpreter and are helpful for documentation and improving code readability.

Q: How can you handle user input in Python?

A: You can handle user input in Python by using the `input()` function. It allows you to prompt the user for input and store the value in a variable.

Q: What is the role of indentation in Python?

A: Indentation is crucial in Python as it defines the structure and hierarchy of the code. Proper indentation is required to group statements within blocks and to indicate flow control and function definitions.

Q: How can you format strings in Python?

A: Strings can be formatted in Python using the `format()` method or by using f-strings (formatted string literals). These allow you to insert variables and expressions within strings in a structured manner.

Q: How can you handle multiple exceptions in Python?

A: Multiple exceptions can be handled in Python using multiple `except` blocks or by catching multiple exceptions within a single `except` block separated by commas.

Q: What is the purpose of a loop in Python?

A: Loops allow you to repeatedly execute a block of code until a certain condition is met. Python provides `for` loops and `while` loops for different looping scenarios.

Q: How can you open and read data from a file in Python?

A: You can open and read data from a file in Python using the `open()` function with the appropriate file mode, such as `'r'` for reading. The `read()` or `readlines()` methods can be used to read the data.

Q: What are the benefits of using functions in Python?

A: Functions promote code reusability, improve code organization, and allow for modular design. They also make the code easier to understand, test, and maintain.

Q: What is the list data type in Python?

A: The list data type in Python is a mutable sequence that can store multiple values. It is represented by square brackets [].

Q: How can you create a list in Python?

A: You can create a list in Python by enclosing comma-separated values inside square brackets [].

Q: How can you access elements in a list?

A: Elements in a list can be accessed using index values. Indexing starts from 0, so the first element is at index 0, the second element is at index 1, and so on.

Q: Can you explain the concept of mutable and immutable data types in Python?

A: Mutable data types, such as lists, can be modified after they are created. Immutable data types, such as strings and tuples, cannot be modified once they are created.

Q: What are augmented assignment operators in Python?

A: Augmented assignment operators, such as +=, -=, *=, /=, allow you to perform an operation and assignment in a single step.

Q: What are some methods available for working with lists?

A: Some commonly used methods for lists include append(), insert(), remove(), sort(), reverse(), count(), and index().

Q: Can you provide an example program using a list, such as a Magic 8 Ball?

A: Yes, a Magic 8 Ball program can utilize a list of possible answers and use random selection to provide a response.

Q: How can you concatenate or combine two lists?

A: You can concatenate two lists using the + operator or the extend() method.

Q: How can you check if an element exists in a list?

A: You can use the in keyword to check if an element exists in a list. It returns True if the element is present and False otherwise.

Q: What are list-like types in Python?

A: List-like types in Python include strings and tuples, which share some similarities with lists, such as indexing and slicing.

Q: How are strings similar to lists?

A: Strings are similar to lists in that they are sequences of characters that can be accessed using index values and sliced.

Q: How are tuples similar to lists?

A: Tuples are similar to lists in that they are ordered sequences, but unlike lists, tuples are immutable.

Q: What are references in Python?

A: In Python, variables are references to objects. Multiple variables can refer to the same object, allowing for efficient memory usage and flexibility.

Q: What is the dictionary data type in Python?

A: The dictionary data type in Python is an unordered collection of key-value pairs. It is represented by curly braces {}.

Q: How can you access values in a dictionary?

A: You can access values in a dictionary by specifying the corresponding key inside square brackets [].

Q: What is the purpose of pretty printing in Python?

A: Pretty printing refers to formatting complex data structures, such as dictionaries or lists, in a more visually appealing and readable way.

Q: How can you use data structures to model real-world things?

A: Data structures, such as lists, dictionaries, and tuples, can be used to represent and organize data that models real-world entities or relationships.

Q: How can you add a key-value pair to a dictionary?

A: You can add a key-value pair to a dictionary by assigning a value to a new key or by using the update() method.

Q: How can you remove a key-value pair from a dictionary?

A: You can remove a key-value pair from a dictionary using the del keyword or the pop() method.

Q: Can you provide an example of using dictionaries to model real-world data?

A: Sure! For example, you can use a dictionary to represent a student record, where the keys could be "name", "age", and "grade", and the values could be the corresponding information for each key.

Q: What are some advantages of using dictionaries?

A: Dictionaries allow for fast lookup of values based on keys, provide a convenient way to represent structured data, and support flexible data manipulation and retrieval.

Q: How can you iterate over the elements of a list or dictionary?

A: You can use a loop, such as a for loop, to iterate over the elements of a list or dictionary and perform operations on each element.

Q: How can you sort a list in Python?

A: You can use the `sort()` method to sort a list in ascending order. Additionally, you can use the `sorted()` function to obtain a sorted version of the list.

Q: What is the difference between a shallow copy and a deep copy of a list?

A: A shallow copy of a list creates a new list object but references the same elements, while a deep copy creates a completely new list with its own separate copies of the elements.

Q: How can you convert a list to a string in Python?

A: You can use the `join()` method to concatenate the elements of a list into a single string.

Q: How can you find the length of a list or dictionary in Python?

A: You can use the `len()` function to find the number of elements in a list or the number of key-value pairs in a dictionary.

Q: How can you check if two lists are equal in Python?

A: You can use the `==` operator to check if two lists have the same elements in the same order.

Q: Can you explain the concept of nested lists in Python?

A: Nested lists are lists that contain other lists as elements. They allow for the creation of more complex data structures and multi-dimensional arrays.

Q: How can you update or modify elements in a dictionary?

A: You can update or modify elements in a dictionary by assigning a new value to the corresponding key.

Q: How can you manipulate strings in Python?

A: Strings in Python can be manipulated using various built-in string methods such as `lower()`, `upper()`, `strip()`, `split()`, `replace()`, and `join()`.

Q: What are some useful string methods in Python?

A: Some useful string methods in Python include `startswith()`, `endswith()`, `find()`, `count()`, `isalpha()`, `isdigit()`, and `format()`.

Q: Can you explain the project "Password Locker" in Python?

A: The "Password Locker" project involves creating a program that stores and retrieves passwords for different accounts, providing secure storage and easy access.

Q: Can you explain the project "Adding Bullets to Wiki Markup" in Python?

A: The "Adding Bullets to Wiki Markup" project involves creating a program that takes text input and adds bullet points to each line, allowing for easy formatting in Wiki markup.

Q: How can you read and write files in Python?

A: In Python, you can use the `open()` function to open a file for reading or writing. The `read()` and `write()` methods are then used to perform read and write operations.

Q: What is a file path in Python?

A: A file path is the location or address of a file in a file system. It specifies the directory or folder hierarchy leading to the file.

Q: What is the purpose of the `os.path` module in Python?

A: The `os.path` module in Python provides functions for working with file paths, such as joining paths, splitting paths, checking file existence, and extracting file extensions.

Q: How can you read the contents of a file in Python?

A: You can read the contents of a file in Python by using the `read()` or `readlines()` method after opening the file in read mode.

Q: How can you write data to a file in Python?

A: You can write data to a file in Python by using the `write()` or `writelines()` method after opening the file in write mode.

Q: What is the purpose of the `shelve` module in Python?

A: The `shelve` module in Python allows you to store and retrieve Python objects in a file. It provides a simple way to save and load variables.

Q: How can you save variables using the `shelve` module in Python?

A: To save variables using the `shelve` module, you can open a shelve file, assign

values to the keys in the file, and then close the file to save the changes.

Q: How can you format variables using the `print.format()` function?

A: The `print.format()` function allows you to format strings by specifying placeholders and values to be substituted. It provides flexible string formatting options.

Q: Can you explain the project "Generating Random Quiz Files" in Python?

A: The "Generating Random Quiz Files" project involves creating a program that generates random quizzes, with questions and answer choices, and saves them to a file.

Q: Can you explain the project "Multiclipboard" in Python?

A: The "Multiclipboard" project involves creating a program that allows you to store multiple pieces of text to a clipboard and retrieve them when needed.

Q: How can you append data to an existing file in Python?

A: To append data to an existing file in Python, you can open the file in append mode by using the 'a' parameter with the `open()` function. Then, you can use the `write()` method to add new content to the file without overwriting the existing content.

Q: What are some common file modes used in Python?

A: Some common file modes used in Python include:

'r': Read mode

'w': Write mode

'a': Append mode

'x': Exclusive creation mode (for writing new files)

't': Text mode (default)

'b': Binary mode

Q: How can you handle exceptions when working with files?

A: When working with files, you can use try-except blocks to handle exceptions. For example, you can use a try-except block to handle `FileNotFoundError` when opening a file.

Q: What is the purpose of the with statement when working with files?

A: The with statement in Python is used to ensure that resources, such as file handles, are properly managed and released. It automatically handles the opening and closing of files, even if exceptions occur.

Q: Can you explain the concept of file encoding in Python?

A: File encoding refers to the way in which characters and symbols are stored in

a file. Common file encodings include UTF-8, ASCII, and UTF-16. It is important to specify the correct encoding when reading or writing files to ensure proper handling of characters.

Q: What is the purpose of the `shutil` module in Python?

A: The `shutil` module provides a high-level interface for working with files and directories. It offers functions for copying, moving, renaming, and deleting files, as well as for creating and removing directories.

Q: How can you copy files and directories using the `shutil` module?

A: You can use the `shutil.copy()` function to copy individual files, and the `shutil.copytree()` function to copy directories and their contents.

Q: How can you walk through a directory tree in Python?

A: You can use the `os.walk()` function to iterate through a directory tree, returning the path, directories, and files at each level.

Q: What is the purpose of the `zipfile` module in Python?

A: The `zipfile` module provides tools for creating, reading, and extracting files from ZIP archives.

Q: How can you compress files into a ZIP archive using the `zipfile` module?

A: You can create a ZIP archive and add files to it using the `zipfile.ZipFile()` class and its `write()` method.

Q: Can you explain the project "Renaming Files with American-Style Dates to European-Style Dates"?

A: The "Renaming Files with American-Style Dates to European-Style Dates" project involves creating a program that renames files with dates in the format "MM-DD-YYYY" to the format "DD-MM-YYYY" for consistency.

Q: Can you explain the project "Backing Up a Folder into a ZIP File"?

A: The "Backing Up a Folder into a ZIP File" project involves creating a program that recursively compresses an entire folder and its contents into a ZIP file for easy backup and storage.

Q: What is the purpose of raising exceptions in Python?

A: Raising exceptions allows you to signal that an error or exceptional condition has occurred in your program, allowing you to handle it appropriately.

Q: How can you get the traceback as a string in Python?

A: You can use the `traceback.format_exc()` function to obtain the traceback information as a formatted string, which can be helpful for debugging and error

logging.

Q: What are assertions in Python and how are they used for debugging?

A: Assertions are statements that check whether a given condition is true and raise an exception if it's false. They are used as debugging aids to identify and verify assumptions in code.

Q: What is logging in Python and how is it useful for debugging?

A: Logging is a module in Python that provides a flexible and powerful way to record messages during program execution. It allows you to track the flow of your program and debug issues by outputting informative messages to different output streams.

Q: Can you explain how to use IDLE's debugger in Python?

A: IDLE's debugger is a built-in feature of the IDLE development environment. It allows you to step through your code, set breakpoints, and inspect variables to debug and troubleshoot issues in your program.

Q: How can you delete files and directories using the shutil module?

A: You can use the `shutil.rmtree()` function to delete directories and their contents, and the `os.remove()` function to delete individual files.

Q: How can you handle file and directory permissions using the shutil module?

A: The `shutil` module provides functions like `shutil.chmod()` to modify file permissions. You can use this function to set the desired permissions on files or directories, such as read, write, and execute permissions.

Q: What is the purpose of the `os.path` module in Python?

A: The `os.path` module in Python provides functions for working with file paths. It offers methods for manipulating path strings, such as joining paths, splitting paths into directory and file components, and checking for path validity.

Q: How can you determine if a file or directory exists using the `os.path` module?

A: You can use the `os.path.exists()` function to check if a file or directory exists. It returns `True` if the specified path exists, and `False` otherwise.

Q: Can you explain the process of reading and writing files in Python?

A: To read a file in Python, you can open it in read mode using the `open()` function and then use methods like `read()`, `readline()`, or `readlines()` to access the file's contents. To write to a file, you open it in write mode and use the `write()` method or related functions.

Q: What is the purpose of the `shelve` module in Python?

A: The `shelve` module provides a simple way to store and retrieve Python objects in a file. It allows you to save and load variables, dictionaries, and other objects using key-value pairs.

Q: How can you save variables using the `shelve` module in Python?

A: To save variables using the `shelve` module, you can open a `shelve` file using `shelve.open()`, assign values to the keys in the file as if it were a dictionary, and then close the file to save the changes.

Q: Can you explain the project "Generating Random Quiz Files" in Python?

A: The "Generating Random Quiz Files" project involves creating a program that generates random quizzes by randomly selecting questions from a pool of available questions. The program then saves the generated quiz questions and answer options to a file for distribution.

Q: What are programmer-defined types in Python?

A: Programmer-defined types are custom data types created by the programmer using classes. They allow you to define your own data structures and behaviors.

Q: What are attributes in Python classes?

A: Attributes are variables that belong to an object. They store data specific to that object and can be accessed using dot notation.

Q: How can you create instances of a class in Python?

A: Instances of a class can be created by calling the class name as if it were a function, which invokes the class's constructor.

Q: Can you explain the concept of objects being mutable in Python?

A: In Python, objects of certain types, such as lists and dictionaries, are mutable, meaning their values can be changed after they are created. This allows for modifying the object's attributes or contents.

Q: How can you copy an object in Python?

A: You can create a copy of an object by using the `copy()` method or the `copy.deepcopy()` function. The former creates a shallow copy, while the latter creates a deep copy.

Q: What is the purpose of the `__init__` method in a class?

A: The `__init__` method is a special method in Python classes that is automatically called when a new instance of the class is created. It is used to initialize the object's attributes.

Q: What is the role of the `__str__` method in a class?

A: The `__str__` method is a special method that returns a string representation of an object. It is invoked when the `str()` function or the `print()` function is called on an object.

Q: How can you overload operators in a Python class?

A: Operator overloading allows you to define the behavior of operators, such as `+`, `-`, `*`, and `/`, for objects of a class. This is achieved by implementing special methods, such as `__add__`, `__sub__`, `__mul__`, and `__div__`, within the class.

Q: What is polymorphism in object-oriented programming?

A: Polymorphism refers to the ability of objects of different classes to be used interchangeably in code. It allows for code reuse and flexibility in implementing different behaviors for different objects.

Q: What is the difference between interface and implementation in object-oriented programming?

A: The interface of a class defines the methods that can be called on objects of that class. It describes what an object can do. The implementation, on the other hand, includes the actual code that defines how the methods work.

Q: What is a pure function in Python?

A: A pure function is a function that always returns the same output for the same input and has no side effects. It does not modify any external state and does not depend on any external state.

Q: How can you define a class in Python?

A: You can define a class in Python by using the `class` keyword, followed by the class name and a colon. Inside the class, you can define attributes and methods.

Q: How can you access the attributes of an object in Python?

A: You can access the attributes of an object using dot notation. For example, if `obj` is an object and `attr` is an attribute, you can access it as `obj.attr`.

Q: What is the purpose of the `self` parameter in Python class methods?

A: The `self` parameter is a reference to the current instance of the class. It allows methods to access and modify the object's attributes and call other methods within the class.

Q: How can you implement method overloading in Python?

A: Unlike some other programming languages, Python does not support method overloading in the traditional sense, where multiple methods with the same name but different parameter lists are defined. However, you can achieve similar

functionality by using default parameter values or variable-length argument lists.

Q: What is the difference between class attributes and instance attributes in Python?

A: Class attributes are shared among all instances of a class and are defined outside of any methods. Instance attributes, on the other hand, are specific to each instance of a class and are typically defined within the `__init__` method or other instance methods.

Q: How can you access class attributes and instance attributes in Python?

A: Class attributes can be accessed using either the class name or an instance of the class, while instance attributes can only be accessed using an instance of the class. Class attributes are shared among all instances, while each instance has its own set of instance attributes.

Q: What is the purpose of the `super()` function in Python classes?

A: The `super()` function is used to call a method from the superclass (the class that the current class inherits from). It allows you to invoke the superclass's methods and override them if necessary. By using `super()`, you can ensure that all the necessary initialization and behavior from the superclass is inherited and executed.

Q: What is type-based dispatch in Python?

A: Type-based dispatch, also known as method overloading, is a programming technique that allows different implementations of a method to be called based on the type of the arguments passed to it. It provides flexibility in handling different types of objects with specific methods.

Q: How can you print objects in Python using the `__str__` method?

A: By defining the `__str__` method in a class, you can specify how the object should be represented as a string when the `str()` function or the `print()` function is called on the object. This allows for customized string representations of objects.

Q: Can you provide an example of a more complicated class in Python?

A: Sure! Let's consider a class called `Car` that represents a car object. It could have attributes like `make`, `model`, `year`, and methods like `start_engine()`, `accelerate()`, and `brake()`.

Q: What is the purpose of the `super()` function in Python classes?

A: The `super()` function is used to call a method from the superclass (the class that the current class inherits from). It allows you to invoke the superclass's methods and override them if necessary.

Q: How can you implement inheritance in Python classes?

A: Inheritance in Python is implemented by creating a new class that derives from an existing class. The new class inherits the attributes and methods of the existing class and can add its own attributes and methods or override the inherited ones.

Q: What is encapsulation in object-oriented programming?

A: Encapsulation is the concept of bundling data and the methods that operate on that data within a class. It allows for data hiding and abstraction, ensuring that the internal representation and functionality of an object are hidden from external code.

Q: How can you handle exceptions in Python?

A: Exceptions in Python can be handled using the try and except statements. The code that may raise an exception is enclosed within the try block, and the exception handling code is written in the except block.

Q: What is the purpose of the finally block in exception handling?

A: The finally block is used to define a set of statements that will be executed regardless of whether an exception was raised or not. It is typically used to clean up resources or perform necessary actions before exiting the exception handling block.

Q: Can you explain the concept of method overriding in Python?

A: Method overriding occurs when a subclass defines a method with the same name as a method in its superclass. The subclass's method overrides the superclass's method, allowing for customized behavior in the subclass.

Q: How can you implement method overriding in Python?

A: Method overriding in Python is achieved by redefining a method in the subclass with the same name as the method in the superclass. This allows the subclass to provide its own implementation of the method.

Q: What is the purpose of the pass statement in Python?

A: The pass statement is a placeholder statement in Python that does nothing. It is often used as a placeholder for code that will be implemented later or as a placeholder in empty code blocks.

Q: How can you create a new instance of a class in Python?

A: To create a new instance of a class, you can call the class name followed by parentheses. This invokes the class's constructor and returns a new instance of the class.

Q: How can you implement method overloading in Python?

A: Method overloading, as seen in some other programming languages, where multiple methods with the same name but different parameter lists are defined, is not directly supported in Python. However, you can achieve similar functionality by using default parameter values or variable-length argument lists.

Q.What is the role of the `__init__` method in a class?

A.The `__init__` method in a class is a special method in Python that is automatically called when an object of the class is created. It stands for "initialize" and is used to initialize the attributes of an object.

The primary role of the `__init__` method is to set up the initial state of an object by defining and assigning values to its attributes. It is commonly used to initialize instance variables based on the arguments passed to the class constructor.

XX

Compiled and Edited by
Palguni G T