Infosys®
Navigate your next

COURSE COMPLETION CERTIFICATE

The certificate is awarded to

SUSHANT KUMAR

for successfully completing the course

Python Concurrent Programming: Introduction to Concurrent Programming

on October 19, 2025

Infosys | Springboard

Congratulations! You make us proud!

Satheesha B.N.
Satheesha B. Nanjappa
Senior Vice President and Head
Education, Training and Assessment
Infosys Limited

Issued on: Tuesday, October 21, 2025
To verify, scan the QR code at https://verify.onwingspan.com

Date: 25/7/25

Task 1: Running Python script and various expression in an interactive interpreter

Aim: Run python script in an interactive interpreter

a. Create a program to enter number and operform display result after calculation (simple)

Algorithm:

1. Start
2. Cret two no. in variable x, y
3. addition; x+y, print
4. Multiplication; x*y, print
5. Divison x/y
6. Stop

Programm:

```
x= int(input("Enter no.:"))
y= int(input("Enter no.:"))
add: x+y
sub: x-y
```

Output (a):

First No. : 1

Second No : 2

addition : 3

subtract : -1

product : 2

division : 0.5

Output (b):

First No. : 1

second No. : 2

3rd No. : 3

1 > 2 is false

1 < 2 is True

3 == 2 False

1 != 2 True

1 >= 2 is False

print ("addition", add)

print ("subtraction", sub)

print ("Divison", div)

b. Create a python programm to enter 2 nos and then perform relational expression:

>, <, ==, !=, >=, <=

Algorithm:

1. Start
2. Get the input from user
3. Perform relational expression (>, <, >=, <=)
4. Print the result
5. Stop

Program:

```
a = int(input("Enter a"))
b = int(input("Enter b"))
c = int(input("Enter c"))
#using operators
print(a, "> ", b, "is", a>b)
print(a, "< ", b, "is", a<b)
print(a, "<= ", b, "is", a<=b)
```

output (c):

Enter first no. = 1

Enter second no. = 2

Enter third no. = 3

Logical operation result:

False
False
True
True

c. Program to enter 3 no's and perform, print result of logical operations and, or, not

Algorithm:

1. Start
2. Get input
3. Perform logical operations
4. Print results
5. Stop

Program:

```
a= int (input ("Enter a"))
b= int (input ("Enter b:"))
c= int (input ("Enter c"))
print ("\n logical results:")
print ((a>b) and (b>c))
print (not (a>b))
print (not (b > c))
print ((a>b) or (b>c))
```

Result:

Thus the python program to run python script in an interactive interpreter was done successfully, output verified

VEL TECH - CSE	
EX NO.	1
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	11/8/25

Date: 1/8/25

Task 2: Implement conditional control and looping statements

Aim: To Implement conditional control and looping statements

a. Develop a grade management system

The grading system

if score => 90 grade = A

if score => 80 or <= 89 grade = B

if score => 70 or <= 79 grade C

if score => 60 or <= 69 grade D

if score <= 60 grade : F

Algorithm :

1. Start
2. Input the mark
3. use if - elif - else statement for grading system
4. print result
5. Stop

Output (b):

The first 10 natural are

1
2
3
4
5
6
7
8
9
10

~~Output (c):~~

Program:

```
score = int(input("Enter mark: "))
if score >=90:
    print("Grade A")
elif(score <=89 and score >=80):
    print("B")
elif(score <=79 & score >=70):
    print("C")
elif(score <=69 & score >=60):
    print("D")
else: print("The grade is F")
```

b. Write a program to print first 10 natural nos.

Algorithm:

1. Start
2. Use loop for generating the numbers
3. Print
4. Stop

Program:

```
print("The first 10 natural numbers are:")
for i in range(1,11):
    print(i)
```

Output (s):

Enter the no. = 24

The no. of digit in 24 is = 2

C WAP to count the total no. of digits in a given number

Algorithm:

1. start
2. Get input from user
3. Convert integer to string using str()
4. use len functions to find no. of digits
5. Print output

Program:

```
digit = int(input("Enter the number:"))
string = str(digit)
count = len(string)
print("The number of digits in "digit" is:", cnt)
```

VEL TECH - CSE	
EX NO.	2
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	11/9/24

Result:

Thus, the python program to implement conditional, control and looping statements was done successfully

Output:

Addition : 15

Subtraction : 5

Multiplication : 50

Division : 20

Date : 8/8/25

Task 3 : Importing Python modules and packages in python programming

Aim : To write python demonstrating importing module and package

a. Create a calculator with main program to handle user input, call appropriate module, display result

Algorithm : 1. Define functions for add, sub, mul, div.

2. Handle division by zero by error

3. Import math eg math.<function_name>(a,b)

4. Print result after each function

5. Stop

Output:

Circle area (radius=7): 153.9300

Rectangle (5×10): 50

Triangle area (b=16, h=8): 240

Program:

```
def add(a,b):
    return a+b
def subtract(a,b):
    return a-b
    raise ValueError("Cannot divide by zero")
    return a/b
import math
print("Multiplication:", mymath.multiply(a,b))
print("Div", mymath.divide(a,b))
```

b.

Create a package with 2 subpackages with 2 modules mathfuns and areafns

Algorithm:

1. Create mathfunctions.py module
2. Create areafunctions.py module
3. Create __init__.py file in pack1 and pack2:
4. Create main.py
5. Print the output

Program:

1. Create mathfunctions .by module

```
def add(a,b):
    return a+b
def subtract(a,b):
    return a-b
return "Error! Division by zero"
return a/b
```

2. Create areafunctions .by module

```
import math
def circle_area:
    return mathpi * r * r
def rectangle_area(l,w):
    return l*w
```

3. Create main .by file from pack import mathfunctions from pack import area functions .

VEL TECH - CSE	
EX NO.	8
ERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
OTAL (20)	15
N WITH DATE	14/9/25

Result: Thus, the program for importing python modules & packages was successfully executed and output verified

14/9/25

Output 4.1

[10, 20, 30]

[10, 20]

[30]

[5, 8, 9, 15, 30, 89]

The min value is : 5

The max value is : 89

Sum : 156

Average : 26.0

Date: 22/8/25

Task 4 : Use various data types, List, Tuples, and Dictionary in python programming

Aim : To use various data types, List, Tuples and Dictionary in python programming

Q. WAP to i) Add elements
ii) Remove elements
iii) Sort elements
iv) Find min, max in a list

Algorithm: 1) Start

2) Create a list

3) To remove use pop or remove

4) For sorting use sorted(list)

5) To find min max values min(list)

4.2

output:

(10, 'hello', 3.14)

10

hello

3.14

(3,14)

(10, 'hello', 3.14)

(10, hello, 3.14, 5)

TypeError: 'tuple' object is not callable

Program:

```
list = [10,20]
a = 30
list.append(a)
list.pop(1)
list.remove(10)
l = [5,8,9,15,30,89]
print(sorted(l))
print("The minimum value is:", min(l))
print("The maximum value is:", max(l))
print("The sum is:", sum(l))
```

b. Create a program to showcase operations on tuples. i) Create tuple
ii) Access Elements
iii) Concatenate tuples
iv) Immutable Nature

Algorithm:

1. Start
2.

2. To create a tuple
   "tuple_name = (values)".

3. To concatenate (tuple1 "+" tuple2).

4. Print output

5. End Stop

Program:

```
tuple = (10, 'hello', 3.14, 'world')
print (tuple)
for i in tuple:
    print(i)
print (tuple[1:3])
print (tuple)
print (t3)
tuple[3] = "PI"
```

ERROR

4.3

output:

{'name': 'Alice', 'age': 30, 'city': 'NY'}

Alice

30

{'name': 'James', 'age': 30, 'city': 'NY'}

{'name': 'James', 'age': 30}

KEY: name

KEY: age

dict_items([('name', 'James'), ('age', 30)])

C. Create python program to showcase operation on dictionary

1. Create a Dictionary
2. Access values
3. Modify dictionary
4. Iterate

Algorithm:

1. Start program
2. Define a dictionary with key value pairs
3. Modify Dictionary
4. Iterate over dictionary
5. Stop the program

Program:

```
dictionary = {'name': 'Alice', 'age': 30, 'city': 'New York'}
print(dictionary)
```

```
print (dictionary ['name'])
print (dictionary ['age'])
# Modify dictionary : Updat
print (dictionary)
    for ik in dictionary:
        print ("KEY:", ik)
print (dictionary.items())
```

VEL TECH - CSE	
X NO.	4
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
OTAL (20)	15
N WITH DATE	

Result:

Thus, various data types, List, Tuples and Dictionary in Python programming was used and verified successfully

29/8/25

Task 5 - Implement various Searching and sorting Operations in Python programming

Aim: To implement various searching and sorting operations in python programming

5.1 A company stores employee records in a list of dictionaries where each dictionary contains id, name and department, Create a dictionary to return the employee with matching ID or none.

Algorithm:

1. Input Definition
2. Define function find_emp_by_id
   a. A list of dictionaries, each dictionary represents emp record
   b. An Integer (target_id) representing emp ID
3. Iterate through List:
4. Check for matching ID:

Output:

{'id': 2, 'name': 'Bob', 'department': 'Engineering'}

29101

5. Return Matching Record
6. Handle No Match
   If loop completes no match return None.

Program 5.1

```
def find_emp ( emp , t_id ):
    for i in emp :
        if i['id'] == t_id :
            return i
    return None

emp = [
    {'id' : 1 , 'name' : 'Alice', 'department' : 'AR'},
    {'id' : 2 , 'name' : 'Bob' , 'department' : 'Engi'},
    {'id' : 3 , 'name' : 'Charlie' , 'depdent' : 'sales'},
]
print ( find_emp (emp, 2 ))
```

Output:

Before Sorting

{'name': 'Alice', 'score': 88}
{'name': 'Bob', 'score': 95}
{'name': 'Charlie', 'score': 75}
{'name': 'Diana', 'score': 85}

After sorting

{'name': 'Charlie', 'score': 75}
{'name': 'Diana', 'score': 85}
{'name': 'Alice', 'score': 88}
{'name': 'Bob', 'score': 95}

Regestation

5.2 You are developing a grade management system for a school. The system maintains a list of student records, each record is represented as a dictionary containing name and records, Bubble sort algorithm

Algorithm:

1. Initialization
   - Get len
2. Outer loop:
   - Iterate from i: 0 to n-1
3. Track swaps:
   - Initialize a boolean variable swapped to False.
4. Inner Loop
   - Iterate from j=0 to n-i-2
5. Compare and swap:
   - For each pair of adjacent elements
     - Compare their score values
     - If student[j]['score'] > student[j+1]['score']
     - Set swapped to True to indicate swap made
6. Early Termination:
   - After each pass of the inner loop, check if swapped is false
7. Completion
   - The function modifies the students list in place, sorting it by score

Some Changes

Program : 5.2

```
def bubble_s (students):
    n = len (students)
    for i in range (n):
        swapped = False
        for j in range (0, n-i-1):
            if students [j] ['score'] > students[j+1]
            ['score']:
                students [j], students [j+1] = students [j+1],
                students [j]
                swapped = True
        if not swapped:
            break

students = [
    {'name': 'Alice', 'score': 85},
    {'name': 'Bob', 'score': 95},
    {'name': 'Charlie', 'score': 75},
    {'name': 'Diana', 'score': 85}
]
print ("Before sorting:")
for student in students: print (student)
```

VEL TECH - CSE	
EX NO.	5
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	11/9/25

Result:

Thus, the Program for various searching and sorting operations is executed and verified successfully.

11/9/25

12/9/25

Task 6 - Implement various text file operation

Aim: To write a python program implent various text file operation

Problem 6.1

write "Error objects are thrown when runtime error occur. The error obj can also be used as base obj for user-defined exceptions" into a text file named log.txt

Algorithm:

- write a file
  - Open a file in
  - write following text to file:
  - "Error message"
  - Close file.
- Read from file:
  - Define readline fn.
  - open file
  - Read entire file content
  - Print content
- Execute the program
  - call write file ("text") to attempt to read from a file named text and print it's content

OBJECT ARE THROWN

RUN TIME ERROR OCCUR

THE ERROR OBJECT CAN ALSO BE USED AS BASE OBJECT FOR USER-DEFINED FUNCTION.

Program 6.1

```
def writefile(name):
    f=open("log.txt","w")
    f.write("Error message")
    f.close()
def readfile(filename):
    with open(filename,"r") as file:
    content=file.read()
    print(content)
writefile("write")
readfile("text")
```

Program 6.2

Text file log.txt containing logs of system write a fn that counts the no. of lines containing the word "ERROR".

Algorithm:

- Initialize Error counter:
  - Define function count_error_lines(name):
    - Initialize error_count to 0
- Open and Read File:
  - Open file specified by filename in read mode using a with statement

6-2

Output

Number of lines with error : 2

- Check each line for "ERROR":
  - If loop through each line in file:
    - If line contains "ERROR"
      - error_count + 1
- Return Error count:
  - After reading all lines return value of error_count.
- Execute program
  - Call count_error_lines("log.txt") to count "ERROR" in "log.txt".
  - Print result "ERROR" (error_lines)".

Program 6.2

```
def count_error_lines(name):
    error_count = 0
    with open(filename, "r") as file:
        for line in file:
            if "ERROR" in line:
                error_count += 1
    return error_count
error_lines =
count_error_lines("log.txt")
print(f"No of lines ERROR : {error_lines}")
```

Problem 6.7

You need to report with details (name, dept) of the emp in list. WA Python function to write this report to a file named employee_report.txt

Algorithm:

1. Create Emp Data:
   - Define the fn write_employee_report (name):
   - create a list containg dictionaries, each with "name" and "dept" key for individual emps.
   - open file for writing
     - open file specified by filename
   - write Employee Data to file
     - Loop through each employee in list.
       - For each emp format a string
       - write formatted string to file, followed by a newline character (\n).
   - Execute program:
     - Call write_emp_report("emp_report.txt")

6.3

output

employees.txt

Name: Alice, Department: HR

Name: Bob, Department: Engineering

Name: Charlie, Department: Finance

Program 6.3

```
def write_emp_rec(filename):
    employees = [{"name": "Alice", "dept": "HR"},
                 {"name": "Bob", "dept": "HR"}
                 {"name": "Rama", "dept": "Finance"}
                 ]
    with open(file, "w") as file:
        for emp in employees:
            f"employee['name']}, Dept: {employee['
            department']}\n"
            file.write(line)

# Example usage
write_employee_report("employees.txt")
```

PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	5
VIVA VOCE (3)	5
RECORD (4)	—
TOTAL (15)	15
SIGN WITH DATE	

Result:

Thus, the python program implement various text file operations was successfully executed and the output was verified

7.1 output

No. of students : 4

type of stu_name list : <class 'list'>

Type of student_grades list : <class 'list'>

Highest grade : 92

Lowest grade : 78

Sorted grades : [78, 85, 90, 92]

Reversed grade : [92, 90, 85, 78]

Cricrade indices from 1 to no. of students : [1, 2, 3, 4]

O/P

19/9/25

Task-7 Utilizing 'Functions' concepts in Python Programming

Aim: To write the python program using 'Functions' concepts in Python Programming

7.1 Python script to analyze and manipulate a list of student grades for class project.

- Python program that satisfies print(), len(), type(), max(), min(), sorted(), reversed(), and range().

Algorithm:

- Start
- Print welcome message
- Print no. using len().
- Print type of the list type() to show type
- Find, print lowest & highest grade max(), min()
- Print sorted list of grades uses sorted()
- Print reversed reversed()
- Enumerate, print a range of grade indices range()
- Stop

7.2 Output

Arithmetic operations:

Sum of 10 & 5 : 15

Difference btwn 10 and 5 : 5

Product of 10 and 5 : 50

Quotient of 10 and 5 : 2.0

Greeting:

Hello, Alice! welcom to the program.

O/P V

```
def a_sg():
    s_name = ["A", "B", "C", "D"]
    s_grade = [85, 92, 72, 82]
    print("Welcome to student grade analy prgm")

    num_s = len(s_name)
    print("No. of students:", num_s)
    print("Type of s_name list:", type(s_name))
    print("Type of s_grade list:" type(s_grade))

    highest_grade = max(s_grade)
    l_grade = min(s_grade)
    print("Highest grade:", h_grade)
    print("Lowest grade:", l_grade
    so_grade = sorted(s_grade)
    print("Sorted grades:", so_grade)
    r_grade = list(reversed(s_grade))
    print("Reversed grades:", r_grade)

    g_indices = list(range(1, num_s + 1))
    print("In grade indices from 1 to no of students:", g_indices)

a_sg()
```

7.2 Create a small calculator application, perform basic arithmetic operations, the application program +, −, ×, /

Algorithm:

- Start
- User input for two numbers
- User input for operation, to choose an arithmetic operation (+, −, /, *)
- Perform operation: Based on user's choice,
- Display Result:
- Stop

```
def add(a,b):
    return a+b
def subtract(a,b):
    return a-b
def multiply(a,b):
    return a*b
def division(a,b)
if b!=0:
    return a/b
else:
    return "Error: Division by zero"
def greet(name):
    return f"Hello, {name}! welcome to the program."
```

Output

```
def main():
    n = 10
    m = 5
print("Arithmetic operations:")
print(f"Sum of {n} & {m}:", add(n,m))
print(f"Difference of {n} & {m}:", subtract(n,m))
print(f"Multiplication of {n} & {m}:", multiply(n,m))
print(f"Division of {n} & {m}:", division(n,m))
user_name = "Alia"
print("In country:")
print(greet(user_name))

if __name__ == "__main__":
    main()
```

VEL TECH - CSE	
EX NO.	7
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	

Result: Thus, the python program using 'Functions' concept was successfully executed & output was verified

Output

8.1 (a)

Enter the starting number: 2

Enter ending number: 5

Enter step value: 1

2

3

4

5

o/p V

26/9/25

26/9/25

Task - 8 Implement python generator and decorators

Aim: Write a python program to implement generator and decorator

8.1

- UAP include generator and decorator fn to produce a seq of nos. The generator should be able to
  - a) Produce seq of nos, start, end, step value
  - b) Produce default seq of nos 0, end to step 1

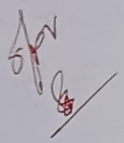
a) Algorithm:

1. Define Generator fn
   - Define fn name seq (start, end, step=1)
   - Initialize Current value
     - set current to value of start
   - Generate Seq:
     - while current is less or equal to end:
     - yield current value of current
     - Increment current step by step

8.1 (b)

output :

0
1
2



Aim: Produce a default sequence of numbers

Start

- Print Generated Sequence
  - Iterate over values by generator obj
  - Print each value

Program:

```
def no_seq(start, end, step=1):
current = start
while current <= end:
    yield current
    current += step
start = int(input("Enter num"))
end = int(input("Enter end"))
seq_generator = no_seq(start end stp
for number in seq_gen:
```

b) Produce a default seq. of nos start from 0 and 10, and with step of 1 of no values

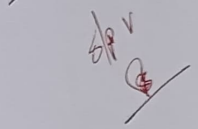
Algorithm:

1. Start Function
   - Define fn my_gen(n) that takes n
   - Initialize Counter:
     - set value to 0
   - Generate values
     - while val < n
     - yield curr value

Output 8.2

HI, I AM CREATED BY A FUNCTION
PASSED AS AN ARGUMENT.

hi, i am created by a function
passed as an argument



• Create & Print values :
  • For each value produced by generator object
  • Print value

b) Program:

```
def my_gen(n):
    value = 0
    while value < n:
        yield value
        value += 1

my_gen
for value in my_gen(3):
    print(value)
```

8.2 Imagine you are working on a messaging app. Two decorators: upper_dec, lower_dec. These dec modify the func they decorate by converting text to various cases. WAP to implement it.

Algorithm:

1. Create Decorators:
   • Define upper_dec to convert to uppercase
   • Define lower_dec to convert to lowercase
2. Define greet fn
   • Accepts a fn as input
3. Execute program
   • Call greet(shout) to print greeting
   • call greet(whisper) to print greeting

Program:

```
def uppercase_decorator(func):
    def wrapper(text):
        return func(text).upper()
    return wrapper

@uppercase_decorator
def greet(text):
    return text

@lowercase_decorator
def whisper(text):
    return text

def get_greeting(func):
    greeting = func("Hi, I am created by a function")
    print(greeting)

get_greeting(greet)
get_greeting(whisper)
```

VEL TECH - CSE	
EX NO	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
...N WITH DATE	

Result:

Thus the python program to implement function generators and decorators was successfully executed & the output was verified

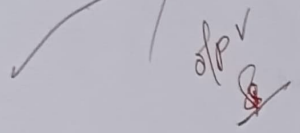
Output

9.1

Grades List : [85, 90, 78, 92, 88]

Enter index of grade you want to view: 3

The grade at index 3 is : 92



10/10/25

Task 9: Implement Exceptions and Exceptional Handling in Python

Aim: To implement exceptions & Exception Handling in Python

Problem: User select a grade by specifying an index number, Ensure that program handles out of range numbers

Algorithm:

- Start
- Initialize list of grades
- Enter index
- Attempt to display index grade at specified index
- If index out of range, catch IndexError, print message "Invalid Index". Please at valid index

Program:

```
grades = [85, 90, 78, 92, 88]
print("Grades list", grades)

try:
    index = int(input("Enter index of grade to view: "))
    print(f"Grade at index {index} is: grades[index]")
except IndexError:
    print("Invalid index")
except ValueError:
    print("Invalid input.")
```

Output

9.2

Enter numerator : 3

Enter denominator : 5

Result : 0.6

o/p ✓

Problem 9.2

Python calculator program to perform basic operations. Dividing numbers by zero, does not allow program to crash if not handled properly

Algorithm:

- Start
- Prompt user to enter 2 nos
- Attempt to divide numerator by denominator
- If den = 0, ZeroDivisionError and display "Error : Division by zero isn't allowed"

Program

```
def div_no.():
    try:
        num = float(input("Enter number:"))
        den = float(input("Enter"))
        result = num/den
        print(f"Result: {result}")
    except ZeroDivisionError:
        print("Error")
    except ValueError:
        print("Error: Please enter valid number")

division operation
divide_numbers()
```

Output

9.3

Enter a number : 24

Eligible to vote

Problem 9.3

Python application to determine whether person is eligible to vote or not 18 or older.

InvalidAgeException raised when an age < 18 entered

Algorithm:

- Define custom exception
- Prompt user for input
- Check if age < 18
- Raise exception if condition met
- Handle exception with custom error message

Program:

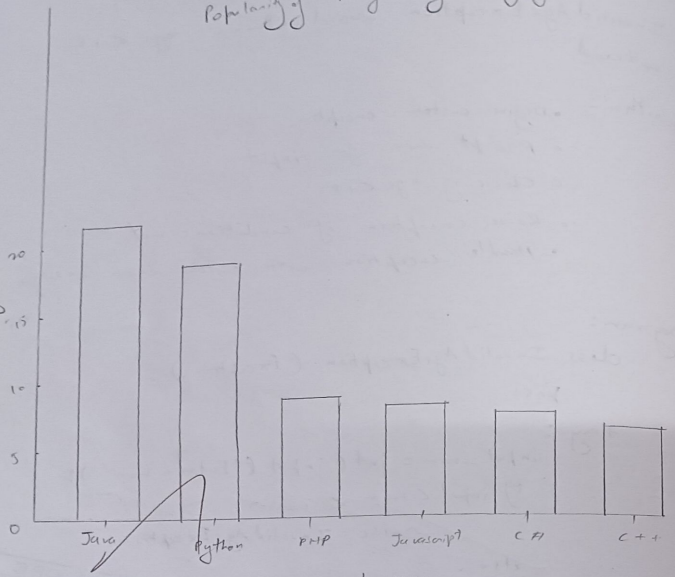
```
class InvalidAgeException (Exception):
    pass
try:
    input_num = int(input("Enter a no:"))
    if input < num number:
        raise InvalidAgeException
    else:
        print("Eligible to vote")
except InvalidAgeException:
    print("Exception occured.
Invalid Age")
```

VEL TECH-CSE	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	

Result:

Thus, program for Implement Exceptions and Exceptions Handling is executed and verified successfully

Output 10.1



17/10/25

Task 10. Use Matplotlib module for plotting in python

Aim: To use Matplot lib module for plotting in python

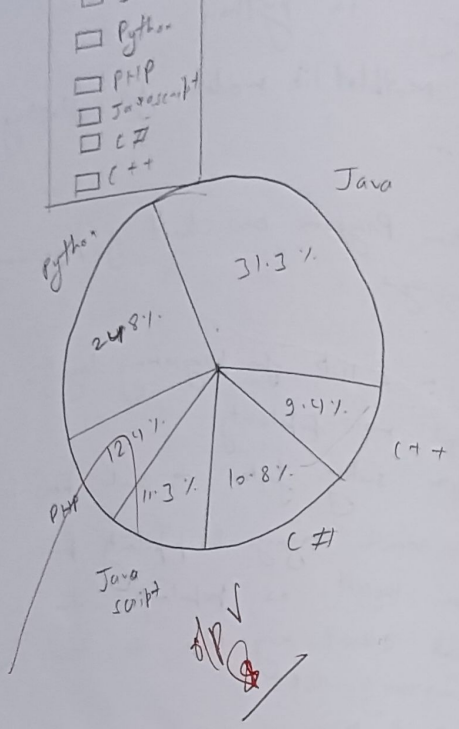
Problem 10.1 Python Program bar chart of programing languages

Algorithm:
- Define 2 list for programing languages & popularity
- Find max popularity
- Define scaling factor to scale bar heights
- For each language & popularity bar, calculate bar height as popularity value by scaling factor
- Print chart using loop to iterate over programing list:
  a) Print language name
  b) Use loop to print bar chart by printing bar chat
  c) Print popularity value
  d) Print newline character

Program:

```
import matplotlib.pyplot as plt
lang = ['Java', 'Python', 'PHP', 'C#', 'C++']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
plt.bar(language, popularity, color='b')
plt.title('Popularity of Programing Languages')
plt.xlabel('Programing Langes')
plt.ylabel('Popularity')
plt.show()
```

Output 10.2



Problem 10.2

Pg WAP python to create a pie chart of popularity of programming languages

Algorithm:

- Create list of Programming Languages and Popularity
- Create a pie chart using matplot library
- set title & legend for pie chart
- show pie chart

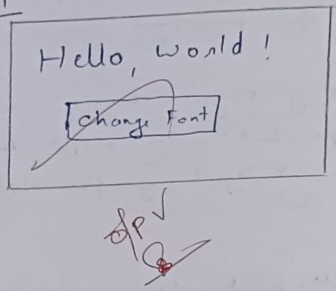
Program:

```
import matplotlib.pyplot as plt
lang = ['Java', 'Python', 'PHP', 'JavaScript', 'C#']
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
plt.pie(popularity, labels = language, autopct = '
  %.1,1f%%.%')
plt.title('Popularity of Programming Languages')
plt.legend(language, loc = 'best')
plt.show()
```

Results: Thus, Python program use Matplotlib module for plotting is executed and verified successfully

VEL TECH-CSE	
EX NO.	10
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	5
TOTAL (20)	20
SIGN WITH DATE	

Output 11.1



Date: 24/10/25

Task 11. Use Tkinter module for UI design

Aim: To use Tkinter module for UI design

Problem 11.1 Python GUI program to create a label and change label font style using tkinter module

Algorithm:

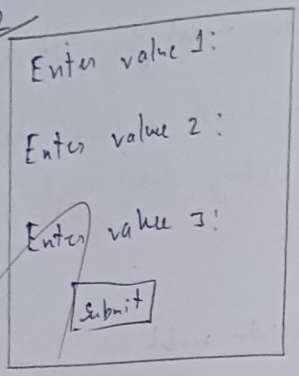
- Import tkinter module
- Create main window
- Create label with desired text
- Add label to main window using pack() method
- Define fn to change font style
- Create button to call the fn
- Add button to main window pack()
- Start main loop

Program:

```
import tkinter as tk

def change_font():
    label.config(font=("Arial", 18, "bold"))

root = tk.Tk()
label = tk.Label(root, text="Hello, World!", font=("Helvetica", 16))
label.pack()
button = tk.Button(root, text="Change Font", command=change_font)
button.pack()
root.mainloop()
```

Output

11.2



o/p

11.2 write a python GUI program to create 3 single line text-box to accept a value for the using tkinter module

Algorithm:

- Import tkinter module
- Create main window
- Add labels & text boxes to the main window
- Set the size of text-boxes
- Create a button to submit the values entered in text boxes
- Get the values entered in the text-boxes when the button is clicked.
- Close the main window when the button is clicked

Program:

```
import tkinter as tk

root = tk.Tk()
root.title("Input")
label1 = tk.Label(root, text="Enter value 1:")
entry1 = tk.Entry(root)
```

```
label 2 = tk.Label (root, text = "Enter value 2 :")
entry 2 = tk.Entry (root)
label 3 = tk.Label (root,
entry 3 = tk.Entry (root)
entry 1.config (width = 30)
entry 2.config (width = 30)
entry 3.config (width = 30)
def get_values ():
    val1 = entry 1.get ()
    print ("value 3", val3)
label 2.pack ()
entry 3.pack ()
submit_button.pack ()
root.mainloop ()
```

VEL TECH - CSE	
EX NO.	11
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	

Result :

Thus, the Program using Tkinter module for UI design was executed and verified successfully

31/10/25

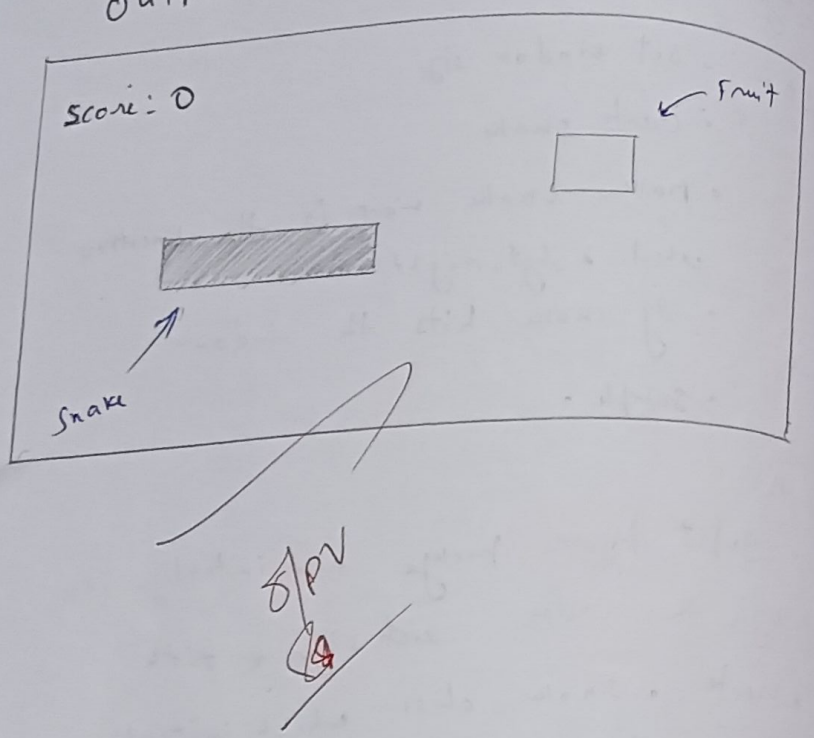
Task 12: Simulate Gaming concepts using Pygame package.

Conditions:
- set window size
- Create snake
- Make snake move in the direction called (left, right)
- If snake hits the window
- ~~Sample~~.

Algorithm:
- Import pygame package and initialize it
- Define the window with size & title
- Create a snake class which initializes pos, color and movements
- Create a Fruit class which initializes fruit pos and color
- Create a fn to check if the snake collides with the window and end the game
- Create fn to update snake position based on mov
- Create fn to update game display draw the snake fruit
- Create a game loop to continuously update game disp
- End the game if user quits or snake collides with

12

OUTPUT



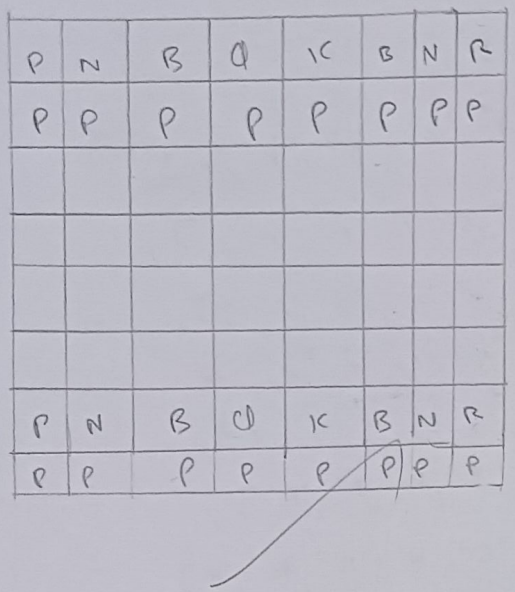

```
import pygame
import sys

pygame.init()
screen_size = (640, 640)
screen = pygame.display.set_mode(screen_size)
pygame.display.set_caption('Chess Board')
black = (0,0,0)
white = (255, 255, 255)
font = pygame.font.SysFont('arial', 50, bold=True)
def draw_board():
    square_size = 80
    for row in range(8):
        for col in range(8):
            color = white if (row + col) % 2 == 0
else black
            pygame.draw.rect(screen, color, (col * square_size, row
square_size, square_size, square_size))
def draw_pieces(board):
    square_size = 80
    for row in range(8):
        for col in range(8):
            piece = board[row][col]
            if piece != ' ':
if piece.isupper():
            text_color = (240, 240, 240)
```

```
text_surface = font.render(piece.name(), True, (0,0,0))
screen.blit(text_surface, text_rect)

board = [
    ['r','n','b','q','k','b','n','r'],
    ['p','p','p','p','p','p','p','p'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['.','.','.','.','.','.','.','.'],
    ['P','P','P','P','P','P','P','P'],
    ['R','N','B','Q','K','B','N','R']
]

running = True
while running:
        screen.fill(white)
        draw_board()
        draw_pieces(board)
        pygame.display.update()

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False

pygame.quit()
sys.exit()
```

Result: Thus the program for pygame is successfully executed & verified

VEL TECH-CSE	
EX NO.	12
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	—
TOTAL (20)	15
SIGN WITH DATE	XXX

Output 12.2



Program:

```
import pygame
import time
import random

snake_speed = 15

window_x = 720
window_y = 480

black = pygame.Color(0,0,0)
white = pygame.Color(255,255,255)
red = pygame.Color(255,0,0)
green = pygame.Color(0,255,0)
blue = pygame.Color(0,0,255)

pygame.init()

pygame.display.set_caption('Welcome for Snake Game')
game_window = pygame.display.set_mode((window_x,
window_y))

fps = pygame.time.Clock()

snake_position = [100,50]

snake_body = [[100,50],
              [90,50]
              [80,50]
              [70,50]]
```

```
fruit_posit = [random.randrange(1, (window_x//10))*10,
               random.randrange(1, (window_y//10))*10]

fruit_spawn = True

direction = 'Right'

change_to = direction

score = 0

def show_score(choice, color, font, size):
    score_font = pygame.font.SysFont(font, size)

if direction == 'UP':
    snake_position[1] -= 10

if direction == 'DOWN':
    snake_position[1] += 10

if direction == 'RIGHT':
    snake_position[0] += 10

snake_body.insert(0, list(snake_position))

if snake_position[0] == 10
    score += 10
    fruit_spawn = False
else:
    snake_body.pop()

if not fruit_spawn:
    fruit_position = [random.randrange(1, (window_x//10))
                      * 10,
```

```
if snake_position [0] == block [0] and snake_
position [1] > window_y-10:
    game_over()
for block in snake_body [1:]:
    if snake_position [0] == block [0] and snake
position [1] == block [1]:
        game_over

show_score (1, white, 'times new roman', 20)
pygame.display.update()
fps.tick (snake_speed)
```

Problem 2 :

write a python program to develop a chess board using pygame

Algorithm :

- import pygame and initialize it
- set screen size & title
- Define colors for board & pieces
- Define a function to draw the pieces on board by loading image for each piece and placing them on corresponding sq
- Define initial state of the board as a list
- Draw the board and pieces on screen
- start the game loop

1/11/26

USE CASE: 1

use case - Finding the winning strategy in a card game in python

problem Description: Imagine a card game where each player recives a hand of cards with values. Objective is to find best way to maximize the score for a player

Assumptions:

- Each player tries to maximize their score
- Cards are represented by integers, indicating values
- 2 players, alternate turns, and each player picks a card from either begining or end

plan: we can solve this problem using Dynamic Programming by calculating the optimal score for every possible scenario, for players.

Steps:

- Define the game: Represent the pile of cards as a list of integers
- Recursive strategy: A fn will recursively determine the best score a player can
- Dynamic Programing: Store intermediate results to avoid recalculating the

Explanation:

- Dynamic Programming Table (dp):
  Each cell dp[i][j] represents the diff in score btwn 1st player & opponent in game i to index j
- Two choices: For each move, either
  1. Pick leftmost card, leaving opponent to play optimally on remaining cards
  2. Pick rightmost card cards[j] leaving opponent rest of cards'
- Recursive Relation: The value of each subproblem is determined by maximizing the score diff btwn current player & opponent.

Example walkthrough:
Consider the array of cards: [3,9,1,2].

1. First player (you) can choose btwn:
   - Taking leftmost card [3], leaving [9,1,2]
   - Taking rightmost card (2), leaving [3,9,1]
- The opponent will then take their turn, playing optimally to minimize first player's score

First player's optimal score: 5

Program:

```
def find_opt_strategy(cards):
    n = len(cards)
    dp = [[0]*n for _ in range(n)]
    if i == j:
        dp[i][j] = cards[i]
    else:
    choices:
        take_left = cards[i] - dp[i+1][j]
        take_right = cards[j] - dp[i][j-1]
        dp[i][j] = max(take_left, take_right)
    return (dp[0][n-1] + sum(cards))//2
cards = [3,9,1,2]
print("First player's optimal score:",
find_optimal_strategy(cards))
```

Optimizing strategy:
By using Dynamic Programming, we ensure that the solution is computed efficiently, avoiding redundant calculations. This approach ensures both players play optimally, and the first player gets the highest score possible given opponent's best move

VEL TECH-CSE	Use case
EX NO.	
PERFORMANCE (5)	5
RESULT AND ANALYSIS (5)	5
VIVA VOCE (5)	5
RECORD (5)	
TOTAL (20)	15
SIGN WITH DATE	