1.	**Attempt *any three* of the following:**	**15**
a.	**Define the term "Debugging". What are the different types of errors in programming?** The process of identifying and removing errors is referred to as debugging. **Types of Errors in Programming** • **Syntax Errors** – Syntax is the structure of statements in a program. – A syntax error is an error in the syntax of a programming language, i.e., an error in the source code of a program. – If syntax errors are present in the program then the compilation of the program fails and is terminated after showing the list of errors and the line number where the errors have occurred. – In some cases the line number may not exactly indicate the correct place of the error. – In some other cases, a single syntax error can result in a long list of errors. – Correction of one or two errors in the program may remove the entire list of errors. – E.g. print("Hello World) • **Semantic Errors** – A semantic occurs due to the wrong use of the variables. – The syntax of the programming language would be correct but the problem may be caused due to the use of wrong variables, wrong operations or in a wrong order. – Semantics errors are not traced by the compiler, as a result of which the program executes but does not return the desired result. – Semantic errors occur during the execution of the code. – To track a semantic error, the programmer must check the output of the program. – E.g. Subscript out of range, forgetting to divide by 100 when calculating a percentage amount. • **Logical Errors** – As the name itself implies, these errors are related to the logic of the program. – Logical error is a mistake in a program's source code. – Logical errors are also not detected by compiler and cause incorrect results. – To trace a logical error, the programmer must check the logic / algorithm of the program. – E.g. Infinite loop a=10 while (a>0) { …… …… } • **Compile Time and Runtime Errors** – An error that occurs during the compilation of a program is known as a compile time error. E.g. Syntax Error – An error that occurs during the execution of a program is known as runtime error. E.g. Dynamic Semantic Errors, Logical Errors.	

b. **List and explain the data types in Python.**

- **Number**
  - **int**: Represents negative and positive integers without fractional part.
  - **float**: Represents negative and positive numbers with fractional part.
  - **complex**: A complex number consists of an ordered pair of real floating-point numbers denoted by x + yj, where x and y are the real numbers and j is the imaginary unit.

- **Bool**
  - Represents the truth values False and True.

- **String**
  - Strings start and end with single or double quotes.
  - Single and double quoted strings are same and a single quote can be used within a string when it is surrounded by double quote and vice versa.
  - Triple quotes are used to span the string across multiple lines.
  - **Special characters in strings**: The backslash (\) character is used to introduce a special character.
    - ⇨ \n Newline
    - ⇨ \t Horizontal Tab
    - ⇨ \\ Backslash
    - ⇨ \' Single Quote
    - ⇨ \" Double Quote

- **List**
  - A list is a collection of elements of same or different data types, separated with commas and enclosed within brackets.

- **Tuple**
  - A list is a collection of elements of same or different data types, separated with commas and optionally enclosed within parentheses.

- **Dictionary**
  - A dictionary is a collection of key:value pairs, separated with commas and enclosed within braces.

- **Set**
  - A set is a collection of elements of same or different data types, separated with commas and enclosed within braces.

c. **Write a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.**

```
n = input("Enter your Name:")
a = int(input("Enter your Age:"))

diff = 100 - a
year = 2018 + diff

print("\n\n", n, "will turn 100 years old in the year", year)
```

d. **Write a program to accept a number from the user and count its number of divisors.**

```
n = int(input("Enter a number:"))
div=0
for x in range(1, n+1):
        if(n%x ==0):
                div+=1
print("\nNo. of Divisors =", div)
```

e. **Explain the continue statement with a suitable example**.

The continue statement is used in a while or for loop to take the control to the beginning of the loop without executing the rest statements inside the loop

Syntax:

```
while (condition1) :
    statement_1
    statement_2
    .....
    if condition2 :
      continue
```

```
for variable_name in sequence :
    statement_1
    statement_2
    ……
    if condition1:
      contiune
```

Example

f. **Explain the use of Logical Operators and Membership Operators in Python**.

- **Python Logical Operators**

Logical operators are used to combine conditional statements

Operator	Example	Result
and	(x and y)	is True if both x and y are true.
or	(x or y)	is True if either x or y is true.
not	(x not y)	If a condition is true then Logical not operator will make false.

E.g.

Command	Output
>>> x = 9	
>>> y = 11	
>>> x > 10 and y > 10	False
>>> x > 10 or y > 10	True
>>>not(x>10 and y> 10)	True

- **Python Membership Operators**

Membership operators are used to test the membership of elements in sequences like strings, list and tuples.

Operator	Example	Result
in	x in y	Returns True if an element with the specified value exists in the sequence
not in	x not in y	Returns True if an element with the specified value does not exist in the sequence

E.g.

Command	Output
>>> x = [10, 20, 30, 40, 50]	
>>> y = 15	
>>> y in x	False
>>> y not in x	True

2.	**Attempt *any three* of the following:**	**15**
a.	**Explain the concept of composition in functions with a suitable example**. • Composition means having a container object that contains other objects. • Composition means that an object knows another object, and explicitly delegates some tasks to it. • Composition can be used in two ways: – Calling one function from another function. – Combining functions in such a way that the return value of one function is passed as an argument to another function. **Example**	
b.	**Write a program to print the sum of natural numbers using recursive function**. `def sum_natural(n):` `  if (n==1):` `  return 1` `  else:` `  return n + sum_natural(n-1)` `num = int(input("Enter stop value:"))` `print("Sum = ", sum_natural(num))`	
c.	**Write a function to check whether the number entered by the user is an Armstrong Number.** `def armstrong(num):` `  s = 0` `  temp = num` `  p = len(str(num))` `  while (temp != 0):` `  digit = temp % 10` `  s += digit ** p` `  temp //= 10` `  if (s == num):` `  print(num, "is an Armstrong Number")` `  else:` `  print(num, "is not an Armstrong Number")` `n = int(input("Enter a Number:"))` `armstrong(n)`	
d.	**What is a fruitful function? Explain with the help of an example**. A function that returns a value, is called Frutiful Function **Example**	
e.	**How can strings be traversed with a loop? Give suitable example**. • for loop is used to iterate through a string. • Traversal is a method which starts with the first character in a string, and selects each character one at a time, up to the last character. • A string can be traversed using for loop as well as while loop.	

	Example: s = "BScIT" for c in s: print(c) print("\n\n") i = 0 while i < len(s): print(s[i]) i+=1	
f.	**Explain the following string functions with example:** **i) startswith()** **ii) rstrip()** **i)** startswith(str ,begin=0,end=n) It returns a Boolean value if the string starts with given str between begin and end. **ii)** rstrip() It removes all trailing whitespace of a string and can also be used to remove particular character from trailing. **Exmple**	
3.	**Attempt *any three* of the following:**	**15**
a.	**What are the different methods for adding elements to a list? Give example for each method.** • append() method is used to add a single element at the end of the list. • extend() method is used to add multiple elements at the end of the list. • insert() method is used to add a single element at the desired location. **Example**	
b.	**Explain the use of slice operator for accessing elements of a tuple**. • Slice operator can be used with tuples to extract a part of the tuple. – [n] Extracts element at specified index number. – [n:m] Extracts elements from index 'n' to index 'm' (Does not extract element at index 'm'). – [n:m:c] Extracts elements from index 'n' to index 'm', skipping 'c' number of elements. (Does not extract element at index 'm'). – [n:] Extracts all elements starting at index 'n' upto the end of the list. – [:m] Extracts all elements upto index 'n' (excluding element at index 'm'). – [n][m] Extract element from a nested tuple.	
c.	**What is an Exception? Identify the exception that will be raised by the following code. Justify your answer.** **i) import math** **print("\nPower =", math.power(10,2))** **ii) x = 10** **y = '10'** **print (x + y)** **iii) s = "BScIT"** **for i in range(len(s)):** **print(s[i+1])** **iv) x = int(input("Enter a Number:")** **print (x)**	

	Exception is any abnormal condition in a program that results in disruption of the flow of the program. Whenever an exception occurs the program displays an error message and stops executing the program. **i)** AttributeError – power() method does not belong to math module **ii)** TypeError – cannot add int and str **iii)** IndexError – the last iteration tries to access s[5], which does not exist **iv)** SyntaxError – there should be 2 closing brackets at the end of first line	
d.	**Write a Python program to take a character from the user and search that character in the file. If the character is present then print total count of that character in the file or else display the message "No such character".** `count = 0` `f=open("file1.txt", "r")` `text=f.read()` `char = input("Enter any character:")` `for c in text:` `  if c == char:` `  count +=1` `if count == 0:` `  print("No such character")` `else:` `  print("{} appears {} times in the file".format(char, count))`	
e.	**What is Exception Handling? How does it work?** Exceptions (errors) are resolved by interrupting the normal flow of the program to execute a special function or block of code, known as the exception handler **Working:** • The code that can generate an error can be handled by using the try block. • The code which could raise an exception is placed inside the try block. • The try block is followed by except blocks which contain code to be executed if the exception occurs. • Except block specifies the exception which can occur and the corresponding code to be executed if exception occurs. • It is further followed by else block which contains code to be executed if no exception occurs. **Syntax:** try: code that can generate an exception except Exception1: execute code except Exception2: execute code .... except ExceptionN: execute code else: Executes the else block if no exception occurs	

f.	**Write a program to sort a dictionary in ascending and descending order of values**. `sub = {'s1':'PP', 's2':'DS', 's3':'CN', 's4':'DBMS', 's5':'AM/MP'}` `sub2={}` `for k,v in sub.items():` `  sub2.update({v:k})` `print("\n Dictionary in ascending order of Values:")` `for k in sorted(sub2):` `  print ("{}:{}".format(sub2[k], k))` `print("\n Dictionary in descending order of Values:")` `for k in sorted(sub2, reverse=True):` `  print ("{}:{}".format(sub2[k], k))`	
4.	**Attempt *any three* of the following:**	**15**
a.	**What is method overloading? Write a program to demonstrate method overloading.** • Defining a method in such a way that there are multiple ways to call it, is known as method overloading. • Depending on the method definition, it can be called with zero, one, two or more parameters **Program**	
b.	What are the methods of Thread Class? – **start()** ▪ Start the thread's activity. ▪ It must be called only once for each thread object. It invokes the object's run() method. ▪ This method will raise a RuntimeError if called more than once on the same thread object. – **run()** ▪ This method is the entry point for the thread. – **join([timeout])** ▪ This method waits for a thread to terminate. ▪ The default value for timeout is None, which indicates that the method will wait till the thread terminates. ▪ The timeout value must be specified as a floating point value (specifying the timeout value in seconds or fractions thereof). – **isAlive()** ▪ Checks whether a thread is still executing. ▪ This method returns True just before the run() method starts until just after the run() method terminates. – **getName()** ▪ The getName() method returns the name of a thread. – **setName()** ▪ The setName() method sets the name of a thread.	
c.	**Create a module "Area.py" with functions area_circle(), area_triangle() and area_rect(). Create a new file. Use area_circle(), area_triangle() and area_rect() from the Area module to calculate the areas.**	

Area.py

```
import math
def area_circle(r):
    a = math.pi * r * r
    return a

def area_triangle(b, h):
    a = 1/2 * b * h
    return a

def area_rect(l, b):
    a = l * b
    return a
```

Shape.py

```
import area
r = int(input("Enter radius of circle:"))
print("Area of circle = " , area.area_circle(r))

b = int(input("\nEnter base of triangle:"))
h = int(input("Enter height of triangle:"))
print("Area of triangle = " , area.area_triangle(b ,h))

b = int(input("\nEnter breadth of rectangle:"))
ℓ = int(input("Enter length of rectangle:"))
print("Area of triangle = " , area.area_rect(ℓ, b))
```

d. **What is multilevel inheritance? Write a program to implement multilevel inheritance.**

Deriving a class from another derived class is known as Multilevel Inheritance

Program

e. **How are threads synchronized?**

- Thread synchronization is defined as a mechanism which ensures that two or more concurrent threads do not simultaneously access the same resource, e.g., variable.
- The threading module provided with Python includes a locking mechanism that allows synchronization of threads.
- A new lock is created by calling the Lock() method, which returns the new lock.
- The acquire(*blocking*) method of the new lock object is used to force the threads to run synchronously.
- When invoked with the *blocking* argument set to True or 1 (the default), block until the lock is unlocked with release().
- When invoked with the *blocking* argument set to False or 0, do not block.

The release() method of the new lock object is used to release the lock when it is no longer required.

f. **Explain the replace() and split() Regular Expression methods with suitable examples.**

- **replace()**
  - Replaces all occurrences of a substring with another substring.

  Syntax

  string.replace (old, new, count)

  - old is the substring to be searched.
  - new is the substring that should replace old substring.
  - count is the number of occurrences to be replaced.

- **split()**
  - Splits a sting into a list of substrings.

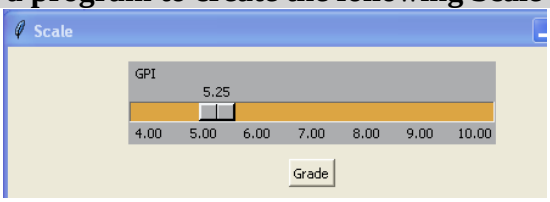
  Syntax

  string.split (RE)

  - RE is the specified split expression.
  - If no expression is specified, the default split character is whitespace.

Example

5.	**Attempt *any three* of the following:**	**15**
a.	**Write a program to create the following Scale Widget:**	



Start Value: 4, End Value: 10, Increment: 0.1, Tick Interval: 1, Label: GPI. When the user selects a value and clicks on the button for Grade, it should display the corresponding grade as a messagebox

GPI	**Grade**
10.00	**O**
9.00 – 9.99	**A+**
8.00 – 8.99	**A**
7.00 – 7.99	**B+**
6.00 – 6.99	**B**
5.00 – 5.99	**C**
4.00 – 4.99	**D**

```
import tkinter
window = tkinter.Tk()
window.title("Scale")
window.geometry('500x500')
def grade():
    if gpi.get() == 10.00:
        g = 'O'
    elif gpi.get() >= 9.00:
        g = 'A+'
    elif gpi.get() >= 8.00:
        g = 'A'
    elif gpi.get() >= 7.00:
        g = 'B+'
    elif gpi.get() >= 6.00:
        g = 'B'
    elif gpi.get() >= 5.00:
        g = 'C'
    elif gpi.get() >= 4.00:
        g = 'D'
    m = tkinter.messagebox.showinfo("Grade", g)

gpi = tkinter.DoubleVar()
s = tkinter.Scale(window, label='GPI', from_=4, to=10, resolution=0.01,
            tickinterval=1, orient='horizontal', length=300, variable=gpi)
b = tkinter.Button(text='Grade', command=grade)
s.pack(padx=10, pady=10)
b.pack()
window.mainloop()
```

b.	**Write a program to create a popup menu with two options – Black and White. The background color of the window should change when user selects the option.** `import tkinter` `window = tkinter.Tk()` `window.title("Menubutton")` `window.state('zoom')` `def colorW():` `  window.configure(background="white")` `def colorK():` `  window.configure(background="black")` `popup = tkinter.Menu(window, tearoff=0)` `popup.add_command(label='White', command=colorW)` `popup.add_command(label='Black', command=colorK)` `def disp_popup(event):` `  popup.tk_popup(event.x_root, event.y_root)` `  popup.grab_release()` `window.bind("<Button-3>", disp_popup)` `window.mainloop()`	
c.	**What is the use of Entry Widget? Explain any five properties of Entry Widget.** • A widget to accept single-line text from the user. **Properties** • **bg** – Normal background color. • **fg** – Normal text color. • **bd** – Border width in pixels. Default is 2. • **selectbackground** – Background color of the widget when text is selected. • **selectforeground** – Text color when the text in the widget is selected. • **width** – Width of the widget in characters. • **relief** – Specifies border type. – Values are sunken, raised, groove, ridge. – Default is flat. • **textvariable** – Control variable of class StringVar that can be used to retrieve the value entered in the widget.	

d. **Explain the standard attribute "Font" along with its options.**

- Changes the font of text.

Syntax

```
font = font.Font(options)
```

Options

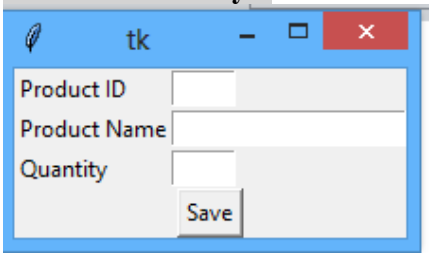
- **family**
  - The font family name.
- **size**
  - The font height as an integer in points. To get a font n pixels high, use -n.
- **weight**
  - "bold" for boldface, "normal" for regular weight.
- **Slant**
  - "italic" for italic, "roman" for unslanted.
- **underline**
  - 1 for underlined text, 0 for normal.
- **overstrike**
  - 1 for overstruck text, 0 for normal.

e. **What are the different functions to retrieve rows from a table? Explain with a suitable example.**

- The fecthone() method retrieves the next row of a query result set and returns a tuple with the values of the row, or None if no more rows are available.
- The fecthmany() method retrieves the next specified number of rows from the table and returns a list of tuples.
- The fetchall() method retrieves all (or all remaining) rows of a query result set and returns a list of tuples. If no more rows are available, it returns an empty list.
- rowcount property returns the number of rows in a cursor.

```
import mysql.connector as mysql
conn = mysql.connect(user="root", password="", host='127.0.0.1', db='myDB')
cursor = conn.cursor()
cursor.execute("SELECT * FROM employee")
row = cursor.fetchone()
print(row)
rows = cursor.fetchmany(2)
print(rows)
all = cursor.fetchall()
print(all)
print("Total no. of rows = " , cursor.rowcount)
conn.close()
```

f. **Write a program using the following layout to save product details in the Product Table (pro_id, pro_name, quantity) and display the message "Record saved successfully".**



```
import tkinter
import mysql.connector as mysql
window=tkinter.Tk()

conn = mysql.connect(user="root", password="", host='127.0.0.1', db='myDB')
cursor = conn.cursor()

pid = tkinter.StringVar()
pname = tkinter.StringVar()
qty = tkinter.StringVar()

def save():
    sql = "INSERT INTO product VALUES(" + pid.get() + ", '" + pname.get() + "', " +
qty.get() + ");"
    try:
        cursor.execute(sql)
        m = tkinter.messagebox.showinfo("Save", "Record Successfully Saved")
        pid.set('')
        pname.set('')
        qty.set('')
        conn.commit()
    except mysql.Error as err:
        print(err)

L1 = tkinter.Label(window, text="Product ID").grid(row=1,column=1, sticky='w')
L2 = tkinter.Label(window, text="Product Name").grid(row=2,column=1, sticky='w')
L3 = tkinter.Label(window, text="Quantity").grid(row=3,column=1, sticky='w')

p_id = tkinter.Entry(window, width=5, textvariable=pid).grid(row=1,column=2,
sticky='w')
p_name = tkinter.Entry(window, textvariable=pname).grid(row=2,column=2,
sticky='w')
p_qty = tkinter.Entry(window, width=5, textvariable=qty).grid(row=3,column=2,
sticky='w')

b1 = tkinter.Button(window, text="Save", command=save).grid(row=5, column=1,
columnspan=2)

window.mainloop()
conn.close()
```