Model Question Paper-sem I/II with effect from 2022-23 (CBCS Scheme)

USN | | | | | | | | | | |

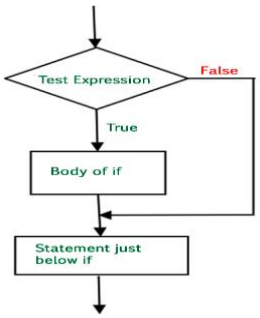
First/Second Semester B.E. Degree Examination

Introduction to Python Programming

TIME: 03 Hours **Max. Marks: 100**

Note: 01. Answer any **FIVE** full questions, choosing at least **ONE** question from each **MODULE**.

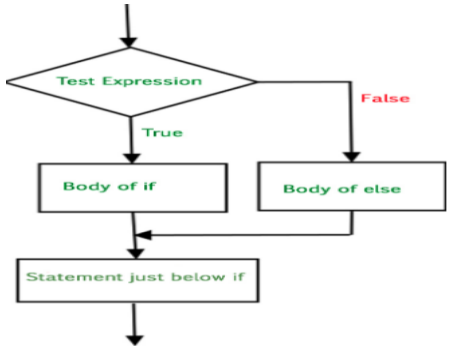
		Module -1	Bloom's Taxonomy Level	Marks
Q.01	a	What is an arithmetic expression? . What is the output of this statement? **'hello world' + 100 + 'how are you'** explain the reason if the statement produces an error. An arithmetic expression is a combination of numbers, operators, and parentheses that represents a mathematical calculation. The operators can be addition (+), subtraction (-), multiplication (*), division (/), exponentiation (^), or any other valid arithmetic operator. **'hello world' + 100 + 'how are you'** **OUTPUT:** TypeError: can only concatenate str (not "int") to str The above statement will produce an error because you cannot concatenate a string and an integer without converting the integer to a string first. Here's what happens when you run the statement: The string 'hello world' is concatenated with the integer 100, which causes a TypeError because you can only concatenate strings together.	CO2-L3	6
	b	Discuss various methods of importing modules in Python programs. Which method is best?. Explain. To use functions, classes, or variables defined in a module, it needs to be imported in the program. There are several ways to import modules in Python programs, including: 1. **import module_name**: This statement imports the entire module and allows you to use its functions, classes, and variables with the **module_name.** prefix. For example: import math print(math.pi) 2.	CO1 – L1	7

		`from module_name import function_name`: This statement imports a specific function from a module and allows you to use it without the **`module_name.`** prefix. For example: `from math import pi` `print(pi)` **`3.`** **`import module_name as alias_name`**: This statement imports a module and gives it an alias name to use in the program. For example: `import numpy as np` `arr = np.array([1, 2, 3])` **`4.`** **`from module_name import *`**: This statement imports all functions, classes, and variables from a module into the program's namespace. However, this method is generally discouraged because it can lead to naming conflicts and make the code harder to read and understand.		
	c	What is the lambda function? Explain with an example of addition of two numbers. In Python, a lambda function is a small anonymous function that can take any number of arguments, but can only have one expression. Lambda functions are useful when you need a simple function for a short period of time and do not want to define a full function using the **`def`** statement. Lambda functions are defined using the keyword **`lambda`**, followed by the function's arguments and expression. Here is an example of a lambda function that adds two numbers: `addition = lambda x, y: x + y` `print(addition(2, 3))` In this example, we defined a lambda function called **`addition`** that takes two arguments **`x`** and **`y`** and returns their sum. The lambda function is then called with the arguments **`2`** and **`3`**, and the result **`5`** is printed to the console.	CO2 – L1	7
		OR		
Q.02	a	What is a flow control statement?. Discuss **if** and **if else** statements with flow chart. A flow control statement is a programming statement that controls the flow of execution in a program. It allows the program to make decisions and choose different paths of execution based on certain conditions. The two most common flow control statements in Python are **`if`** and **`if-else`**. **Simple if:** *If statements* are control flow statements that help us to run a particular code, but only when a certain condition is met or satisfied. A *simple if* only has one condition to check. 	CO1 - L1	7

EXAMPLE:

```
n = 10
if n % 2 == 0:
print("n is an even number")
```

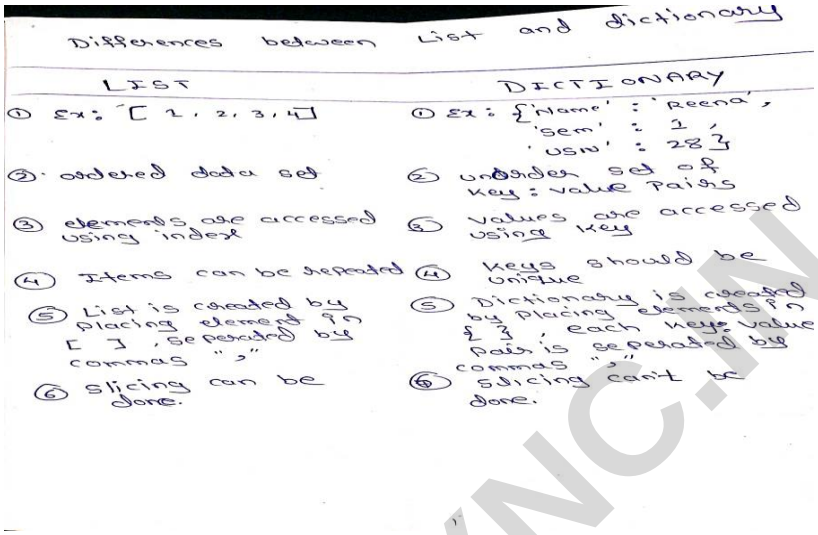
if-else: The *if-else statement* evaluates the condition and will execute the body of if if the test condition is True, but if the condition is False, then the body of else is executed.



EXAMPLE:

```
n = 5
if n % 2 == 0:
print("n is even")
else:
 print("n is odd")
```

	b	Write a python program to add **n** numbers accepted from the user. `n = int(input("How many numbers do you want to add? "))` `total = 0` `for i in range(n):` `  num = float(input("Enter a number: "))` `  total += num` `print("The sum of the entered numbers is:", total)` **OUTPUT:** How many numbers do you want to add? 5 Enter a number: 1 Enter a number: 2 Enter a number: 3 Enter a number: 4 Enter a number: 5 The sum of the entered numbers is: 15.0	CO2 – L3	7
	c	How can you prevent a python program from crashing? discuss different ways to avoid crashing. There are several ways to prevent a Python program from crashing. Here are some of them: **Input validation:** One common cause of program crashes is invalid input. You can prevent this by validating user input before processing it. **Error handling:** Python has built-in error handling mechanisms, such as try-except blocks, which can catch and handle errors that may cause a program to crash. **Memory management:** You can prevent these types of errors by managing	CO2 – L2	6

		memory carefully, such as releasing unused memory or avoiding infinite loops. **Testing and debugging**: Finally, one of the best ways to prevent program crashes is to thoroughly test and debug your code. This includes writing unit tests to validate your code and using debugging tools, such as print statements or a debugger, to identify and fix issues before they cause a crash.		
		Module-2		
Q. 03	a	Discuss list and dictionary data structure with example for each.  	CO2 – L1	6
	b	write a python program to accept n numbers and store them in a list. Then print the list without ODD numbers in it. **n = int(input("How many numbers do you want to enter? "))** **numbers = []** **for i in range(n):** **num = int(input("Enter a number: "))** **numbers.append(num)** **even_numbers = [num for num in numbers if num % 2 == 0]** **print("The even numbers in the list are:", even_numbers)** OUTPUT: How many numbers do you want to enter? 5 Enter a number: 3 Enter a number: 4 Enter a number: 5 Enter a number: 2 Enter a number: 6 The even numbers in the list are: [4, 2, 6]	CO2 – L2	6
	c	For a=[‘hello’, ‘how’, [1,2,3], [[10,20,30]]] what is the output of following statement (i) print(a[: :]) (ii) print(a[-3][0]) (iii) print(a[2][: -1]) (iv) print(a[0][: : -1])	CO2 – L3	8

		For the given list **a=['hello', 'how', [1,2,3], [[10,20,30]]]**, the output of the following statements will be: (i) **print(a[::])** will print the entire list **a** as it is. The output will be: ['hello', 'how', [1, 2, 3], [[10, 20, 30]]] (ii) **print(a[-3][0])** will print the first element of the third element of the list **a**. The third element of **a** is the list **[1, 2, 3]**, and the first element of that list is **1**. The output will be: 1 (iii) **print(a[2][:-1])** will print all the elements of the third element of the list **a**, except the last element. The third element of **a** is the list **[1, 2, 3]**, and all its elements except the last element is **[1, 2]**. The output will be: [1, 2] (iv) **print(a[0][::-1])** will print the first element of the list **a** in reverse order. The first element of **a** is the string **'hello'**, and its reverse order is **'olleh'**. The output will be: **'olleh'**		
		OR		
Q.04	a	write a python program to read dictionary data and delete any given key entry in the dictionary. `# Read the dictionary from the user` `n = int(input("How many key-value pairs do you want to enter? "))` `d = {}` `for i in range(n):` `  key = input("Enter a key: ")` `  value = input("Enter a value: ")` `  d[key] = value` `# Get the key to delete from the user` `key_to_delete = input("Enter the key to delete: ")` `# Delete the key from the dictionary if it exists` `if key_to_delete in d:` `  del d[key_to_delete]` `  print("Key deleted successfully.")` `else:` `  print("Key not found in dictionary.")` `# Print the updated dictionary` `print("Updated dictionary:", d)` **OUTPUT:** How many key-value pairs do you want to enter? 3 Enter a key: 1 Enter a value: 10 Enter a key: 2 Enter a value: 20 Enter a key: 3 Enter a value: 30 Enter the key to delete: 2 Key deleted successfully.	CO3 – L23	7

		Updated dictionary: {'1': '10', '3': '30'}		
	b	Explain different clipboard functions in python used in wiki markup Here are some of the commonly used clipboard functions in Python that can be used in wiki markup: **pyperclip.copy(text)** - This function copies the given text to the clipboard. The **text** argument is a string that contains the text to be copied. **pyperclip.paste()** - This function retrieves the current contents of the clipboard and returns it as a string. **pyperclip.cut()** - This function cuts the current selection from the clipboard and returns it as a string. **pyperclip.clear()** - This function clears the contents of the clipboard.	CO2 – L2	6
	c	Using string slicing operation write python program to reverse each word in a given string (eg: **input**: "hello how are you", **output**: "olleh woh era uoy") ``` # Reverse each word of a Sentence # Function to Reverse words def reverseword(s): w = s.split(" ") # Splitting the Sentence into list of words. # reversing each word and creating a new list of words # apply List Comprehension Technique nw = [i[::-1] for i in w] # Join the new list of words to for a new Sentence ns = " ".join(nw) return ns # Driver's Code s = input("ENTER A SENTENCE PROPERLY ::") print(reverseword(s)) ``` OUTPUT: ENTER A SENTENCE PROPERLY :: hello how are youolleh woh era uoy	CO2 – L3	8
		Module-3		
Q. 05	a	Discuss different paths of file system. In Python, the **os** module provides several functions for working with the file system. These functions allow you to manipulate file paths, check if files or directories exist, create new directories, and perform other operations on the file system. Here are some of the commonly used paths of file system in Python: **Absolute path:** An absolute path is the full path of a file or directory, starting from the root directory of the file system. In Unix-based systems, the root directory is typically denoted by a forward slash (/), while in Windows systems, it is denoted by a backslash (). For example, "/home/user/myfile.txt" is an absolute path on a Unix system, while "C:\Users\User\Documents\myfile.txt" is an absolute path on a Windows system. **Relative path:** A relative path is a path that is relative to the current working directory of the program. For example, "mydir/myfile.txt" is a relative path that specifies a file called "myfile.txt" located in a directory called "mydir" that is located in the current working directory. **Parent path:** A parent path is a path that points to the parent directory of a file or	CO3 – L2	6

		directory. This can be specified using the ".." directory name. For example, if the current working directory is "/home/user/mydir", then the parent directory can be referred to as "../".		
	b	Explain how to read specific lines from a file?. illustrate with python Program To read specific lines from a file in Python, you can use a combination of the **open()** function and the **readlines()** method. The **open()** function is used to open the file in read mode, while the **readlines()** method is used to read all the lines in the file and return them as a list of strings. Once you have the list of lines, you can access specific lines by using list indexing. Here's an example Python program that demonstrates how to read specific lines from a file: `# Open the file in read mode` `with open("example.txt", "r") as f:` `  # Read all the lines in the file and store them in a list` `  lines = f.readlines()` `  # Print the first line of the file` `  print("First line:", lines[0])` `  # Print the third line of the file` `  print("Third line:", lines[2])` In this program, we open a file called "example.txt" in read mode using the **open()** function. We then use the **readlines()** method to read all the lines in the file and store them in a list called **lines**.	CO5 – L2	6
	c	What is logging? how this would be used to debug the python program? Logging is a technique used in programming to record and report events that occur during the execution of a program. It provides a way to track the behavior of a program and helps in identifying issues and debugging problems. The **logging** module in Python provides a flexible and efficient way to handle logging in Python programs. Let's understand the following example. **Example -** `import logging` `logging.debug('The debug message is displaying')` `logging.info('The info message is displaying')` `logging.warning('The warning message is displaying')` `logging.error('The error message is displaying')` `logging.critical('The critical message is displaying')` **Output:** `WARNING:root:The warning message is displaying` `ERROR:root:The error message is displaying` `CRITICAL:root:The critical message is displaying`	CO3 – L3	8

		OR		
Q. 06	a	What is the use of ZIP? how to create a ZIP folder explain. Compressed ZIP files reduce the size of the original directory by applying compression algorithm. Compressed ZIP files result in faster file sharing over a network as the size of the ZIP file is significantly smaller than original file. **Example** **Following is an example to create ZIP file using multiple files –**	CO3 – L2	6
		import os from zipfile import ZipFile **# Create a ZipFile Object** with ZipFile('/home/secabiet/test.zip', 'w') as zip_object: **# Adding files that need to be zipped** zip_object.write('/home/secabiet/test/test1') zip_object.write('/home/secabiet/test/a.py') zip_object.write('/home/secabiet/test/b.py') **# Check to see if the zip file is created** if os.path.exists('/home/secabiet/test.zip'): print("ZIP file created") else: print("ZIP file not created")		
		OUTPUT **ZIP file created**		

	b	wite an algorithm for implement multi clipboard functionality Here is an algorithm for implementing multi clipboard functionality: 1. Create an empty dictionary to hold the clipboard data. 2. Define a function named `addToClipboard` that takes two arguments - a `name` for the clipboard and the `data` to be stored in the clipboard. 3. Inside the function `addToClipboard`, add the `name` and `data` to the clipboard dictionary as a key-value pair. 4. Define another function named `getFromClipboard` that takes one argument - the `name` of the clipboard to retrieve data from. 5. Inside the function `getFromClipboard`, retrieve the `data` associated with the `name` key from the clipboard dictionary. 6. Return the `data` retrieved from the clipboard. 7. Finally, implement a user interface that allows the user to interact with the clipboard functions. This can include commands to add data to the clipboard, retrieve data from the clipboard, and list all available clipboards.	CO5 – L2	6
	c	Discuss how lists would be written in the file and read from the file? **Here's an example of how to write a list to a file:** # Define a list to write to the file mylist = [1, 2, 3, 4, 5] # Open the file for writing with open('my_list.txt', 'w') as f: # Convert the list to a string and write it to the file f.write(str(mylist)) To read the list from the file, we can use the **ast.literal_eval()** function from the **ast** module to safely evaluate the string as a Python expression: import ast # Open the file for reading with open('my_list.txt', 'r') as f: # Read the string from the file str_list = f.read() # Convert the string to a list using ast.literal_eval() mylist = ast.literal_eval(str_list) # Print the list to verify that it was read correctly print(mylist)	CO3 – L3	8
		Module-4		
Q. 07	a	Define the terms with example: (i) class (ii) objects (iii) instance variables (i) Class: A class in Python is a blueprint for creating objects. It is a code template that defines the attributes and methods that an object can have. In other words, a class is a collection of related data and functions that operate on that data. (ii) Objects: An object is an instance of a class. It is created from the blueprint provided by the class, and has its own unique identity, state, and behavior. In other words, an object is a specific instance of a class that has its own values for the attributes defined by the class.	CO3 – L1	6

		(iii) Instance variables: Instance variables are attributes that are specific to each instance of a class. They are created when an object is instantiated, and can have different values for different instances of the same class. Instance variables are accessed using the **self** keyword within the class methods.		
	b	create a Time class with hour, min and sec as attributes. Demonstrate how two Time objects would be added.	CO3 – L3	8
		class Time: hour=0 minute=0 second=0 def print_time(t): print('%.2d:%.2d:%.2d' % (t.hour, t.minute, t.second)) def add_time(t1, t2): sum=Time() sum.hour = t1.hour + t2.hour sum.minute = t1.minute + t2.minute sum.second = t1.second + t2.second return sum t1 = Time() t1.hour = 9 t1.minute = 45 t1.second = 0 t2 = Time() t2.hour = 1 t2.minute = 35 t2.second = 0 t3 = Time() t3=Time.add_time(t1,t2) Time.print_time(t3)		
		OUTPUT: **10:80:00**		
	c	Discuss __str__() and __init__() methods used in class definition. The init () method is used to initialize the object's attributes class Person: def init (self, name, age): self.name = name self.age = age The__str__() method is used to provide a string representation of the object. class Person: def __init__(self, name, age): self.name = name self.age = age def __str__(self): return f"Name: {self.name}, Age: {self.age}" p = Person("John", 25) print(p) # Output: Name: John, Age: 25	CO3 – L2	6
		OR		
Q. 08	a	What is Encapsulation? Discuss with an example in which access specifiers are used in class definition.	CO3 – L2	4

Encapsulation is a principle of object-oriented programming that involves bundling data and methods that operate on that data within a single unit, such as a class.

Encapsulation provides several benefits, including:

- Improved code maintainability
- Better code organization
- Reduced code complexity

```
class Car:
  def __init__(self, make, model, year):
    self._make = make
    self._model = model
    self._year = year

  def _start_engine(self):
    print("Engine started.")

  def drive(self):
    self._start_engine()
   print("Driving...")
```

In this example, the attributes **make**, **model**, and **year** and the method **_start_engine** are intended to be private, as indicated by the leading underscore in their names.

These members are still accessible from outside the class, but it is a convention that they should not be accessed directly.

Instead, we should use the public methods of the class, such as the **drive** method, which can call the private **_start_engine** method as needed.

	b	What is a class diagram? Create empty class and corresponding class diagram for following statements (i) class A derives from class B and Class C (ii) Class D derived from Class A defined in statement (i)	CO3 – L3	8

Using this syntax, we can create the following empty classes and corresponding class diagram for the given statements:

(i) Class A derives from class B and Class C

```
class B:
   pass

class C:
   pass

class A(B, C):
   pass
```

OUTPUT:

```
    B    C
     \  /
      \/
       A
```

		(ii) Class D derived from Class A defined in statement (i) class D(A): pass **OUTPUT:** **B C** \ / \ / **A** | **D**		
	c	discuss polymorphism and demonstrate with and python program. The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types. The key difference is the data types and number of arguments used in function. **Example 1:** # len() being used for a string print(len("SECAB")) # len() being used for a list print(len([10, 20, 30])) OUTPUT: 5 3 **Example 2:** def add(x, y, z = 0): return x + y+z # Driver code print(add(2, 3)) print(add(2, 3, 4)) OUTPUT: 5 9	CO3 – L2	8
		Module-5		
Q. 09	a	write python program to read cell **2C** from sheet 2 of workbook	CO4 – L2	6
	b	explain how pdf pages would created in a pdf document with example.	CO4 – L2	6
	c	What is JSON? discuss with example. Compare it with dictionary	CO4 – L3	8
		OR		
Q. 10	a	compare and contrast Excel and CSV files.	CO4 – L2	6
	b	Demonstrate how a Class would be converted into JSON object with an example.	CO4 – L3	8
	c	Explain how a page from different PDFs files would be merged into a new PDF file?	CO4 – L3	6