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

Internal Assessment Test 1 – MAY 2025

Sub:	INTRODUCTION TO PYTHON PROGRAMMING				Sub Code:	BPLCK205B	Branch:	ISE
Date:		Duration:	90 min's	Max Marks: 50	Sem/Sec:	II ALL SECTIONS		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1a)	Explain the use of following functions/methods with code snippets i) input () ii) type () iii) format () iv) remove ()	[4]	CO1	L2
1b)	Apply your understanding and write a python program to print following pattern: * * * * * * * * * * * * * * *	[6]	CO1	L2
2	Write Python Program to implement Menu Driven Calculator using functions	[10]	CO1	L3
3 a)	Explain the concept of Local scope and global scope of the variables in Python with example code snippets.	[6]	CO1	L2
3 b)	How exceptions can be handled in Python? Explain with an example.	[4]	CO1	L2
4 a)	Differentiate between the following: a. del () and remove() methods of list. b. append() and insert() methods of list.	[4]	CO2	L3
4 b)	Make a list of the first eight letters of the alphabet, then using the slice operation do the following operations: a. Print the first three letters of the alphabet. b. Print any three letters from the middle. c. Print the letters from any particular index to the end of the list.	[6]	CO2	L3
5 a)	Explain with examples the way how indexing and slicing can be done in Lists	[5]	CO2	L2
5 b)	Identify and explain the dictionary methods like get(), item(), keys() and values() in python with examples.	[5]	CO2	L2
6 a)	Explain dictionary in Python? Explain get() and set default () method with example	[5]	CO2	L2
6 b)	Differentiate between lists and dictionary. Explain with an example.	[5]	CO2	L2

Internal Assessment Test 1 – MAY 2025

Sub:	INTRODUCTION TO PYTHON PROGRAMMING					Sub Code:	BPLCK205B	Branch:	ISE
Date:		Duration:	90 min's	Max Marks:	50	Sem/Sec:	II ALL SECTIONS		OBE

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1a)	Explain the use of following functions/methods with code snippets i) input() ii) type() iii) format() iv) remove() 4X1M=4M	[4]	CO1	L2

Solution:

i) input()

Takes user input from the keyboard as a string.

Syntax:

```
input("Your message here")
```

Example:

```
name = input("Enter your name: ")
print("Hello", name)
```

Output:

```
Enter your name: Alice
Hello Alice
```

ii) type()

Returns the data type of a given variable or value.

Example:

```
x = 10
print(type(x))     # Output: <class 'int'>

y = "Python"
print(type(y))     # Output: <class 'str'>
```

iii) format()

Used to format strings by inserting values into placeholders {}.

Syntax:

	"{}".format(value) Example: name = "Alice" age = 25 print("My name is {} and I am {} years old.".format(name, age)) Output: My name is Alice and I am 25 years old. **iv) remove()** Removes the first occurrence of a specified element from a list. Example: fruits = ["apple", "banana", "cherry", "banana"] fruits.remove("banana") print(fruits) Output: ['apple', 'cherry', 'banana']			

1b)	Write a python program to print following pattern:
	*
	* *
	* * *
	* * * *
	* * * * *
	
	rows = 5
	for i in range(1, rows + 1):
	for j in range(i):
	print("*", end=" ")
	print()
		[6]	CO1	L2

2	Write Python Program to implement Menu Driven Calculator using functions	[10]	CO1	L3

8M PROGRAM
2M OUTPUT

Solution:

```
while True:
    print("\n--- Calculator ---")
    print("1. Addition \n2. Subtraction \n3. Multiplication \n4. Division")

    operator = int(input("Enter your choice (1-4): "))
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if operator == 1:
        print("Result =", num1 + num2)

    elif operator == 2:
        print("Result =", num1 - num2)

    elif operator == 3:
        print("Result =", num1 * num2)

    elif operator == 4:
            print("Result =", num1 / num2)
    else:
        print("Invalid number! Please enter a number from 1 to 5.")

choice=(input("Do you want to perform another calculation? (yes/no): "))
if choice=="yes":
    continue
else:
    break
```

3 a)	Explain the concept of Local scope and global scope of the variables in Python with example code snippets. EXPLANATION LOCAL: 1M	[6]	CO1	L2

	EXPLANATION GLOBAL: 1M
	EXAMPLE LOCAL: 2M
	EXAMPLE GLOBAL: 2M
	Solution:
	Scope of a variable refers to the region where a variable is recognized and can be accessed.
	Types of Scope:
	Local Scope: Variable declared inside a function, accessible only within that function.It is destroyed once the function ends
	Example:
	def greet():
	message = "Hello, World!" # local variable
	print(message)
	greet()
	print(message) # This will give an error because 'message' is local
	Global Scope: Variable declared outside any function, accessible throughout the program. It exists until program ends
	Example:
	greeting = "Hello, Universe!" # global variable
	def greet():
	print(greeting)
	greet()
	print(greeting) # Works because 'greeting' is global			

3 b)	How exceptions can be handled in Python? Explain with an example. EXPLANATION: 2M EXAMPLE: 2M **Solution:**	[4]	CO1	L2

In Python, exceptions are handled using try, except, and finally blocks.

It helps prevent program crash and allows graceful error handling.

```
try:
    a = int(input("Enter a number: "))
    b = int(input("Enter another number: "))
    result = a / b
    print("Result:", result)
except ZeroDivisionError:
    print("Error! Cannot divide by zero.")
except ValueError:
    print("Error! Please enter a valid number.")
finally:
    print("This block always runs.")
```

4 a) Differentiate between the following:
a. del () and remove() methods of list.
b. append() and insert() methods of list.

[4] CO1 L3

DIFFERENTIATE 5M

Solution

Feature	del()	remove()
Purpose	Deletes an element at a specific index or deletes the entire list.	Removes the first occurrence of a specific value from the list.
Parameter	Takes an **index** (or a slice) to delete.	Takes a **value** to remove.
Error if not found	If wrong index is given, throws IndexError.	If the value is not found, throws ValueError.
Syntax	del list[index] or del list	list.remove(value)
Example	del mylist[2] (removes element at index 2)	mylist.remove(5) (removes first occurrence of value 5)

	Feature	**append()**	**insert()**		
	Purpose	Adds an element at the **end** of the list.	Inserts an element at a **specific index** in the list.		
	Parameter	Takes a **single value** to be added.	Takes two parameters: **index** and **value**.		
	Modification	Increases list size by 1 by adding to the end.	Increases list size by 1 by inserting at the specified position.		
	Syntax	list.append(value)	list.insert(index, value)		
	Example	mylist.append(10) (adds 10 at the end)	mylist.insert(2, 10) (inserts 10 at index 2)		

4 b) Make a list of the first eight letters of the alphabet, then using the slice operation do the following operations:
a. Print the first three letters of the alphabet.
b. Print any three letters from the middle.
c. Print the letters from any particular index to the end of the list.

[6] CO2 L3

3X2M=6M

Solution:

alphabet = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H']

Now, performing the required slice operations:

a. Print the first three letters of the alphabet:

print(alphabet[0:3])

Output:

['A', 'B', 'C']

b. Print any three letters from the middle:

Example: Let's print 4th, 5th, and 6th letters (D, E, F):

print(alphabet[3:6])

Output:

['D', 'E', 'F']

c. Print the letters from any particular index to the end of the list:

Example: Starting from index 5 (letter 'F') to the end:

print(alphabet[5:])

Output:

['F', 'G', 'H']

5 a) Explain with examples the way how indexing and slicing can be done in Lists [5] CO2 L2

EXPLANATION AND EXAMPLES: 5M

Solution:

=> **Indexing in Lists**

- **Indexing** means accessing individual elements in a list using their position (index).
- In Python, **indexing starts from 0** (first element is at index 0, second at 1, and so on).
- You can also use **negative indexing** where -1 refers to the last element, -2 to second-last, and so on.

Example:

my_list = ['A', 'B', 'C', 'D', 'E']

Element	**A**	**B**	**C**	**D**	**E**
Positive Index	0	1	2	3	4
Negative Index	-5	-4	-3	-2	-1

- Access first element

```
print(my_list[0])
```

Output: A

- Access last element using negative indexing

```
print(my_list[-1])
```

Output: E

Slicing in Lists

- **Slicing** means taking a part (sublist) of the list.
- Syntax: list[start:stop]
  - start → starting index (inclusive)
  - stop → ending index (exclusive)
- If start is omitted, slice starts from the beginning.
- If stop is omitted, slice goes till the end.

Example:

my_list = ['A', 'B', 'C', 'D', 'E']

- Get first three elements:

```
print(my_list[0:3])
```

Output: ['A', 'B', 'C']

- Get elements from index 2 to end:

```
print(my_list[2:])
```

Output: ['C', 'D', 'E']

- Get elements from start to index 3 (excluding 3):

```
print(my_list[:3])
```

Output: ['A', 'B', 'C']

- Get last two elements using negative slicing:

```
print(my_list[-2:])
```

Output: ['D', 'E']

5 b)	Identify and explain the dictionary methods like get(), item(), keys() and values() in python with examples. EXPLANATION AND EXAMPLES: 5M **Solution:**	[5]	CO2	L2

1. `get()`

- Purpose

Returns the value for a specified key if the key exists in the dictionary.

If the key does not exist, it returns a default value instead of raising an error.

-syntax

Dictionary.get(key, default_value)

Example

```
My_dict = {'name': 'Alice', 'age': 25}

Print(my_dict.get('name'))        # Output: Alice
Print(my_dict.get('gender', 'N/A')) # Output: N/A (because 'gender' key doesn't exist)
```

2.Items():

-Purpose

Returns a view object that displays a list of dictionary's (key, value) pairs.

Syntax

Dictionary.items()

Example

```
My_dict = {'name': 'Alice', 'age': 25}

Print(my_dict.items())
```

Output:

```
Dict_items([('name', 'Alice'), ('age', 25)])
```

	3. `keys()` Purpose Returns a **view object** containing **all the keys** in the dictionary. Syntax Dictionary.keys() Example My_dict = {'name': 'Alice', 'age': 25} Print(my_dict.keys())			

```

**Output:**

```

Dict_keys(['name', 'age'])

```

4. `values()`:

Purpose

Returns a view object containing all the values in the dictionary.

Syntax

Dictionary.values()

Example

My_dict = {'name': 'Alice', 'age': 25}

Print(my_dict.values())

```

Output:

```

Dict_values(['Alice', 25])

| | | | | | |
|---|---|---|---|---|---|
| 6 a | Explain dictionary in Python? Explain get() and set default () method with example<br>DICTIONARY EXPLANATION: 1M<br>get()-2M<br>setdefault()-2M<br>Solution:<br>• A **dictionary** in Python is a **collection of key-value pairs**.<br>• Each **key** is unique, and it is associated with a **value**. | | 5 | CO2 | L2 |
```

- Dictionaries are **unordered** (in older versions of Python <3.7) and **mutable** (can be changed after creation).
- Defined using **curly braces {}**.

Example:

```
my_dict = {
  'name': 'Alice',
  'age': 25,
  'city': 'Bangalore'
}
```

- 'name', 'age', 'city' are **keys**.
- 'Alice', 25, 'Bangalore' are their corresponding **values**.

get() Method:

- The get() method is used to **retrieve the value** for a given key.
- If the key **exists**, it returns the value.
- If the key **does not exist**, it returns **None** by default, or you can specify a default return value.

Syntax: dictionary.get(key, default_value)

Example:

```
person = {'name': 'John', 'age': 30}
# Key exists
print(person.get('name')) #
Output: John
# Key does not exist
print(person.get('city'))
Output: None
# Key does not exist, but with default value
print(person.get('city', 'Unknown'))
Output: Unknown
```

setdefault() Method:

- The setdefault() method **returns the value** of a key if it is present.

	• If the key **does not exist**, it **adds the key with a specified default value**.
	• Useful to **insert a new key-value pair safely** without overwriting existing ones.
	Syntax: dictionary.setdefault(key, default_value)
	Example:
	person = {'name': 'John', 'age': 30}
	# Key exists
	print(person.setdefault('name', 'Unknown'))
	Output: John
	# Key does not exist, so it will add 'city' with value 'Bangalore'
	print(person.setdefault('city', 'Bangalore'))
	Output: Bangalore
	# Now person dictionary looks like:
	print(person)
	Output: {'name': 'John', 'age': 30, 'city': 'Bangalore'}				

6 b	Differentiate between lists and dictionary. Explain with an example. DIFFERENTIATE: 5M Solution:		5	CO2	L2

Feature	List	Dictionary
Definition	An **ordered** collection of elements.	An **unordered** collection of **key-value** pairs.
Indexing	Elements are accessed by **position/index** (starting from 0).	Elements are accessed by **keys**.
Syntax	Defined using **square brackets** `[ ]`.	Defined using **curly braces** `{ }`.
Order	Maintains **insertion order** (since Python 3.7).	Also maintains **insertion order** (since Python 3.7), but primarily key-based.
Use case	When you need an ordered collection of items.	When you need to associate **keys** with **values** (like a database).
Example	`list1 = [10, 20, 30]`	`dict1 = {'a': 10, 'b': 20, 'c': 30}`
Mutable	Yes (elements can be changed).	Yes (key-value pairs can be changed).

HoD Signature CCI signature Course Instructor