Shiv Nadar Institute of Eminence
End Term Examination
Monsoon 2024

COURSE CODE: CCC634 **MAX. DURATION: 1.5 hr**

COURSE NAME: A Gentle Introduction to Python

COURSE CREDIT: 1.5 **MAX. MARKS: 70**

Roll No: _____ Name of Student: _____

Department/ School: _____

INSTRUCTIONS: -

- Do not write anything on the question paper except name, enrolment number and department/school.
- Carrying mobile phones, smartwatches and any other non-permissible materials in the examination hall is an act of UFM.
- All Questions are mandatory.
- Draw clear Diagrams, wherever it is required.
- Read the question carefully before attempting.

SECTION A (Max Marks = 30 Marks)

1. What is the difference between explicit and implicit data type conversion? Show with an example. (**2 marks**)

   Implicit Conversion: This occurs automatically when Python converts one data type to another without the programmer's intervention. For example, when you add an integer to a float:

```
num_int = 5
num_float = 3.2
result = num_int + num_float  # 'num_int' is implicitly converted
to float
print(result)  # Output: 8.2
```

   Explicit Conversion: This occurs when the programmer explicitly converts a data type into another using built-in functions.

```
num_str = "10"
num_int = int(num_str)  # 'num_str' is explicitly converted to an
integer
print(num_int + 5)  # Output: 15
```

2. Name any four types of operators in Python, each with an example. (**2 marks**)

   Arithmetic Operators (**, *,/,//,%,+,_): e.g. x = 5 + 3
   Comparison Operators (<, <=, >, >=, ==): if (x > 10)
   Assignment Operators (=, +=, -=, *=, /=): x+=5
   Logical Operators (and, or, not): if (x>10) and (x%3==0)
   Bitwise Operators (&, |, ~, >>, <<, ^): y = x>>2
   Membership Operators (in, not in): x in my_list

3. What are the other two control structures in Python apart from normal "Sequential Control". Give an example of each of them. (**2 marks**)

   Selection control: Selectively executes the instructions. e.g. If Else

```
x = 10
if x > 5:
        print("Greater than 5")
else:
        print("Less than or equal to 5")
```

   Iterative control: Repeatedly executes the instructions. e.g. Loops.

```
count = 0
while count < 5:
        print(count)
        count += 1
```

4. Write a code snippet which defines a list and iterates over its elements? (**2 marks**)

```
my_list = [1, 2, 3, 4, 5]
for element in my_list:
        print(element)
```

5. What are nested dictionaries? Define and illustrate with an example. Show how you would access the data within one of the child dictionaries. (**3 marks**)

   A **nested dictionary** is a dictionary that has another dictionary as a value.

```
nested_dict = {
        "person1": {"name": "Alice", "age": 30},
        "person2": {"name": "Bob", "age": 25}
        }

# Accessing data within one of the child dictionaries
print(nested_dict["person1"]["name"])  # Output: Alice
```

6. Describe the usage of the split() and strip() functions in Python? What are the function input(s) and output(s). (**4 marks**).

   `split()`: This function is used to split a string into a list at specified separator(s).

   Input: `"Hello World".split()` **Output**: `['Hello', 'World']`

   `strip()`: This function removes leading and trailing whitespace from a string.

   Input: `" Hello World ".strip()` **Output**: `"Hello World"`

7. What are lambda functions? Illustrate with an example how you declare it and use it within your main program. (**4 marks**)

   Lambda functions are anonymous functions defined using the `lambda` keyword. They can take any number of arguments but can only have one expression.

```
# Declaration and usage
add = lambda x, y: x + y
result = add(5, 3)
print(result)  # Output: 8
```

8. Show via an example, how "method overriding" works in python? (**4 marks**)

   Method overriding occurs when a derived class has a method that overrides a method with the same name in its base class.

```
class Animal:
        def sound(self):
                return "Some sound"

class Dog(Animal):
        def sound(self):
                return "Bark"

# Demonstration
animal = Animal()
dog = Dog()

print(animal.sound())  # Output: Some sound
print(dog.sound())     # Output: Bark
```

9. Find the output of the following programs [in case of an error, mention the error, fix the code and write output after fixing]. (**5 marks**)

(a)

```
for i in range(4, -1, -1):
    print("^"*i)
```

```
Output:
^^^^
^^^
^^
^
```

(b)

```
L1 = [4, 6, 7, 9]
T1 = tuple(L1)
print(T1)
L1[2] = 9
print(T1)
```

```
print(T1)   # Output: (4, 6, 7, 9)
L1[2] = 9   # This is a valid operation
print(T1)   # Output: [4, 6, 9, 9]
```

(c)

```
my_dict = {
            "name": "Shiv",
            "university": "SNIoE",
            "year": 2023
          }
for x, y in my_dict.items():
    print(x, y)
```

```
Output:
name Shiv
university SNIoE
year 2023
```

(d)

```
s = "hello"
for char in s:
    new_str = char + new_str
```

print(new_str)

Error: `new_str` is not defined before concatenation in the loop.

Fixed Code:

```
s = "hello"
new_str = ""  # Initialize new_str
for char in s:
        new_str = char + new_str
        print(new_str)  # Output: olleh
```

(e)

```
def func(a, b, c = 'm'):
        out = b*c
        return a, out
x, y = func(3, 2)
print(x, y)
```

Output: 3 mm

SECTION B (Max Marks = 40 Marks)

Answer any two questions

1. Write a python program which takes student details as input and store all the information in a nested dictionary. Ask the no. of students as an input from the user. For 'n' no. of students store details of 'name', 'age', 'year', and 'GPA' as a nested dictionary. Your program should allow to 'add'[1] a new student, 'update' [2] the details of an existing student, and 'show' [3] details of all students. Ask the user options to either perform [1], [2], [3], or 'None' and only stop the code execution when the option is 'None'. (**20 marks**)

```
def add_student(students):

    """Function to add a new student."""
    student_id = input("Enter student ID for new student: ")

    if student_id in students:
        print("Student ID already exists. Please use a unique ID.")
        return

    name = input("Enter name: ")
    age = int(input("Enter age: "))
    year = input("Enter year: ")
    gpa = float(input("Enter GPA: "))

    students[student_id] = {
```

```
        'name': name,
        'age': age,
        'year': year,
        'GPA': gpa
    }
    print(f"Student ID {student_id} added successfully.")


def update_student(students):
    """Function to update existing student details."""
    student_id = input("Enter student ID to update: ")
    if student_id not in students:
        print("Student ID not found.")
        return

    # Prompt for new details, keeping existing ones if left blank
    name = input("Enter new name (leave blank to keep current): ")
    age_input = input("Enter new age (leave blank to keep current): ")
    year = input("Enter new year (leave blank to keep current): ")
    gpa_input = input("Enter new GPA (leave blank to keep current): ")

    if name:
        students[student_id]['name'] = name
    if age_input:
        students[student_id]['age'] = int(age_input)
    if year:
        students[student_id]['year'] = year
    if gpa_input:
        students[student_id]['GPA'] = float(gpa_input)

    print(f"Student ID {student_id} updated successfully.")


def show_students(students):
    """Function to display all student details."""
    if not students:
        print("No student details available.")
        return

    for student_id, details in students.items():
        print(f"Student ID: {student_id}, Details: {details}")



"""Main program to manage student records."""
students = {}

n = int(input("Enter the number of students: "))
for x in range(n):
    add_student(students)

while True:
    print("\nOptions:")
    print("1: Add a new student")
    print("2: Update existing student details")
    print("3: Show all student details")
    print("Type 'None' to exit")

    option = input("Choose an option: ")

    if option == '1':
```

```
            add_student(students)
        elif option == '2':
            update_student(students)
        elif option == '3':
            show_students(students)
        elif option.lower() == 'none':
            print("Exiting the program.")
            break
        else:
            print("Invalid option. Please try again.")
```

2. Write python code snippets to perform the below tasks both iteratively (using for/while loops) **and** via recursion (no usage of loops). You should have one single main program for one task and define two separate functions within that program one for 'iterative' and one for 'recursive' approach. (**20 marks**)
    - [A] Find sum of digits of a given number. Ask the number as user input. Call the functions in the main program and print the 'sum' as output.
    - [B] Generate a Fibonacci sequence up to 'n' terms. Ask the 'n' as input. Call the functions in the main program and store the sequence terms in a list and print that list as output.

[A]

```
def sum_of_digits_iterative(n):
    """Calculate sum of digits of a number iteratively."""
    total = 0
    while n > 0:
        total += n % 10  # Add the last digit to total
        n //= 10  # Remove the last digit
    return total

def sum_of_digits_recursive(n):
    """Calculate sum of digits of a number recursively."""
    if n == 0:
        return 0
    else:
        return (n % 10) + sum_of_digits_recursive(n // 10)

"""Main function to manage the sum of digits task."""
number = int(input("Enter a number to find the sum of its digits: "))

iterative_sum = sum_of_digits_iterative(number)
recursive_sum = sum_of_digits_recursive(number)

print(f"Iterative sum of digits: {iterative_sum}")
print(f"Recursive sum of digits: {recursive_sum}")
```

[B]

```
def fibonacci_iterative(n):
    """Generate Fibonacci sequence iteratively up to n terms."""
    fib_sequence = []
    a, b = 0, 1
    for x in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence

def fibonacci_recursive(n, a=0, b=1, fib_sequence=None):
    """Generate Fibonacci sequence recursively up to n terms."""
    if fib_sequence is None:
        fib_sequence = []
    if n <= 0:
        return fib_sequence
    fib_sequence.append(a)
    return fibonacci_recursive(n - 1, b, a + b, fib_sequence)

n = int(input("Enter the number of terms for the Fibonacci sequence: "))

iterative_fib_sequence = fibonacci_iterative(n)
recursive_fib_sequence = fibonacci_recursive(n)

print(f"Iterative Fibonacci sequence: {iterative_fib_sequence}")
print(f"Recursive Fibonacci sequence: {recursive_fib_sequence}")
```

3. Create a python class called Rectangle. It inherits some properties from a parent class Polygon and has the attributes of ‘length’, ‘width’, ‘area’, and ‘perimeter’. It should have an “__init__” method for initialization. Implement methods to calculate the area and perimeter of the rectangle. Also include methods to change the length or width. Include appropriate error handling (e.g., for negative dimensions). Further, the class should have a class attribute which stores the no. of rectangles created so far. In the main function create two instances of the above class and print the no. of objects using the object-count variable in the class definition. (**20 marks**)

```
class Polygon:
    """Base class for Polygon."""
    pass  # You can add more properties or methods for Polygon if needed.


class Rectangle(Polygon):
    """Class representing a rectangle."""

    # Class attribute to count the number of Rectangle instances
    count = 0

    def __init__(self, length, width):
        """Initialize the rectangle with length and width."""
        self.set_length(length)
        self.set_width(width)
        Rectangle.count += 1  # Increment count for each new instance
```

```
    def set_length(self, length):
        """Set the length of the rectangle with validation."""
        if length < 0:
            print("Length cannot be negative.")
            return
        self.length = length

    def set_width(self, width):
        """Set the width of the rectangle with validation."""
        if width < 0:
            print("Width cannot be negative.")
            return
        self.width = width

    def calculate_area(self):
        """Calculate and return the area of the rectangle."""
        return self.length * self.width

    def calculate_perimeter(self):
        """Calculate and return the perimeter of the rectangle."""
        return 2 * (self.length + self.width)


"Main Program"
rect1 = Rectangle(5, 3)
rect2 = Rectangle(4, 6)


print(f"Area of Rectangle 1: {rect1.calculate_area()}")  # Output: 15
print(f"Perimeter of Rectangle 1: {rect1.calculate_perimeter()}")  #
Output: 16

print(f"\nArea of Rectangle 2: {rect2.calculate_area()}")  # Output: 24
print(f"Perimeter of Rectangle 2: {rect2.calculate_perimeter()}")  #
Output: 20
print(f"\nNumber of Rectangle instances created: {Rectangle.count}")
#Output: 2
```