Shiv Nadar Institute of Eminence
Mid 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: 50**

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 = 20 Marks)

1. Explain the difference between mutable and immutable data types in Python with examples. (**2 marks**)

   Mutable Data Types: These are data types whose values can be changed after they are created. Modifying a mutable object doesn't create a new object; it modifies the existing one in memory.

   ```
   my_list = [1, 2, 3]
   my_list.append(4)  # Modifies the original list
   print(my_list)  # Output: [1, 2, 3, 4]
   ```

   Immutable Data Types: These are data types whose values cannot be changed after they are created. If you try to modify an immutable object, Python creates a new object in memory with the new value. The original object remains unchanged.

   ```
   my_tuple = (1, 2, 3)
   ```

```
#my_tuple[0] = 4 #This will give an error, you cannot change a
tuple, its value is fixed
new_tuple = my_tuple + (4,)  # Creates a new tuple
print(my_tuple)  # Output: (1, 2, 3)
print(new_tuple) #Output: (1, 2, 3, 4)
```

2. What are different types of literals in python? Mention any four types with an example for each. (**2 marks**)

   Numeric Literals: Represent integer and floating point values
   ```
   age = 30
   count = -10
   cost = 10.5
   ```

   String Literals: Represent sequences of characters enclosed in single quotes ('...') or double quotes ("...")
   ```
   name = "Alice"
   message = 'Hello, world!'
   ```

   Boolean Literals: Represent truth values (True or False)
   ```
   is_valid = True
   is_finished = False
   ```

   Collection Literals: Stores collection of literal values
   ```
   my_list = [1, 2, 3]
   name_list = [‘Sam’, ‘Karen’, ‘Manoj’]
   ```

3. In the below piece of code what happens to the value ‘5’ in the memory. How is that handled in Python? (**2 marks**)

   >>> x = 5
   >>> print(x)
   5
   >>> x = 6.5
   >>> print(x)
   6.5

   Garbage Collection: If the integer object '5' is no longer referenced by any variable, Python's garbage collector will eventually reclaim the memory used by that object. The garbage collector automatically identifies, and releases memory occupied by objects that are no longer in use. This helps prevent memory leaks.

4. If `x = 20`, `y = 14`, `z = 0`, evaluate the following expressions: (**4 marks**)

   i. `x ^ y` ii. `x | y` iii. `y & x` iv. `x >> 2` [Note: Show intermediate computations for each of these operations]

   Given `x = 20`, `y = 14`, `z = 0`:

   i. `x ^ y` (Bitwise XOR)
   * `x` (20) in binary: `00010100`

* `y` (14) in binary: `00001110`
* `x ^ y`: `00011010` (26 in decimal)
* Result: 26

ii. `x | y` (Bitwise OR)
* `x` (20) in binary: `00010100`
* `y` (14) in binary: `00001110`
* `x | y`: `00011110` (30 in decimal)
* Result: 30

iii. `y & x` (Bitwise AND)
* `y` (14) in binary: `00001110`
* `x` (20) in binary: `00010100`
* `y & x`: `00000100` (4 in decimal)
* Result: 4

iv. `x >> 2` (Right Bit Shift)
* `x` (20) in binary: `00010100`
* `x >> 2`: `00000101` (5 in decimal)
* Result: 5

5. Predict the output of the following Python statements: (**3 marks**)

i. `num = input("Enter a value: ") print(type(num))`

Output: `<class 'str'>` | **Reasoning**: The `input()` function always returns a string, regardless of what the user enters.

ii. `print(15//4)`

Output: 3 | **Reasoning**: // is the floor division operator. 15 divided by 4 is 3.75, and the floor (the largest integer less than or equal to 3.75) is 3.

iii. `print(5%-2)`

Output: -1 | **Reasoning**: The modulus operator (%) returns the remainder of a division. The result will have the sign of the divisor (-2). 5 divided by -2 is -2 with a remainder of -1. Because 5 = (-2) * (-2) - 1

iv. `print(-8//3)`

Output: -3 | **Reasoning**: Floor division. -8 divided by 3 is -2.666..., and the floor of -2.666... is -3. Because -8 = 3*(-3) + 1

v. `print(19/3%6**2)`

Output: `6.333` (approximately) | **Reasoning**:

1. `6**2` (exponentiation) = 36

2. `19/3` (division) = 6.333...
3. `6.333... % 36` (modulus) = 6.333... Since 6.333 is less than 36 the remainder is just 6.333

6. Python has many built-in functions. Illustrate the working of the following functions with examples: (**2 marks**)

   a. sqrt(x)

```
import math  # Important: Need to import the math module
x = 25
result = math.sqrt(x)
print(result)  # Output: 5.0
```

   b. round(x, n)

```
x = 3.14159
result = round(x, 2)  # Rounds to 2 decimal places
print(result)  # Output: 3.14
```

   c. pow(x, y)

```
x = 2
y = 3
result = pow(x, y)  # x raised to the power of y
print(result)  #Output: 8.0(In Python 3, the return is
float)
```

   d. bin(x)

```
x = 10
result = bin(x)
print(result) #Output: 0b1010 (binary representation of 10)
```

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

   (a)

```
for i in range(4):
  for j in range(i):
    print(i, j)
```

Output:

1 0
2 0
2 1
3 0
3 1
3 2

   (b)

```
count = 0
while count < 3:
    print(count)
    count = count + 1
```

```
else:
    print("Loop finished")
print("Outside loop")
```

Output:

```
0
1
2
Loop finished
Outside loop
```

(c)

```
a = 10
b = 20
if a > b:
        print("a is greater")
elif a < b:
        print("b is greater")
print("both are equal")
```

Logical Error: Missing “else” block
Fixed Version:

```
a = 10
b = 20
if a > b:
        print("a is greater")
elif a < b:
        print("b is greater")
else:
        print("both are equal")
```

Output:
b is greater

(d)

```
nums = [10, 20, 30]
for i in range(len(nums)):
        nums[i] += 5
print(nums)
```

Output: [15, 25, 35]

(e)

```
for i in range(3, -1, -1):
        print(10 / i)
```

Runtime Error: Division by zero
Fixed Version:

```
for i in range(3, -1, -1):
        print(10 / (i+1))
```

Output:
2.5

3.3333333333333335

5.0

10.0

SECTION B (Max Marks = 30 Marks)

1. Implement a Python script that simulates a basic calculator performing addition, subtraction, multiplication, division, and remainder operations based on user input. The user should enter the two operands (say 'x' and 'y') and an operator (+, - , *, /, and %). Based on the given operator you should perform the respective operation and display the output. Also verify if the output is a prime number or not. If it is a prime number print "The output is prime" otherwise print "The output is not prime". (**10 marks**)

```
x = float(input("Enter the first operand (x): "))
y = float(input("Enter the second operand (y): "))
operator = input("Enter the operator (+, -, *, /, %): ")

if operator == "+":
    result = x + y
elif operator == "-":
    result = x - y
elif operator == "*":
    result = x * y
elif operator == "/":
    if y == 0:
        print("Error: Division by zero is not allowed.")
        exit()
    result = x / y
elif operator == "%":
    if y == 0:
        print("Error: Modulus by zero is not allowed.")
        exit()
    result = x % y
else:
    print("Invalid operator.")
```

```
    exit()

print(f"{x} {operator} {y} = {result}")

# considering only the integer part
num = int(result)

# Prime check
if num < 2:
    is_prime = False
else:
    is_prime = True
    i = 2
    while i * i <= num: # Replaces sqrt
        if num % i == 0:
            is_prime = False
            break
        i += 1

if is_prime:
    print("The output is prime.")
else:
    print("The output is not prime.")
```

2. Write a program that takes as input your scores for 3 courses. It should then output your "highest" score and your final grade based on the average score. The grading system is as follows:

   A: Average score >= 85
   B: Average score >= 75 and < 85
   C: Average score >= 65 and < 75
   D: Average score >=50 and <65
   E: Average score >=30 and < 50
   F: Average score<30

   Solve the above using nested If-else statements. **Constraint:** Your program should not get a negative score. If the user gives a negative score, then it should print "invalid score". (**10 marks**)

```
scores = []
for i in range(3):
      score = float(input(f"Enter your score for course {i+1}: "))
      if score < 0:
          print("Invalid score. Please enter a non-negative score.")
      else:
          scores.append(score)

# Find the highest score
highest_score = scores[0]
for score in scores:
    if score > highest_score:
```

```
        highest_score = score

# Calculate the average
total = 0
for score in scores:
    total += score
average_score = total / 3

print("Your highest score is: ", highest_score)

if average_score < 0:
    grade = "Invalid score"
elif average_score >= 85:
    grade = "A"
elif average_score >= 75:
    grade = "B"
elif average_score >= 65:
    grade = "C"
elif average_score >= 50:
    grade = "D"
elif average_score >= 30:
    grade = "E"
else:
    grade = "F"

print("Your final grade is: ", grade)
```

3. Write a program that uses a for/while loop to print the following pattern

*****
******
****
****
****
1234
123
12
1
(**10 marks**)
[**Note**: You ***should not*** make use of only print statements to print the above pattern]

```
# Print the asterisk pattern
for i in range(1, 6):
    asterisk_str = ""
    j = 0
    while j < i: # String concatenation in a loop
        asterisk_str += "*"
        j += 1
    print(asterisk_str)
```

```
# Print the numerical pattern
for i in range(4, 0, -1):
    num_str = ""
    j = 1
    while j <= i:  # String concatenation in a loop
        num_str += str(j)
        j += 1
    print(num_str)
```