OOPS & Python Programming (4351108) - Summer 2025 Solution

Milav Dabgar

May 14, 2025

Question 1(a) [3 marks]

What is the purpose of a for loop in Python? Write an example.

Solution

A for loop is used to iterate over a sequence (like list, tuple, string) or other iterable objects and execute a block of code for each item in the sequence.

Code Example:

```python
# Print each fruit in a list
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

- **Iteration**: Automatically repeats code for each item
- **Simplicity**: Cleaner than using while loops with counters

Mnemonic

"For Each Item Do"

Question 1(b) [4 marks]

List out rules for defining variables in python and list out data types in python.

Solution

Rules for defining variables:

Table 1. Variable Rules

Rule	Example	Invalid Example
Must start with letter or underscore	`name = "John"`	`1name = "John"`
Can contain letters, numbers, underscores	`user_1 = "Alice"`	`user-1 = "Alice"`
Case-sensitive	`age` ≠ `Age`	-
Cannot use reserved keywords	`count = 5`	`if = 5`

Python Data Types:

Table 2. Data Types

Data Type	Description	Example
int	Integer numbers	`x = 10`
float	Decimal numbers	`y = 10.5`
str	Text strings	`name = "John"`
bool	Boolean values	`is_active = True`
list	Ordered, changeable	`["apple", "banana"]`
tuple	Ordered, unchangeable	`(10, 20)`
dict	Key-value pairs	`{"name": "John"}`
set	Unordered, unique	`{1, 2, 3}`

Mnemonic

"SILB-DTS: String, Integer, List, Boolean, Dictionary, Tuple, Set"

Question 1(c) [7 marks]

Create a program to print prime numbers between 1 to N.

Solution

```python
def print_primes(n):
    print("Prime numbers between 1 and", n, "are:")

    for num in range(2, n + 1):
        is_prime = True

        # Check if num is divisible by any number from 2 to sqrt(num)
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                is_prime = False
                break

        if is_prime:
            print(num, end=" ")

# Get input from user
N = int(input("Enter a number N: "))
print_primes(N)
```

Algorithm Diagram:

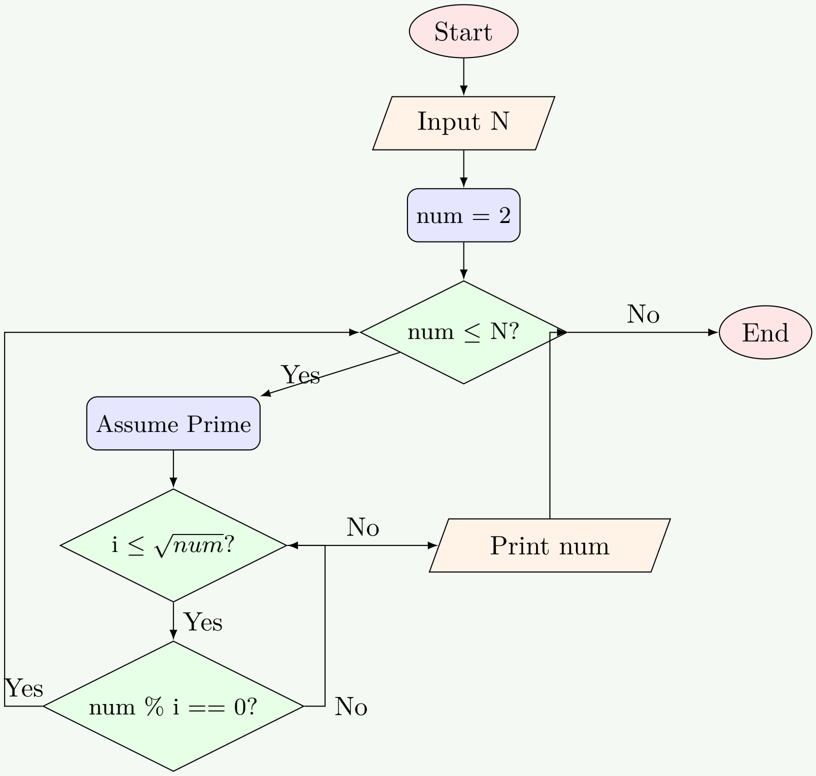

Figure 1. Prime Number Algorithm

- **Time complexity**: $O(N\sqrt{N})$
- **Space complexity**: $O(1)$

Mnemonic

"Divide To Decide Prime"

Question 1(c OR) [7 marks]

Explain working of break, continue and pass statement in Python with examples.

Solution

Table 3. Control Statements

Statement	Purpose	Example
break	Terminates loop completely	Stop search
continue	Skips to next iteration	Skip even nums
pass	Does nothing (placeholder)	Empty function

Flow Control Diagram:

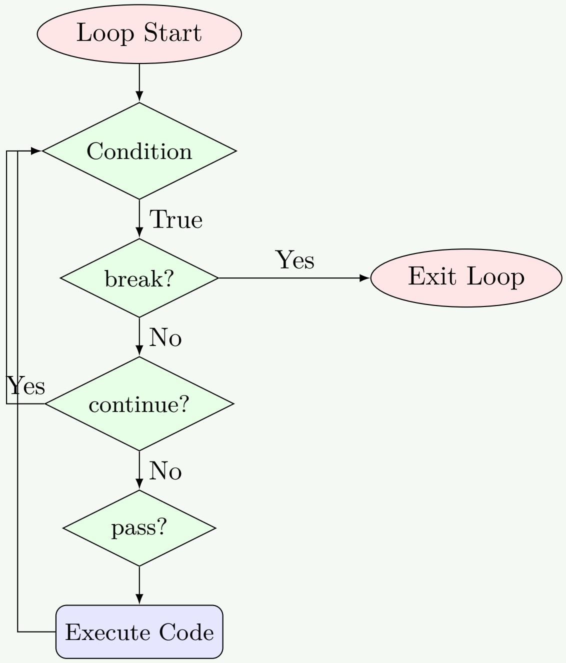

Figure 2. Loop Control Logic

Example Code:

```python
for i in range(5):
    if i == 2: continue  # Skip 2
    if i == 4: break     # Stop at 4
    if i == 0: pass      # Do nothing
    print(i)
# Output: 0, 1, 3
```

Mnemonic

"BCP: Break Completely, Continue Partially, Pass silently"

Question 2(a) [3 marks]

Create a program that asks the user for a year and prints out whether it is a leap year or not.

Solution

```python
year = int(input("Enter a year: "))

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
    print(f"{year} is a leap year")
else:
    print(f"{year} is not a leap year")
```

Decision Tree:

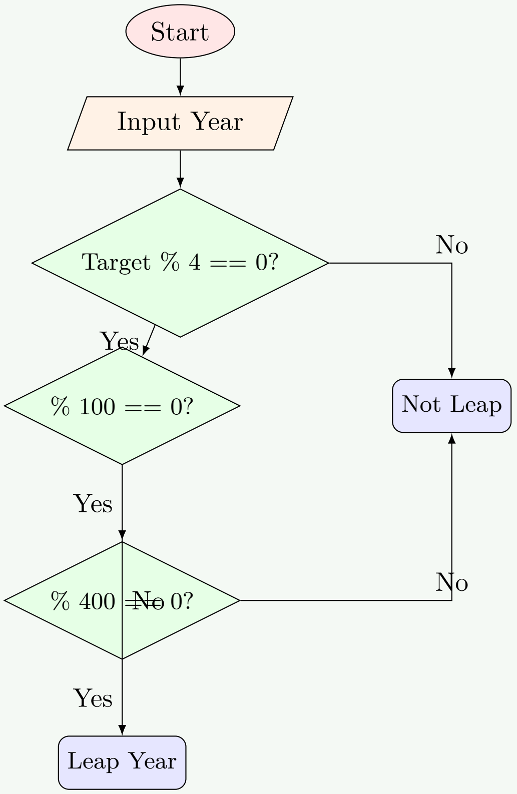

Figure 3. Leap Year Logic

Mnemonic

"4 Yes, 100 No, 400 Yes"

Question 2(b) [4 marks]

What are the key differences between a list and a tuple in Python?

Solution

Table 4. List vs Tuple

Feature	**List**	**Tuple**
Syntax	`[]`	`()`
Mutability	Mutable (Changeable)	Immutable (Fixed)
Performance	Slower	Faster
Use Case	Dynamic collections	Fixed data
Memory	More memory	Less memory

Comparison Diagram:

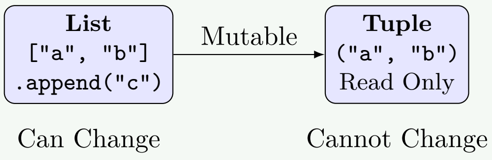

Figure 4. List vs Tuple

Mnemonic

"LIST: Transformable, TUPLE: Unchangeable"

Question 2(c) [7 marks]

Create a program to find the sum of all the positive numbers entered by the user. As soon as the user enters a negative number, stop taking in any further input from the user and display the sum.

Solution

```python
def sum_positives():
    total_sum = 0
    while True:
        num = float(input("Enter number (negative to stop): "))
        if num < 0:
            break
        total_sum += num
    print(f"Sum of positive numbers: {total_sum}")

sum_positives()
```

Process Flow:

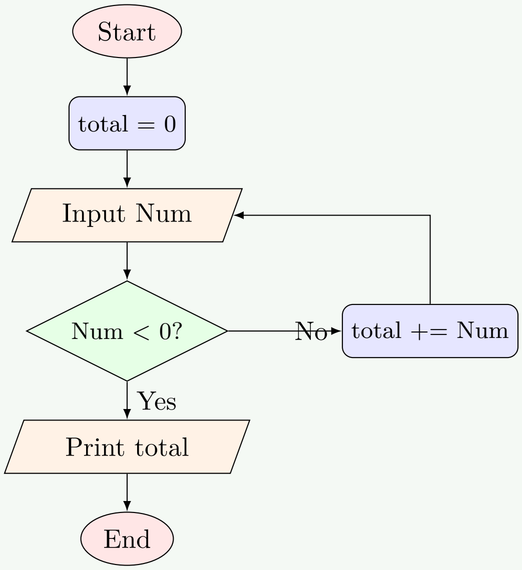

Figure 5. Summation Logic

Mnemonic

"Sum Till Negative"

Question 2(a OR) [3 marks]

Create a program to find a maximum number among the given three numbers.

Solution

```python
n1 = float(input("Num 1: "))
n2 = float(input("Num 2: "))
n3 = float(input("Num 3: "))

if n1 >= n2 and n1 >= n3:
    mx = n1
elif n2 >= n1 and n2 >= n3:
    mx = n2
else:
    mx = n3

print(f"Max: {mx}")
```

Logic Flow:

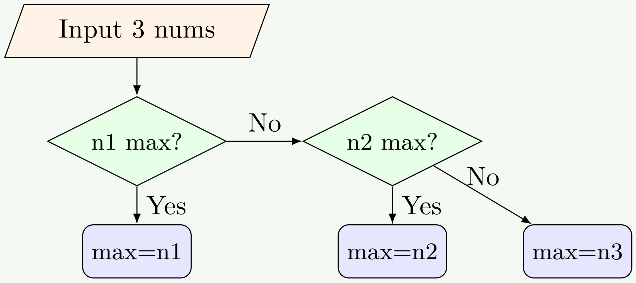

Figure 6. Maximum Finder Logic

Mnemonic

"Compare Each, Take Largest"

Question 2(b OR) [4 marks]

Given the str="abcdefghijklmnopqrstuvwxyz". Write a python program to extract every second character from above string.

Solution

```python
s = "abcdefghijklmnopqrstuvwxyz"
# Slice syntax: [start:end:step]
result = s[0::2]
print("Result:", result)
# Output: acegikmoqsuwy
```

Slicing Visualization:

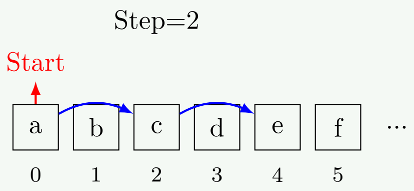

Figure 7. String Slicing Step 2

Mnemonic

"Slice Step Selector"

Question 2(c OR) [7 marks]

Write a Python program to create a dictionary that stores student names and their marks. Display the names of students who have scored more than 75 marks.

Solution

```python
students = {}
n = int(input("Enter count: "))

# Input Loop
for i in range(n):
    name = input("Name: ")
    marks = float(input("Marks: "))
    students[name] = marks

print("\nHigh Scorers (>75):")
for name, marks in students.items():
    if marks > 75:
        print(f"{name}: {marks}")
```

Process Diagram:

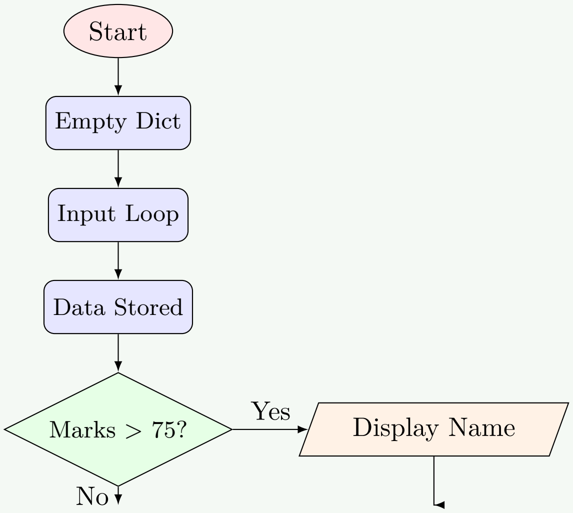

Figure 8. Dictionary Filtering

Mnemonic

"Store All, Filter Some"

Question 3(a) [3 marks]

Write a program to find the length of a string excluding spaces.

Solution

```python
s = input("Enter string: ")
no_spaces = s.replace(" ", "")
length = len(no_spaces)
print(f"Length excluding spaces: {length}")
```

Visualization:

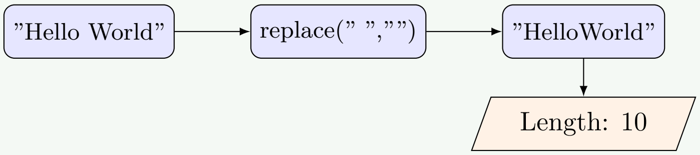

Figure 9. Space Removal

Mnemonic

“Count Characters, Skip Spaces”

Question 3(b) [4 marks]

List the dictionary methods in python and explain each with suitable examples.

Solution

Table 5. Dictionary Methods

Method	Description	Example
`get(k)`	Returns value for key	`d.get('a')`
`keys()`	Returns all keys	`list(d.keys())`
`values()`	Returns all values	`list(d.values())`
`items()`	Returns (key, value) pairs	`d.items()`
`pop(k)`	Removes item	`d.pop('a')`
`update()`	Merges dicts	`d.update(d2)`
`clear()`	Empties dict	`d.clear()`

Mnemonic

“GCUP-KPIV”

Question 3(c) [7 marks]

Explain Python’s List data type in detail.

Solution

List is an ordered, mutable collection that allows duplicate elements and mixed types.

Key Features:

- **Ordered**: Items maintain order.
- **Mutable**: Can add, remove, change items.
- **Heterogeneous**: Can store int, str, float together.

List Operations Diagram:

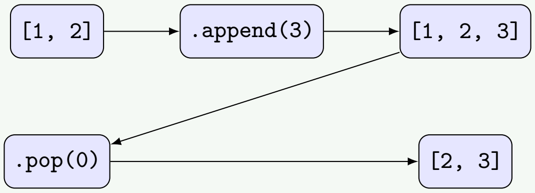

Figure 10. List Operations

Mnemonic

"CAMP-IS: Create, Access, Modify, Process"

Question 3(a OR) [3 marks]

Write a program to input a string from the user and print it in the reverse order without creating a new string.

Solution

```python
s = input("Enter string: ")
print(f"Reversed: {s[::-1]}")
```

Reversing Logic:

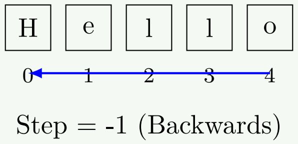

Figure 11. String Reversal

Mnemonic

"Slice Backwards"

Question 3(b OR) [4 marks]

List the dictionary operations in python and explain each with suitable examples.

Solution

Table 6. Dictionary Operations

Operation	Syntax	Description
Access	`d['key']`	Get value
Add/Mod	`d['k'] = v`	Insert/Update
Delete	`del d['k']`	Remove pair
Check	`'k' in d`	Membership
Length	`len(d)`	Count items

Mnemonic

"CADMIL: Create Access Delete Modify Iterate Length"

Question 3(c OR) [7 marks]

Explain Python's set data type in detail.

Solution

Set is an unordered collection of unique elements.

Set Characteristics:

- **Unique**: No duplicates.
- **Unordered**: No index access.
- **Math Ops**: Supports union, intersection.

Set Operations Diagram:

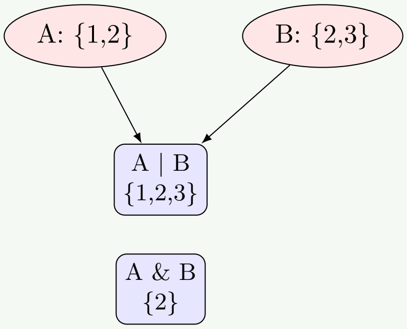

Figure 12. Set Union & Intersection

Mnemonic

"SUMO: Set Unique Mutable Ordered-less"

Question 4(a) [3 marks]

Explain statistics module with any three methods.

Solution

Table 7. Statistics Methods

Method	Description	Example
`mean()`	Average	`mean([1,2,3])` → 2
`median()`	Middle value	`median([1,5,9])` → 5
`mode()`	Most Frequent	`mode([1,1,2])` → 1

Mnemonic

"MMM Stats"

Question 4(b) [4 marks]

Explain function of user define function and user defined module in Python.

Solution

Table 8. Function vs Module

Feature	Function	Module
Unit	Code Block	File (.py)
Creation	`def name():`	Save as .py
Usage	Call `name()`	`import name`
Scope	Local	Global/Imported

Module Structure:

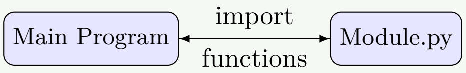

Figure 13. Import Relationship

Mnemonic

"FIR-MID"

Question 4(c) [7 marks]

Write a Python code using user defined function to find the factorial of a given number using recursion.

Solution

```python
def factorial(n):
    # Base case
    if n == 0 or n == 1:
        return 1
    # Recursive case
    else:
        return n * factorial(n-1)

num = int(input("Enter num: "))
print(f"Factorial: {factorial(num)}")
```

Recursion Visualization:

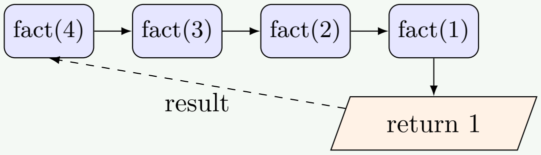

Figure 14. Recursion Chain

Mnemonic

"Number times (Number minus one)!"

Question 4(a OR) [3 marks]

Explain math module with any three methods.

Solution

Table 9. Math Methods

Method	Description	Example
`sqrt()`	Square Root	`sqrt(16)` → 4.0
`pow()`	Power	`pow(2,3)` → 8.0
`ceil()`	Round Up	`ceil(4.1)` → 5

Mnemonic

"SPT Math"

Question 4(b OR) [4 marks]

Explain the concepts of global and local variables in Python.

Solution

Table 10. Variable Scope

Type	Scope	Access
Global	Entire Program	Anywhere
Local	Inside Function	Within Function Only

Scope Diagram:

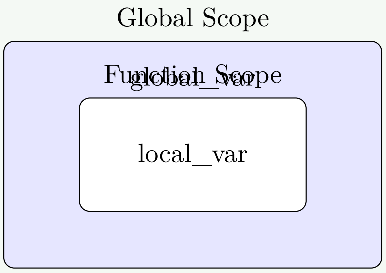

Figure 15. Scope Hierarchy

Mnemonic

"GLOBAL Goes Everywhere, LOCAL Lives in Functions"

Question 4(c OR) [7 marks]

Create code with user defined function to check if given string is palindrome or not.

Solution

```python
def is_palindrome(text):
    raw = text.replace(" ", "").lower()
    return raw == raw[::-1]

s = input("Enter string: ")
if is_palindrome(s):
    print("Palindrome")
else:
    print("Not Palindrome")
```

Logic Flow:

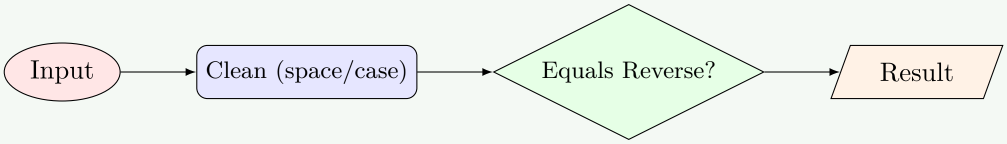

Figure 16. Palindrome Check

Mnemonic

"Clean, Reverse, Compare"

Question 5(a) [3 marks]

Define class and object with example.

Solution

- **Class**: Blueprint/Template.
- **Object**: Instance of class.

Relationship Diagram:

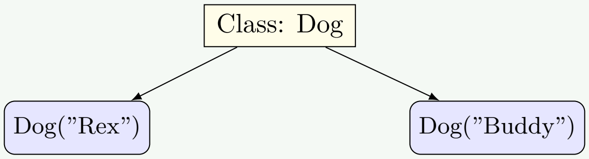

Figure 17. Instantiation

Mnemonic

"CAMBO: Classes Are Molds, Build Objects"

Question 5(b) [4 marks]

Classify constructor. Explain any one in detail.

Solution

Types: Default, Parameterized, Non-parameterized, Copy.

Parameterized Constructor:

```python
class Student:
    def __init__(self, name):
        self.name = name
s = Student("Alice")
```

Execution Flow:

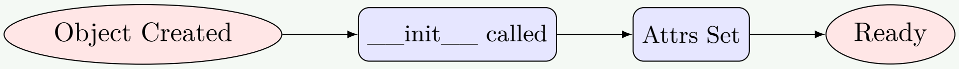

Figure 18. Constructor Lifecycle

Mnemonic

“PICAN”

Question 5(c) [7 marks]

Develop and explain a python code to implement hierarchical inheritance.

Solution

```python
class Vehicle:
    def start(self): print("Engine On")

class Car(Vehicle):
    def drive(self): print("Driving")

class Bike(Vehicle):
    def ride(self): print("Riding")

c = Car(); c.start()
b = Bike(); b.start()
```

Inheritance Tree:

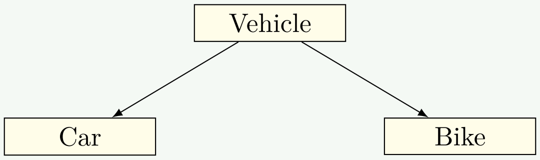

Figure 19. Hierarchical Inheritance

Mnemonic

“Parents Share, Children Specialize”

Question 5(a OR) [3 marks]

What is the __init__ method in Python? Explain its purpose with a suitable example.

Solution

Special method automatically called during object creation to initialize attributes.
Example:

```python
class Rect:
    def __init__(self, w, h):
        self.w = w
        self.h = h
```

Mnemonic

"ASAP: Attributes Set At Production"

Question 5(b OR) [4 marks]

Classify methods in Python class. Explain any one in detail.

Solution

Table 11. Method Types

Type	Access	Decorator
Instance	`self`	None
Class	`cls`	`@classmethod`
Static	None	`@staticmethod`

Instance Method:

```python
def display(self):
    print(self.name)
```

Mnemonic

"SIAM: Self Is Always Mentioned"

Question 5(c OR) [7 marks]

Develop a Python code for Polymorphism and explain it.

Solution

```python
class Dog:
    def sound(self): return "Woof"

class Cat:
    def sound(self): return "Meow"

animals = [Dog(), Cat()]
for a in animals:
    print(a.sound())
```

Polymorphism Diagram:

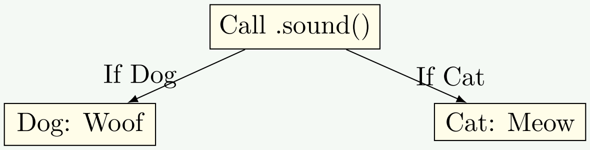

Figure 20. Dynamic Binding

Mnemonic

"Same Method, Different Behavior"