

Python Programming (4311601)

Milav Dabgar

Winter 2023

Question Question 1(a) [03 marks]

What is Flow chart? List out symbols used in Flow chart.

Solution

A **flowchart** is a graphical representation of an algorithm that shows the sequence of steps and decision points in a process using standardized symbols.

Key Points:

- **Visual representation:** Shows program logic graphically
- **Step-by-step:** Displays sequential flow of operations
- **Decision making:** Diamond symbols show conditional branches

Mnemonic

“Flow Charts Show Program Steps Visually”

Question Question 1(b) [04 marks]

Write a short note on for loop.

Solution

The **for loop** is used to iterate over a sequence (list, tuple, string, range) in Python.

Simple Code Example:

```
1 for i in range(3):
2     print(i)
3 # Output: 0, 1, 2
```

Key Features:

- **Automatic iteration:** No manual counter needed
- **Sequence traversal:** Works with any iterable object
- **Range function:** Creates number sequences easily

Mnemonic

“For Loops Iterate Through Sequences”

Question Question 1(c) [07 marks]

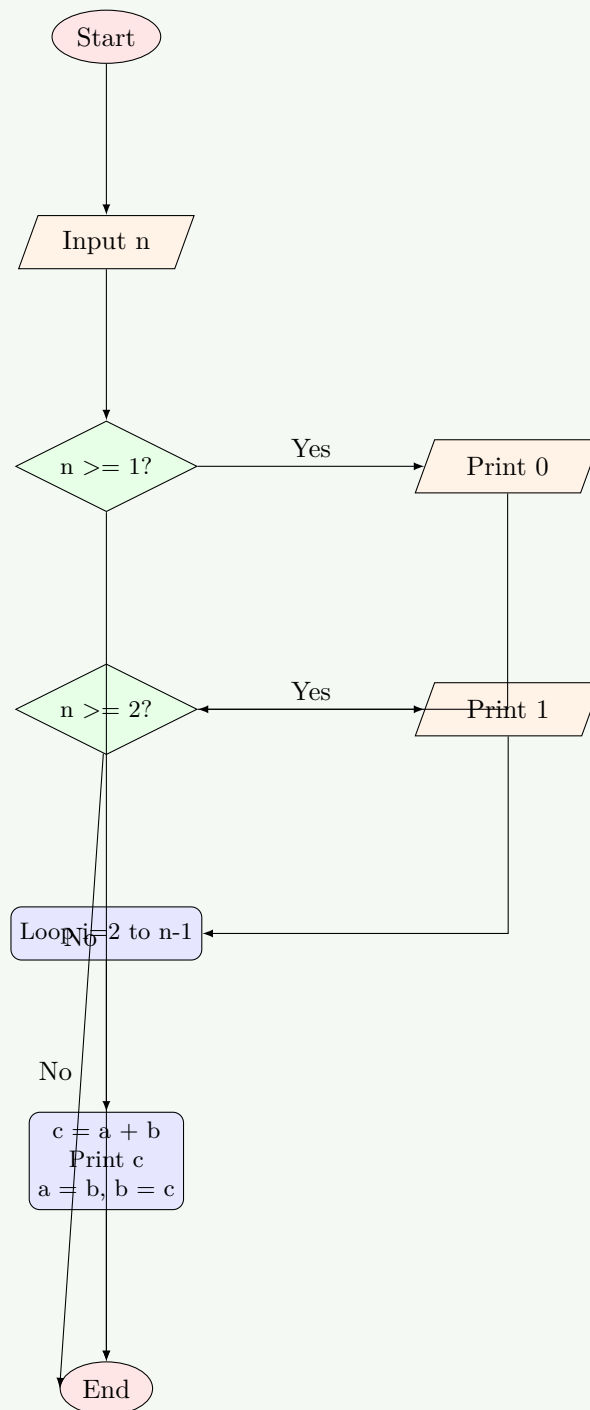
Write a program to display Fibonacci series up to nth term where n is provided by the user.

Solution

Fibonacci Series Program:

```
1  # Get number of terms from user
2  n = int(input("Enter number of terms: "))
3
4  # Initialize first two terms
5  a, b = 0, 1
6
7  # Display first term
8  if n >= 1:
9      print(a, end=" ")
10
11 # Display second term
12 if n >= 2:
13     print(b, end=" ")
14
15 # Generate remaining terms
16 for i in range(2, n):
17     c = a + b
18     print(c, end=" ")
19     a, b = b, c
```

Algorithm Flow:

**Key Concepts:**

- **Sequential generation:** Each term = sum of previous two
- **Variable swapping:** Update a, b values efficiently
- **User input:** Dynamic series length

Mnemonic

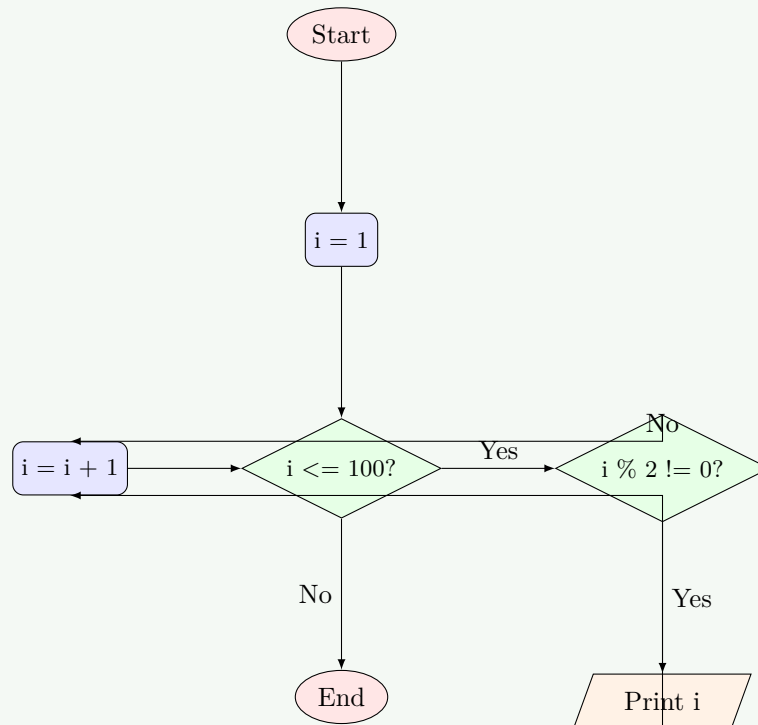
“Fibonacci: Add Previous Two Numbers”

Question Question 1(c OR) [07 marks]

Draw a flow chart to print ODD numbers from 1 to 100.

Solution

Flowchart for ODD Numbers 1 to 100:



Corresponding Python Code:

```

1 for i in range(1, 101):
2     if i % 2 != 0:
3         print(i, end=" ")
  
```

Alternative Method:

```

1 for i in range(1, 101, 2):
2     print(i, end=" ")
  
```

Key Elements:

- **Loop control:** i from 1 to 100
- **Odd check:** $i \% 2 \neq 0$ condition
- **Step increment:** Move to next number

Mnemonic

“Odd Numbers: Remainder 1 When Divided by 2”

Question Question 2(a) [03 marks]

Write a Program to find whether a number is Palindrome or not.

Solution

Palindrome Check Program:

```

1  # Input number
2  num = int(input("Enter a number: "))
3  temp = num
4  reverse = 0
5
6  # Reverse the number
7  while temp > 0:
8      reverse = reverse * 10 + temp % 10
9      temp = temp // 10
10
11 # Check palindrome
12 if num == reverse:
13     print(f"{num} is palindrome")
14 else:
15     print(f"{num} is not palindrome")

```

Algorithm Table:

Key Points:

- **Digit extraction:** Use modulo (%) operator
- **Reverse building:** Multiply by 10 and add digit
- **Comparison:** Original equals reversed

Mnemonic

“Palindrome Reads Same Forward Backward”

Question Question 2(b) [04 marks]

Explain features of Python Programming.

Solution

Python Features Table:

Key Advantages:

- **Beginner-friendly:** Easy to learn and understand
- **Versatile:** Web development, AI, data science
- **Community support:** Large developer community
- **Dynamic typing:** No variable type declaration needed

Mnemonic

“Python: Easy, Powerful, Popular Programming”

Question Question 2(c) [07 marks]

Explain basic structure of Python Program.

Solution

Python Program Structure:

```

1  #!/usr/bin/env python3

```

```

2  # Shebang line (optional)
3
4  """
5  Documentation string (docstring)
6  Describes program purpose
7  """
8
9  # Import statements
10 import math
11 from datetime import date
12
13 # Global variables
14 PI = 3.14159
15 count = 0
16
17 # Function definitions
18 def calculate_area(radius):
19     """Calculate circle area"""
20     return PI * radius * radius
21
22 # Class definitions
23 class Calculator:
24     def __init__(self):
25         self.result = 0
26
27 # Main program execution
28 if __name__ == "__main__":
29     # Program logic here
30     radius = 5
31     area = calculate_area(radius)
32     print(f"Area: {area}")

```

Structure Components Table:

Key Principles:

- **Indentation:** Defines code blocks (4 spaces recommended)
- **Comments:** Use # for single line, """ """ for multi-line
- **Modularity:** Organize code in functions and classes

Mnemonic

“Structure: Import, Define, Execute”

Question Question 2(a OR) [03 marks]

Write a Program to reverse a string.

Solution

String Reversal Program:

```

1  # Method 1: Using slicing
2  string = input("Enter a string: ")
3  reversed_string = string[::-1]
4  print(f"Reversed: {reversed_string}")
5
6  # Method 2: Using loop
7  string = input("Enter a string: ")
8  reversed_string = ""
9  for char in string:

```

```

10     reversed_string = char + reversed_string
11     print(f"Reversed: {reversed_string}")

```

Reversal Methods Table:

Mnemonic

“Reverse: Last Character First”

Question Question 2(b OR) [04 marks]

Explain Logical Operators with example.

Solution

Python Logical Operators:

Example Code:

```

1  a = 10
2  b = 5
3
4  # AND operator
5  if a > 5 and b < 10:
6      print("Both conditions true")
7
8  # OR operator
9  if a > 15 or b < 10:
10     print("At least one condition true")
11
12 # NOT operator
13 if not (a < 5):
14     print("a is not less than 5")

```

Truth Table:

Mnemonic

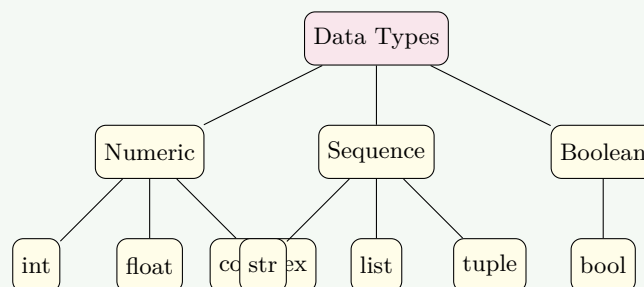
“AND needs All, OR needs One, NOT reverses”

Question Question 2(c OR) [07 marks]

Explain different Data Types in Python Programming language

Solution

Python Data Types Classification:



Data Types Table:**Example Code:**

```

1  # Numeric types
2  age = 25          # int
3  price = 99.99     # float
4  complex_num = 3+4j # complex
5
6  # Sequence types
7  name = "Python"   # string
8  numbers = [1,2,3,4] # list
9  coordinates = (10,20) # tuple
10
11 # Other types
12 is_active = True   # boolean
13 unique_items = {1,2,3} # set
14 student = {"name": "John", "age": 20} # dict

```

Mnemonic

“Python Types: Numbers, Sequences, Collections”

Question Question 3(a) [03 marks]

What is flow control in Python? Explain with example

Solution

Flow control manages the execution order of program statements using conditional and loop structures.

Flow Control Types Table:**Example Code:**

```

1  # Selection example
2  age = 18
3  if age >= 18:
4      print("Adult")
5  else:
6      print("Minor")
7
8  # Iteration example
9  for i in range(3):
10     print(f"Count: {i}")

```

Mnemonic

“Flow Control: Decide, Repeat, Jump”

Question Question 3(b) [04 marks]

Write a program to explain nested if statement.

Solution

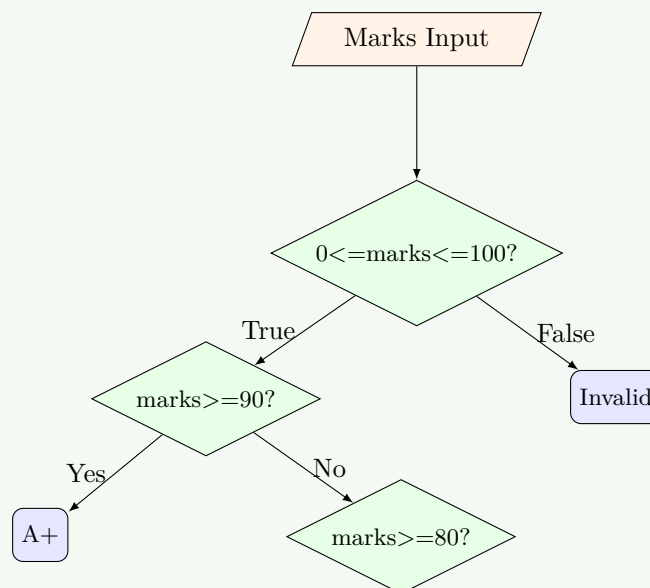
Nested If Statement Program:

```

1  # Grade calculation using nested if
2  marks = int(input("Enter marks: "))
3
4  if marks >= 0 and marks <= 100:
5      if marks >= 90:
6          grade = "A+"
7      elif marks >= 80:
8          if marks >= 85:
9              grade = "A"
10         else:
11             grade = "B+"
12     elif marks >= 70:
13         grade = "B"
14     elif marks >= 60:
15         grade = "C"
16     else:
17         grade = "F"
18     print(f"Grade: {grade}")
19 else:
20     print("Invalid marks")

```

Nested Structure Diagram:



Mnemonic

"Nested If: Decisions Within Decisions"

Question Question 3(c) [07 marks]

Write a program to Explain types of Arguments and Parameters.

Solution

Types of Arguments and Parameters:

```

1 # 1. Positional Arguments
2 def greet(name, age):
3     print(f"Hello {name}, you are {age} years old")
4
5 greet("John", 25) # Positional arguments
6
7 # 2. Keyword Arguments
8 greet(age=30, name="Alice") # Keyword arguments
9
10 # 3. Default Parameters
11 def introduce(name, city="Unknown"):
12     print(f"{name} lives in {city}")
13
14 introduce("Bob") # Uses default value
15 introduce("Carol", "NYC") # Override default
16
17 # 4. Variable-length Arguments (*args)
18 def sum_all(*numbers):
19     return sum(numbers)
20
21 result = sum_all(1, 2, 3, 4, 5)
22 print(f"Sum: {result}")
23
24 # 5. Keyword Variable Arguments (**kwargs)
25 def display_info(**info):
26     for key, value in info.items():
27         print(f"{key}: {value}")
28
29 display_info(name="David", age=28, city="Boston")

```

Parameters Types Table:

Mnemonic

“Parameters: Position, Keywords, Defaults, Variables”

Question Question 3(a OR) [03 marks]

Explain break and continue statement with example.

Solution

Break and Continue Statements:

Break Statement:

```

1 # Break example - exit loop
2 for i in range(10):
3     if i == 5:
4         break
5     print(i)
6 # Output: 0, 1, 2, 3, 4

```

Continue Statement:

```

1 # Continue example - skip iteration
2 for i in range(5):
3     if i == 2:
4         continue
5     print(i)
6 # Output: 0, 1, 3, 4

```

Comparison Table:

Mnemonic

“Break Exits, Continue Skips”

Question Question 3(b OR) [04 marks]

Create a program to display the following pattern

Solution

Pattern:

```
1
12
123
1234
12345
```

Number Pattern Program:

```
1 # Method 1: Using nested loops
2 rows = 5
3 for i in range(1, rows + 1):
4     for j in range(1, i + 1):
5         print(j, end="")
6     print() # New line
7
8 # Method 2: Using string manipulation
9 for i in range(1, 6):
10     line = ""
11     for j in range(1, i + 1):
12         line += str(j)
13     print(line)
```

Mnemonic

“Pattern: Row Number Determines Column Count”

Question Question 3(c OR) [07 marks]

Explain the following mathematical functions by writing a code for each: 1. abs() 2. max()
3. pow() 4. sum()

Solution

Mathematical Functions in Python:

```
1 # 1. abs() - Absolute value
2 numbers = [-5, 3.7, -10.2, 0]
3 print("abs() function examples:")
4 for num in numbers:
5     print(f"abs({num}) = {abs(num)}")
```

```

6
7 # 2. max() - Maximum value
8 list1 = [4, 7, 2, 9, 1]
9 print(f"\nmax() function examples:")
10 print(f"max({list1}) = {max(list1)}")
11 print(f"max(10, 25, 5) = {max(10, 25, 5)}")
12 print(f"max('hello') = {max('hello')}") # Alphabetically
13
14 # 3. pow() - Power function
15 print(f"\npow() function examples:")
16 print(f"pow(2, 3) = {pow(2, 3)}") # 2^3 = 8
17 print(f"pow(5, 2) = {pow(5, 2)}") # 5^2 = 25
18 print(f"pow(8, 1/3) = {pow(8, 1/3)}") # Cube root of 8
19
20 # 4. sum() - Sum of sequence
21 numbers = [1, 2, 3, 4, 5]
22 print(f"\nsum() function examples:")
23 print(f"sum({numbers}) = {sum(numbers)}")
24 print(f"sum({numbers}, 10) = {sum(numbers, 10)}") # With start value

```

Functions Summary Table:

Mnemonic

"Math Functions: Absolute, Maximum, Power, Sum"

Question Question 4(a) [03 marks]

Explain scope of variables.

Solution

Variable Scope refers to the region where a variable can be accessed in a program.

Scope Types Table:

Mnemonic

"Scope: Local Lives in Functions, Global Lives Everywhere"

Question Question 4(b) [04 marks]

Develop a program to create nested LOOP and display numbers.

Solution

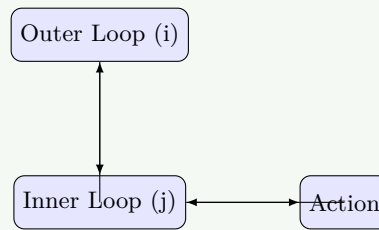
Nested Loop Program:

```

1 # Example 1: Number grid
2 print("Number Grid Pattern:")
3 for i in range(1, 4):
4     for j in range(1, 5):
5         print(f"{i}-{j}", end=" ")
6     print() # New line after each row

```

Nested Loop Structure:

**Mnemonic**

“Nested Loops: Outer Controls Inner”

Question Question 4(c) [07 marks]

Write a program to create a list of ODD and EVEN numbers in range of 1 to 50.

Solution**ODD and EVEN Numbers Program:**

```

1  # Method 1: Using loops and conditions
2  odd_numbers = []
3  even_numbers = []
4
5  for i in range(1, 51):
6      if i % 2 == 0:
7          even_numbers.append(i)
8      else:
9          odd_numbers.append(i)
10
11 print("ODD Numbers (1-50):")
12 print(odd_numbers)
13 print(f"Count: {len(odd_numbers)}")
14
15 print("\nEVEN Numbers (1-50):")
16 print(even_numbers)
17 print(f"Count: {len(even_numbers)}")
  
```

Number Classification Table:**Mnemonic**

“Odd/Even: Remainder 1/0 When Divided by 2”

Question Question 4(a OR) [03 marks]

Explain String Slicing with example.

Solution

String Slicing extracts parts of a string using [start:stop:step] syntax.

Slicing Syntax Table:

Mnemonic

“Slice: Start, Stop, Step”

Question Question 4(b OR) [04 marks]

Write a program using user defined function to find the factorial of a given number.

Solution**Factorial Function Program:**

```

1 def factorial(n):
2     """Calculate factorial using recursion"""
3     if n == 0 or n == 1:
4         return 1
5     else:
6         return n * factorial(n - 1)
7
8 def factorial_iterative(n):
9     """Calculate factorial using loop"""
10    result = 1
11    for i in range(1, n + 1):
12        result *= i
13    return result
14
15 # Main program
16 number = int(input("Enter a number: "))
17 if number < 0:
18     print("Factorial not defined for negative numbers")
19 else:
20     result1 = factorial(number)
21     print(f"Factorial of {number} = {result1}")

```

Mnemonic

“Factorial: Multiply All Numbers Below”

Question Question 4(c OR) [07 marks]

Write a user defined function to check whether a sub string is present in a given string.

Solution**Substring Check Function:**

```

1 def find_substring(main_string, sub_string):
2     """Check if substring exists in main string"""
3     if sub_string in main_string:
4         index = main_string.find(sub_string)
5         return True, index
6     else:
7         return False, -1
8

```

```

9  # Main program
10 text = input("Enter main string: ")
11 search = input("Enter substring to search: ")
12
13 found, position = find_substring(text, search)
14 if found:
15     print(f"Substring '{search}' found at position {position}")
16 else:
17     print(f"Substring '{search}' not found")

```

String Methods Table:

Mnemonic

“Substring: Search, Find, Count, Position”

Question Question 5(a) [03 marks]

Explain how to create and access a List with example.

Solution

List Creation and Access:

```

1  # Creating lists
2  numbers = [1, 2, 3, 4, 5]
3
4  # Accessing elements
5  print(f"First element: {numbers[0]}")      # 1
6  print(f"Last element: {numbers[-1]}")      # 5
7  print(f"Slice: {numbers[1:4]}")            # [2, 3, 4]

```

List Access Methods:

Mnemonic

“Lists: Create, Index, Access”

Question Question 5(b) [04 marks]

List out the operations that can be performed on a LIST. Write a program to create and copy one List into another List.

Solution

List Operations and Copy Program:

```

1  # Original list
2  original = [1, 2, 3, 4, 5]
3  print(f"Original list: {original}")
4
5  # Copying methods
6  shallow_copy = original.copy()
7  slice_copy = original[:]
8  list_copy = list(original)
9

```

```

10 # Modify original
11 original.append(6)
12 print(f"After append: {original}")
13 print(f"Shallow copy: {shallow_copy}")

```

List Operations Table:

Mnemonic

“List Operations: Add, Insert, Remove, Pop, Copy”

Question Question 5(c) [07 marks]

List and give use of various Built in methods of LIST

Solution

Built-in List Methods:

```

1 # Sample list for demonstrations
2 fruits = ['apple', 'banana', 'cherry', 'apple']
3
4 # Modification methods
5 fruits.append('date')           # Add to end
6 fruits.insert(1, 'avocado')     # Insert at index
7 fruits.remove('apple')          # Remove first occurrence
8 last_fruit = fruits.pop()       # Remove and return last
9
10 # Search and count methods
11 count = fruits.count('apple')   # Count occurrences
12 index = fruits.index('banana') # Find first index
13
14 # Sorting and reversing
15 fruits.sort()                  # Sort in place
16 fruits.reverse()               # Reverse in place

```

List Methods Summary:

Mnemonic

“List Methods: Add, Remove, Search, Sort, Copy”

Question Question 5(a OR) [03 marks]

Explain how to create and traverse a string by giving an example.

Solution

String Creation and Traversal:

```

1 # String creation methods
2 string1 = "Hello World"      # Double quotes
3 string2 = 'Python'           # Single quotes
4
5 # String traversal methods
6 text = "Python"

```

```

7
8 # Method 1: Using for loop
9 for char in text:
10     print(char, end=" ")

```

Traversal Methods Table:

Mnemonic

“Strings: Create, Loop, Access”

Question Question 5(b OR) [04 marks]

List out the operations that can be performed on a String. Write a code for any 2 operations

Solution

String Operations:

```

1 # Operation 1: String concatenation
2 first_name = "John"
3 last_name = "Doe"
4 full_name = first_name + " " + last_name
5 print(f"Concatenation: {full_name}")
6
7 # Operation 2: String case conversion
8 sentence = "learn python"
9 title_case = sentence.title()
10 print(f"Title case: {title_case}")

```

String Operations Table:

Mnemonic

“String Operations: Join, Case, Split, Find”

Question Question 5(c OR) [07 marks]

List and give use of various built – in methods of String.

Solution

Built-in String Methods:

```

1 # Sample string for demonstration
2 text = " Python Programming "
3
4 # Case conversion methods
5 print(f"upper(): {text.upper()}")
6 print(f"lower(): {text.lower()}")
7
8 # Whitespace methods
9 print(f"strip(): '{text.strip()}'")
10
11 # Search and check methods

```

```
12 print(f"find('Python'): {text.find('Python')}")
13 print(f"count('P'): {text.count('P')}")
14
15 # Split and join methods
16 words = text.split()
17 print(f"split(): {words}")
```

String Methods Classification:

Mnemonic

“String Methods: Case, Clean, Check, Change”