Python Programming (4311601) - Winter 2024 Solution

Milav Dabgar

January 13, 2025

Question Question 1(a) [03 marks]

Define Problem Solving, Algorithm and Pseudo Code.

Solution

Definitions:
Key Points:

- **Problem Solving**: Breaking down complex problems into manageable steps
- **Algorithm**: Must be finite, definite, effective, and produce correct output
- **Pseudo Code**: Bridge between human language and programming code

Mnemonic

"PAP - Problem, Algorithm, Pseudo"

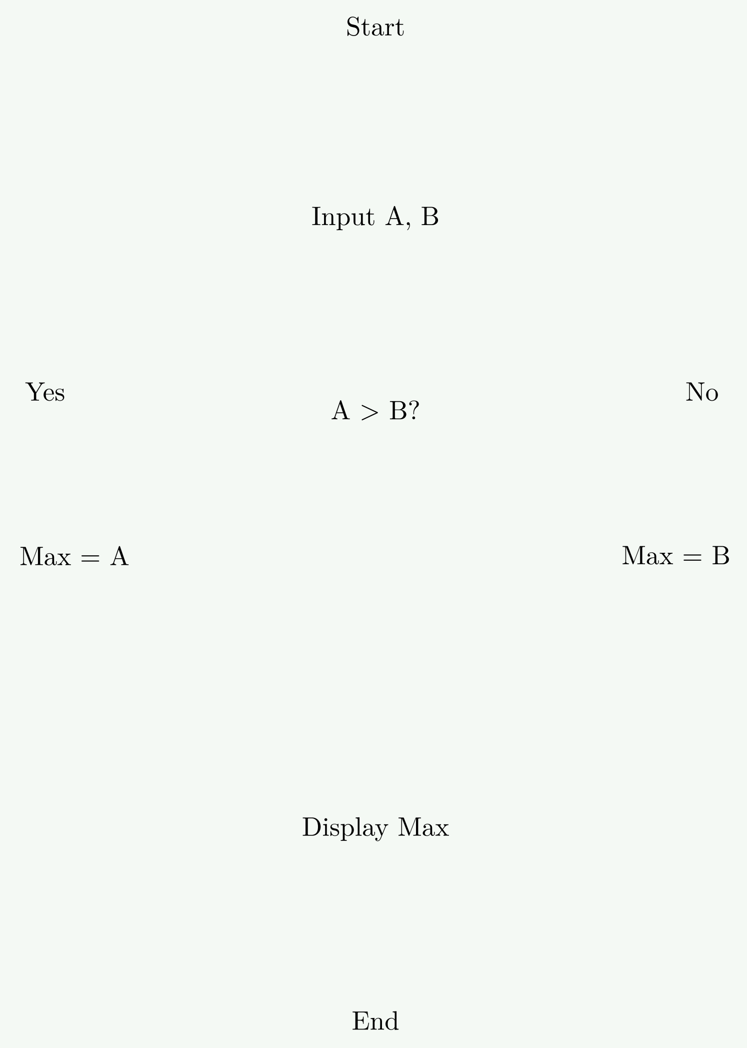
Question Question 1(b) [04 marks]

Explain various Flowchart Symbols. Design a Flowchart to find maximum number out of two given numbers

Solution

Flowchart Symbols:

Flowchart for Maximum of Two Numbers:



Explanation:

- **Start/End**: Entry and exit points
- **Input/Output**: Data flow operations
- **Decision**: Conditional branching
- **Process**: Computational steps

Mnemonic

"SIPO - Start, Input, Process, Output"

Question Question 1(c) [07 marks]

List out various arithmetic operators of python. Write Python Code that performs various arithmetic operations.

Solution

Arithmetic Operators:

Code:

```python
a = 10
b = 3
print(f"Addition: {a + b}")
print(f"Subtraction: {a - b}")
print(f"Multiplication: {a * b}")
print(f"Division: {a / b}")
print(f"Floor Division: {a // b}")
print(f"Modulus: {a % b}")
print(f"Power: {a ** b}")
```

Mnemonic

"Add-Sub-Mul-Div-Floor-Mod-Pow"

Question Question 1(c OR) [07 marks]

List out various comparison operators of python. Write Python Code which performs various comparison operations.

Solution

Comparison Operators:
Code:

```python
x = 8
y = 5
print(f"Equal: {x == y}")
print(f"Not Equal: {x != y}")
print(f"Greater: {x > y}")
print(f"Less: {x < y}")
print(f"Greater Equal: {x >= y}")
print(f"Less Equal: {x <= y}")
```

Mnemonic

"Equal-Not-Greater-Less-GreaterEqual-LessEqual"

Question Question 2(a) [03 marks]

Write short note on membership operators.

Solution

Membership Operators:

Key Points:
- **in operator**: Returns True if element found in sequence
- **not in operator**: Returns True if element not found in sequence
- **Usage**: Lists, strings, tuples, dictionaries

Mnemonic

"In-Not-In for membership testing"

Question Question 2(b) [04 marks]

Define Python. Write down various applications of Python Programming.

Solution

Python Definition: High-level, interpreted programming language known for simplicity and readability.
Applications:
Features:
- **Interpreted**: No compilation needed
- **Cross-platform**: Runs on multiple OS
- **Large libraries**: Extensive standard library

Mnemonic

"Web-Data-AI-Desktop-Games"

Question Question 2(c) [07 marks]

Write python program which calculates electricity bill using following details.

Solution

Table of Rates:

Code:

```python
units = int(input("Enter consumed units: "))

if units <= 100:
    bill = units * 5.00
elif units <= 200:
    bill = units * 7.50
elif units <= 300:
    bill = units * 10.00
else:
    bill = units * 15.00

print(f"Total Bill: Rs {bill}")
```

Explanation:

- **Conditional logic**: if-elif-else structure
- **Rate calculation**: Based on unit slabs
- **User input**: Interactive billing system

Mnemonic

"Input-Check-Calculate-Display"

Question Question 2(a OR) [03 marks]

Write short note on identity operators.

Solution

Identity Operators:
Key Points:

- **is operator**: Compares object identity, not values
- **is not operator**: Checks if objects are different
- **Memory comparison**: Checks same memory location

Mnemonic

"Is-IsNot for object identity"

Question Question 2(b OR) [04 marks]

What is indentation in Python? Explain various features of Python.

Solution

Indentation: Whitespace at line beginning to define code blocks.
Features:

Importance of Indentation:

- **Indentation**: Replaces curly braces
- **Consistent**: Usually 4 spaces per level
- **Mandatory**: Creates code structure

Mnemonic

"Simple-Interpreted-Object-Cross-Large"

Question Question 2(c OR) [07 marks]

Write a python program that calculates Student's class/grade using following details.

Solution

Grading Table:
Code:

```python
percentage = float(input("Enter percentage: "))

if percentage >= 70:
    grade = "Distinction"
elif percentage >= 60:
    grade = "First Class"
elif percentage >= 50:
    grade = "Second Class"
elif percentage >= 35:
    grade = "Pass Class"
else:
    grade = "Fail"

print(f"Grade: {grade}")
```

Explanation:

- **Multiple conditions**: Nested if-elif structure
- **Grade assignment**: Based on percentage ranges
- **Float input**: Handles decimal percentages

Mnemonic

"Distinction-First-Second-Pass-Fail"

Question Question 3(a) [03 marks]

What is Selection Control Statement? List it out.

Solution

Selection Control Statements:

Key Concepts:

- **Selection statements**: Control program flow based on conditions
- **Boolean evaluation**: Uses True/False logic
- **Branching**: Different paths of execution

Mnemonic

"If-IfElse-IfElif-Nested"

Question Question 3(b) [04 marks]

Write short note on nested loops.

Solution

Nested Loops:
Key Points:

- **Nested structure**: Loop inside another loop
- **Complete execution**: Inner loop finishes before outer continues
- **Pattern creation**: Useful for 2D structures

Code Example:

```python
for i in range(3):
    for j in range(2):
        print(f"i={i}, j={j}")
```

Mnemonic

"Outer-Inner-Complete-Pattern"

Question Question 3(c) [07 marks]

Write a user-define function that displays all numbers, which are divisible by 4 from 1 to 100.

Solution

Code:

```python
def display_divisible_by_4():
    print("Numbers divisible by 4 from 1 to 100:")
    for num in range(1, 101):
        if num % 4 == 0:
            print(num, end=" ")
    print()

# Function call
display_divisible_by_4()
```

Alternative with return:

```python
def get_divisible_by_4():
    return [num for num in range(1, 101) if num % 4 == 0]

result = get_divisible_by_4()
```

```python
print(result)
```

Key Concepts:

- **Function definition**: def keyword usage
- **Range function**: 1 to 100 iteration
- **Modulus check**: num % 4 == 0 condition
- **List comprehension**: Alternative approach

Mnemonic

"Define-Range-Check-Display"

Question Question 3(a OR) [03 marks]

What is Repetition Control Statement? List it out.

Solution

Repetition Control Statements:
Key Concepts:

- **Repetition statements**: Execute code blocks repeatedly
- **Iteration control**: Different methods of looping
- **Loop variables**: Track iteration progress

Mnemonic

"For-While-Nested"

Question Question 3(b OR) [04 marks]

Differentiate break and continue statements.

Solution

Difference:
Code Example:

```python
# break example
for i in range(5):
    if i == 3:
        break
    print(i)  # Output: 0, 1, 2

# continue example
for i in range(5):
    if i == 2:
        continue
    print(i)  # Output: 0, 1, 3, 4
```

Mnemonic

"Break-Exit, Continue-Skip"

Question Question 3(c OR) [07 marks]

Write a user-define function which displays all even numbers from 1 to 100.

Solution

Code:

```python
def display_even_numbers():
    print("Even numbers from 1 to 100:")
    for num in range(2, 101, 2):
        print(num, end=" ")
    print()

# Alternative method
def display_even_alt():
    even_nums = []
    for num in range(1, 101):
        if num % 2 == 0:
            even_nums.append(num)
    print(even_nums)

# Function call
display_even_numbers()
```

Explanation:

- **Efficient range**: `range(2, 101, 2)` for even numbers
- **Modulus method**: Alternative checking with % 2 == 0
- **Function design**: Reusable code block

Mnemonic

"Range-Step-Even-Display"

Question Question 4(a) [03 marks]

Define Function. List out various types of Functions available in Python.

Solution

Function: Reusable block of code that performs specific task.
Function Types:
Benefits:

- **Code reusability**: Write once, use many times
- **Modularity**: Breaking complex problems into smaller parts
- **Parameters**: Input values to functions

Mnemonic

"Built-User-Lambda-Recursive"

Question Question 4(b) [04 marks]

Write short note on Scope of a variable.

Solution

Variable Scope:
Code Example:

```python
x = 10  # Global variable

def my_function():
    y = 20  # Local variable
    print(x)  # Access global
    print(y)  # Access local

my_function()
# print(y)  # Error: y not accessible
```

Key Concepts:

- **Variable accessibility**: Where variables can be used
- **LEGB rule**: Local, Enclosing, Global, Built-in

Mnemonic

"Local-Global-Builtin"

Question Question 4(c) [07 marks]

Write Python code which asks user for Main string and Substring and checks membership of a Substring in the Main String.

Solution

Code:

```python
def check_substring():
    main_string = input("Enter main string: ")
    substring = input("Enter substring: ")

    if substring in main_string:
        print(f"'{substring}' found in '{main_string}'")
        print(f"Position: {main_string.find(substring)}")
    else:
        print(f"'{substring}' not found in '{main_string}'")

# Enhanced version with case handling
def check_substring_enhanced():
    main_string = input("Enter main string: ")
    substring = input("Enter substring: ")

    if substring.lower() in main_string.lower():
        print("Substring found (case-insensitive)")
    else:
        print("Substring not found")

check_substring()
```

Explanation:

- **User interaction**: `input()` for string collection
- **Membership testing**: `in` operator usage
- **Case sensitivity**: Optional case handling

Mnemonic

"Input-Check-Report-Position"

Question Question 4(a OR) [03 marks]

What is Local variable and Global variable?

Solution

Comparison:
Example:

```python
global_var = 100  # Global

def function():
    local_var = 50  # Local
    print(global_var)  # Accessible
    print(local_var)   # Accessible

print(global_var)  # Accessible
# print(local_var)  # Error
```

Mnemonic

"Local-Limited, Global-Everywhere"

Question Question 4(b OR) [04 marks]

Explain any four built-in functions of Python.

Solution

Built-in Functions:
Additional Examples:

```python
# len() function
print(len([1, 2, 3, 4]))  # Output: 4

# type() function
print(type(3.14))  # Output: <class 'float'>

# input() function
age = input("Enter age: ")

# print() function
print("Your age is:", age)
```

Mnemonic

"Length-Type-Input-Print"

Question Question 4(c OR) [07 marks]

Write Python code which locates a substring in a given string.

Solution

Code:

```python
def locate_substring():
    main_string = input("Enter main string: ")
    substring = input("Enter substring to find: ")

    # Method 1: Using find()
    position = main_string.find(substring)
    if position != -1:
        print(f"Found at index: {position}")
    else:
        print("Substring not found")

    # Method 2: Using index() with exception handling
    try:
        position = main_string.index(substring)
        print(f"Located at index: {position}")
    except ValueError:
        print("Substring not found")

    # Method 3: Find all occurrences
    positions = []
    start = 0
    while True:
        pos = main_string.find(substring, start)
        if pos == -1:
            break
        positions.append(pos)
        start = pos + 1

    if positions:
        print(f"All positions: {positions}")

locate_substring()
```

Key Methods:

- **find() method**: Returns index or -1
- **index() method**: Returns index or raises exception
- **Multiple occurrences**: Loop to find all positions

Mnemonic

"Find-Index-Exception-Multiple"

Question Question 5(a) [03 marks]

Define String. List out various string operations.

Solution

String: Sequence of characters enclosed in quotes.
Operations:

Characteristics:

- **Immutable**: Strings cannot be changed after creation
- **Indexing**: Access individual characters
- **Methods**: Built-in functions for manipulation

Mnemonic

"Concat-Repeat-Slice-Length-Case"

Question Question 5(b) [04 marks]

How can we identify whether an element is a member of a list or not? Explain with a suitable example.

Solution

Methods:
Example:

```python
fruits = ["apple", "banana", "orange", "mango"]

# Using 'in' operator
if "apple" in fruits:
    print("Apple is available")

# Using 'not in' operator
if "grapes" not in fruits:
    print("Grapes not available")

# Using count() method
count = fruits.count("apple")
if count > 0:
    print(f"Apple found {count} times")
```

Mnemonic

"In-NotIn-Count for membership"

Question Question 5(c) [07 marks]

Write Python code that replaces a substring with another substring of a given string. Consider the given string as 'Welcome to GTU' and replace the substring 'GTU' with 'Gujarat Technological University'.

Solution

Code:

```python
def replace_substring():
    # Given string
    original = "Welcome to GTU"
    old_substring = "GTU"
    new_substring = "Gujarat Technological University"

    # Method 1: Using replace()
    result1 = original.replace(old_substring, new_substring)
    print(f"Original: {original}")
    print(f"Modified: {result1}")

    # Method 2: Manual replacement
    if old_substring in original:
        index = original.find(old_substring)
        result2 = original[:index] + new_substring + original[index + len(old_substring):]
        print(f"Manual method: {result2}")

    # Method 3: Replace all occurrences
    test_string = "GTU offers GTU degree from GTU"
    result3 = test_string.replace("GTU", "Gujarat Technological University")
    print(f"Multiple replacements: {result3}")

replace_substring()
```

Output:

```
Original: Welcome to GTU
Modified: Welcome to Gujarat Technological University
```

Key Points:

- **replace() method**: Built-in string function
- **Slicing method**: Manual string manipulation
- **All occurrences**: Replaces every instance

Mnemonic

"Find-Replace-Slice-All"

Question Question 5(a OR) [03 marks]

Define List. List out various list operations.

Solution

List: Ordered collection of items that can be modified.
Operations:
Features:

- **Mutable**: Lists can be changed after creation
- **Indexed**: Elements accessed by position
- **Dynamic**: Size can grow or shrink

Mnemonic

"Add-Remove-Access-Slice-Sort"

Question Question 5(b OR) [04 marks]

Write short note on String Slicing. Explain with suitable example.

Solution

String Slicing: Extracting parts of string using `[start:end:step]`.
Syntax:
Example:

```python
text = "Python Programming"

print(text[0:6])    # "Python"
print(text[7:])     # "Programming"
print(text[:6])     # "Python"
print(text[::2])    # "Pto rgamn"
print(text[::-1])   # "gnimmargorP nohtyP"
```

Mnemonic

"Start-End-Step-Reverse"