MSBTE(Mah.State Board of Tech.Edu) STUDY RESOURCES
(WWW.MSBTE.ENGG-INFO.WEBSITE)
Website dedicated to Diploma Engineering study Resources

Model Answer	Exam	SUM 2024
Subject: Programming with Python		
	SUB CODE	22616

Important Instructions to STUDENTS

1) *The model answer given here are prepared from the answers from the previously uploaded model answers by Board.*
2) *These model answers are not uploaded by the MSBTE official site but MSBTE study resources website prepared it for students. This model answer has question paper also inbuilt in it, no need to download it separate.*
3) *Please remembter that answers are not checked word to word but based on keywords which must be present in your answer*
4) *The model answer and the answer written by candidate may vary but the examiner may tryto assess the understanding level of the candidate*
5) *While assessing figures, examiner may give credit for principal components indicated in thefigure. The figures drawn bycandidate and model answer may vary. The examiner may give credit for anyequivalent figure drawn*
6) *For programming language papers, credit may be given to any other program based on equivalentconcept*
7) *Students are advised to prepare all the syllabus from recommended book and use these model answers for the purpose of tests.*

Question 1: Attempt any FIVE of the following

---

a) Enlist any four data structures used in Python.

Answer:

Python provides several built-in data structures to organize and manage data efficiently:

1. **List**:
   - **Type**: Ordered, mutable (can be modified).
   - **Example**: `numbers = [1, 2, 3, "apple", 5.5]`.
   - **Use Case**: Storing collections of items where order and modifications are required.
2. **Tuple**:
   - **Type**: Ordered, immutable (cannot be modified).
   - **Example**: `coordinates = (10, 20, 30)`.
   - **Use Case**: Storing fixed data like coordinates or constants.
3. **Dictionary**:
   - **Type**: Unordered, mutable collection of key-value pairs.
   - **Example**: `student = {"name": "Alice", "age": 21}`.
   - **Use Case**: Fast lookups using unique keys (e.g., JSON-like data).
4. **Set**:
   - **Type**: Unordered, mutable collection of unique elements.
   - **Example**: `unique_ids = {101, 102, 103}`.
   - **Use Case**: Removing duplicates or checking membership.

b) Give membership operators in Python.

Answer:

Membership operators check if a value exists in a sequence (e.g., list, string, set):

1. `in`:
   - Returns `True` if the value is found.
   - **Example**:

```python
fruits = ["apple", "banana", "mango"]
print("apple" in fruits)  # Output: True
```

2. `not in`:
   - Returns `True` if the value is **not** found.
   - **Example**:

```python
vowels = {'a', 'e', 'i', 'o', 'u'}
print('z' not in vowels)  # Output: True
```

c) Write syntax for a method to sort a list.

Answer:

Use the `sort()` method to sort a list in-place (modifies the original list):

```python
# Syntax:
list_name.sort(key=None, reverse=False)
```

Examples:

1. **Ascending Order**:

```python
numbers = [3, 1, 4, 2]
numbers.sort()
print(numbers)  # Output: [1, 2, 3, 4]
```

2. **Descending Order**:

```python
numbers.sort(reverse=True)
print(numbers)  # Output: [4, 3, 2, 1]
```

3. **Custom Sorting (e.g., by string length)**:

```python
words = ["apple", "kiwi", "banana"]
words.sort(key=lambda x: len(x))
print(words)  # Output: ["kiwi", "apple", "banana"]
```

d) Write use of matplotlib package in Python.

Answer:

Matplotlib is a plotting library for creating static, interactive, and animated visualizations. Key use include:

1. **Line Plots**: Visualize trends over time (e.g., stock prices).

```python
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [10, 15, 13, 18]
plt.plot(x, y, marker='o', linestyle='--')
plt.title("Monthly Sales")
plt.xlabel("Month")
plt.ylabel("Revenue ($)")
plt.show()
```

2. **Bar Charts**: Compare categorical data (e.g., sales by region).
3. **Histograms**: Display frequency distributions (e.g., exam scores).
4. **Scatter Plots**: Analyze relationships between variables.

e) What is data abstraction and data hiding?

Answer:

- **Data Abstraction**:
  - **Definition**: Hiding complex implementation details while exposing only essential features to the user.
  - **Example**: A `Car` class with a method `start_engine()`, where the internal mechanics are hidden.
- **Data Hiding**:
  - **Definition**: Restricting direct access to an object's internal data to prevent unintended modifications.
  - **Python Implementation**:
    - **Single Underscore (`_variable`)**: Convention for "protected" variables (not enforced).
    - **Double Underscore (`__variable`)**: Name mangling to make variables "private".
  - **Example**:

```python
class BankAccount:
    def __init__(self):
        self.__balance = 1000  # Private variable
    def get_balance(self):
        return self.__balance  # Access via method
```

f) Write the syntax of fopen.

Answer:

In Python, the correct function is `open()`, not `fopen()`.

Syntax:

```python
file_object = open("filename", "mode", buffering=-1, encoding=None)
```

Parameters:

- `filename`: Name/path of the file (e.g., `"data.txt"`).
- `mode`: Specifies the operation (default is `"r"` for read). Common modes:
  - `"r"`: Read mode.
  - `"w"`: Write mode (overwrites existing file).
  - `"a"`: Append mode (adds to the end).
  - `"b"`: Binary mode (e.g., `"rb"`).

g) What is a dictionary?

Answer:

A **dictionary** is an unordered, mutable collection of **key-value pairs**.

Key Features:

- **Keys**: Must be unique and immutable (e.g., strings, numbers).
- **Values**: Can be of any data type (e.g., lists, other dictionaries).
- **Syntax**: `{key1: value1, key2: value2}`.

Question 2: Attempt any THREE of the following

a) Explain with example:

i) Indentation

ii) Variables

Answer:

i) Indentation:

- Python uses indentation (spaces/tabs) to define code blocks instead of braces `{}`.
- **Example**:

```python
if 10 > 5:
    print("True")  # Indented block
    print("This is inside the if statement")
print("Outside the block")  # Unindented
```

ii) Variables:

- Variables store data values and do not require explicit type declaration.
- **Example**:

```python
name = "Alice"    # String variable
age = 25          # Integer variable
marks = 89.5      # Float variable
```

b) Print the following pattern using loop:

```
1 0 1 0 1 0 1
1 0 1 0 1
1 0 1
1
```

Answer:

```python
rows = 4
for i in range(rows, 0, -1):
    for j in range(2*i - 1):
        print("1" if j % 2 == 0 else "0", end=" ")
    print()
```

Explanation:

- The outer loop (`rows`) controls the number of rows (4 to 1).
- The inner loop prints alternating `1` and `0` based on `j % 2` .

c) Write python program to perform following operations on set:

i) Create set of five elements

ii) Access set elements

iii) Update set by adding one element

iv) Remove one element from set

Answer:

python

```python
# i) Create set
my_set = {1, 2, 3, 4, 5}
print("Original Set:", my_set)

# ii) Access elements
print("\nAccessing Elements:")
for element in my_set:
    print(element, end=" ")

# iii) Update set
my_set.add(6)
print("\n\nAfter Adding 6:", my_set)

# iv) Remove element
my_set.remove(3)
print("After Removing 3:", my_set)
```

Output:

```
Original Set: {1, 2, 3, 4, 5}
Accessing Elements: 1 2 3 4 5
After Adding 6: {1, 2, 3, 4, 5, 6}
After Removing 3: {1, 2, 4, 5, 6}
```

d) Describe following Standard Packages:

i) Numpy

ii) Pandas

Answer:

i) Numpy:

- A library for numerical computing with support for arrays and matrices
- **Example**:

```python
import numpy as np
arr = np.array([1, 2, 3])  # Create a 1D array
print(arr * 2)  # Output: [2 4 6]
```

ii) Pandas:

- A library for data manipulation and analysis using DataFrame structure
- **Example**:

```python
import pandas as pd
data = {"Name": ["Alice", "Bob"], "Age": [25, 30]}
df = pd.DataFrame(data)
print(df)
```

Question 3: Attempt any THREE of the following

---

a) What is the output of the following program?

```python
dict1 = {'Google': 1, 'Facebook': 2, 'Microsoft': 3}
dict2 = {'GFG': 1, 'Microsoft': 2, 'Youtube': 3}
dict1.update(dict2)
for key, value in dict1.items():
    print(key, value)
```

Answer:

Output:

```
Google 1
Facebook 2
Microsoft 2
GFG 1
Youtube 3
```

b) Write a Python program that takes a number and checks whether it is a palindrome.

Answer:

Method 1: Using string slicing

```python
num = input("Enter a number: ")
if num == num[::-1]:
    print(f"{num} is a palindrome.")
else:
    print(f"{num} is not a palindrome.")
```

Method 2: Using arithmetic operations

```python
num = int(input("Enter a number: "))
original = num
reverse = 0
while num > 0:
    digit = num % 10
    reverse = reverse * 10 + digit
    num = num // 10
print(f"{original} is a palindrome." if original == reverse else f"{original} is not a palin
e.")
```

c) Write a Python program to create a user-defined module that will ask your program nar display the name of the program.

Answer:

Step 1: Create the module (`program_name.py`)

python

```python
def get_program_name():
    name = input("Enter the name of your program: ")
    return name

def display_program_name():
    program_name = get_program_name()
    print(f"Program Name: {program_name}")
```

Step 2: Use the module in the main program

python

```python
import program_name
program_name.display_program_name()
```

Output:

```
Enter the name of your program: Data Analyzer
Program Name: Data Analyzer
```

d) List data types used in Python. Explain any two with example.

Answer:

Python Data Types:
`int`, `float`, `str`, `list`, `tuple`, `dict`, `set`, `bool`.

Explanation with Code Examples:

1. `int` **(Integer)**:
   - Represents whole numbers (positive, negative, or zero).
   - **Example**:

```python
age = 25
temperature = -10
print(type(age))  # Output: <class 'int'>
```

2. `str` **(String)**:
   - Represents textual data enclosed in single (`' '`) or double quotes (`" "`).
   - **Example**:

```python
greeting = "Hello, World!"
address = '123 Main Street'
print(type(greeting))  # Output: <class 'str'>
```

Question 4: Attempt any THREE of the following

a) Compare list and tuple (any four points).

Answer:

List	Tuple
Mutable (e.g., `[1, 2]`)	Immutable (e.g., `(1, 2)`)
Uses `append()`, `pop()`	No modification methods
More memory consumption	Less memory consumption
Syntax: Square brackets	Syntax: Parentheses

b) Write a Python program to find the sum of digits in a number.

Answer:

```python
num = int(input("Enter a number: "))
total = 0
while num > 0:
    total += num % 10
    num = num // 10
print("Sum of digits:", total)
```

Output: `Enter a number: 123` → `Sum of digits: 6`.

c) Write a function to calculate uppercase and lowercase letters in a string.

Answer:

```python
def count_letters(s):
    upper = sum(1 for c in s if c.isupper())
    lower = sum(1 for c in s if c.islower())
    print(f"Uppercase: {upper}, Lowercase: {lower}")

count_letters("Hello World")  # Output: Uppercase: 2, Lowercase: 8
```

d) Write a Python program to create class Student with roll-no and display contents.

Answer:

```python
class Student:
    def __init__(self, roll_no):
        self.roll_no = roll_no
    def display(self):
        print("Roll No:", self.roll_no)

s = Student(101)
s.display()  # Output: Roll No: 101
```

e) Explain: i) open() ii) write() functions with example.

Answer:

1. `open()` : Opens a file.

```python
file = open("data.txt", "r")  # Read mode
```

2. `write()` : Writes data to a file.

```python
with open("output.txt", "w") as f:
    f.write("Hello World")
```

Q.5 Attempt any two

a) Write the output for the following if the variable `course = "Python"`:

```python
>> course [ : 3 ]
>> course [ 3 : ]
>> course [ 2 : 2 ]
>> course [ : ]
>> course [ -1 ]
>> course [ 1 ]
```

Answer:

```python
>> course[:3]    # Output: 'Pyt'
>> course[3:]    # Output: 'hon'
>> course[2:2]   # Output: '' (empty string)
>> course[:]     # Output: 'Python'
>> course[-1]    # Output: 'n'
>> course[1]     # Output: 'y'
```

Explanation:

- `course[:3]`: Slices from index `0` to `2` (excludes index `3`).
- `course[3:]`: Slices from index `3` to the end.
- `course[2:2]`: Start and end index are the same → no characters.
- `course[:]`: Returns the full string.
- `course[-1]`: Accesses the last character (`n`).
- `course[1]`: Accesses the second character (`y`).

b) Write a Python program to generate five random integers between 10 and 50 using NumPy library.

Answer:

```python
import numpy as np

# Generate 5 random integers between 10 and 50 (exclusive of 50)
random_ints = np.random.randint(low=10, high=50, size=5)
print("Random Integers:", random_ints)
```

Output Example:

```
Random Integers: [12 34 25 45 30]
```

Explanation:

- `np.random.randint()`: Generates integers between `low` (inclusive) and `high` (exclusive).
- `size=5`: Specifies the number of integers to generate.

c) Write a Python program to create a class 'Diploma' having a method 'getdiploma' that prints 'I got a diploma'. It has two subclasses namely 'CO' and 'IF' each having a method with the same name that prints 'I am with CO diploma' and 'I am with IF diploma' respectively. Call the method by creating an object of each of the three classes.

Answer:

```python
# Parent class
class Diploma:
    def getdiploma(self):
        print("I got a diploma")

# Subclass CO
class CO(Diploma):
    def getdiploma(self):
        print("I am with CO diploma")

# Subclass IF
class IF(Diploma):
    def getdiploma(self):
        print("I am with IF diploma")

# Create objects
d = Diploma()
c = CO()
i = IF()

# Call methods
d.getdiploma()  # Output: I got a diploma
c.getdiploma()  # Output: I am with CO diploma
i.getdiploma()  # Output: I am with IF diploma
```

Explanation:

- **Method Overriding**: Subclasses `CO` and `IF` override the `getdiploma()` method of the parent `Diploma`.
- **Polymorphism**: Objects of different classes invoke their respective `getdiploma()` methods.

Answer:

Multiple Inheritance: A class can inherit attributes and methods from **more than one parent class**.

Program:

```python
# Parent Class 1
class Parent1:
    def method1(self):
        print("Method from Parent1")

# Parent Class 2
class Parent2:
    def method2(self):
        print("Method from Parent2")

# Child Class inheriting from Parent1 and Parent2
class Child(Parent1, Parent2):
    def child_method(self):
        print("Method from Child")

# Create object
obj = Child()

# Access methods from all classes
obj.method1()      # Output: Method from Parent1
obj.method2()      # Output: Method from Parent2
obj.child_method() # Output: Method from Child
```

Explanation:

- **Child** inherits from both **Parent1** and **Parent2**.
- The child class can access methods from all parent classes.

b) Write a Python program to create a user-defined exception that checks whether the passwor correct or not.

Answer:

```python
# Custom exception class
class PasswordError(Exception):
    pass

def validate_password(input_password):
    correct_password = "secret123"
    if input_password != correct_password:
        raise PasswordError("Incorrect Password! Access denied.")
    else:
        print("Access granted!")

# Test the exception
try:
    user_input = input("Enter password: ")
    validate_password(user_input)
except PasswordError as e:
    print(e)
```

- **Output Examples**:

  - **Correct Password**:

    ```
    Enter password: secret123
    Access granted!
    ```

  - **Incorrect Password**:

    ```
    Enter password: hello
    Incorrect Password! Access denied.
    ```

Explanation:

- **PasswordError** is a custom exception raised when passwords do not match.
- **try-except** block handles the exception gracefully.

c) Describe various modes of file object. Explain any three in detail.

Answer:

File Modes:

- `r` (Read), `w` (Write), `a` (Append), `r+` (Read+Write), `b` (Binary), `x` (Exclusive Creation).

  - `r` **(Read Mode)**:
    - **Purpose**: Opens a file for reading (default mode).
    - **Example**:

```python
file = open("data.txt", "r")
content = file.read()
```

    - **Note**: File must exist; else, raises `FileNotFoundError`.
  - `w` **(Write Mode)**:
    - **Purpose**: Opens a file for writing. Overwrites existing content.
    - **Example**:

```python
file = open("output.txt", "w")
file.write("New content")
```

    - **Note**: Creates a new file if it doesn't exist.
  - `a` **(Append Mode)**:
    - **Purpose**: Opens a file for appending data to the end.
    - **Example**:

```python
file = open("logs.txt", "a")
file.write("\nNew log entry")
```

    - **Note**: Preserves existing content; adds new data at the end.

CREATED BY

MSBTE STUDY RESOURCES

www.msbte.engg-info.website