Assignment

Assignment Topic:

Course Code: CSE 221

Course Title: Object Oriented Programming II

Submitted To

Name: Nasima Islam Bithi (NIB)

Designation: Lecturer

Department of CSE

Daffodil International University

Submitted By

Name: Afjal

ID:0242220005101730

Section:63_M

Department of CSE

Daffodil International University

Date of Submission: 20 September 2024

1.Dictionary

```
def manage_courses():
    course = {
        "CSE101": {
            "Course name": "Introduction to programming",
            "Credits": 3,
            "Instructor": "Dr. Alice",
        },
        "CSE102": {
            "Course name": "Data Structures",
            "Credits": 4,
            "Instructor": "Dr. Bob",
        },
        "CSE103": {
            "Course name": "Database Systems",
            "Credits": 3,
            "Instructor": "Dr. Carol",
        },
    }
    # Update the instructor's name for CSE102 to "Dr. Bob Jr."
    course["CSE102"]["Instructor"] = "Dr. Bob Jr."
```

```
    # Add new course
    course["CSE104"] = {
        "Course name": "Algorithms",
        "Credits": 4,
        "Instructor": "Dr. Dave",
    }
    if "CSE101" in course:
        del course["CSE101"]
    for course_code, details in course.items():
        for key, value in details.items():
            print(f"{key}: {value}")
        print()
manage_courses()
```

Output

```
Course name: Data Structures
Credits: 4
Instructor: Dr. Bob Jr.
Course name: Database Systems
Credits: 3
Instructor: Dr. Carol
Course name: Algorithms
```

Credits: 4

Instructor: Dr. Dave

[Done] exited with code=0 in 0.066 seconds

2.String

```
def process_string():
    sentence = "Learning Python is fun and rewarding."
    # "Python is fun"
    substring = sentence[-28:-15]
    print(f"Extracted substring: {substring}")
    modified_sentence = sentence.replace("rewarding", "exciting")
    print(f"Modified sentence: {modified_sentence}")

    position = modified_sentence.find("exciting") + len("exciting")
    final_sentence = (
        modified_sentence[:position]
        + " Keep practicing!"
        + modified_sentence[position:]
    )
    print(f"Sentence after inserting: {final_sentence}")
    capitalized_sentence = final_sentence.title()
```

```
    print(f"Final capitalized sentence: {capitalized_sentence}")
process_string()
```

Output

Extracted substring: Python is fun

Modified sentence: Learning Python is fun and exciting.

Sentence after inserting: Learning Python is fun and exciting Keep practicing!.

Final capitalized sentence: Learning Python Is Fun And Exciting Keep Practicing!.

3.List

```
def manage_customers():

    customers = ["Alice", "Bob", "Charlie", "David", "Eve"]
    third_customer = customers[2]
    print(f"Third customer: {third_customer}")
    customers[1] = "Ben"
    customers.append("Frank")
    customers.remove("David")
    customers.sort()
```

```
    print(f"Final sorted customer list: {customers}")
manage_customers()
```

Output

```
Third customer: Charlie
Final sorted customer list: ['Alice', 'Ben', 'Charlie', 'Eve', 'Frank']
```

4.Control flow

```
def categorize_grades(grades):
    print("Grade Categories:")
    for score in grades:
        if score > 80:
            grade = "A"
        elif 60 <= score <= 80:
            grade = "B"
        elif 40 <= score <= 60:
            grade = "C"
        else:
            grade = "F"
        print(f"Score: {score} - Grade: {grade}")
```

```
def boost_grades(grades):
    boosted_grades = list(map(lambda x: x * 1.05, grades))
    return boosted_grades

def find_grades_above_90(boosted_grades):
    above_90 = list(filter(lambda x: x > 90, boosted_grades))
    return above_90



grades = [85, 78, 92, 45, 33, 67, 88, 41]
categorize_grades(grades)
boosted_grades = boost_grades(grades)
print("\nBoosted Grades:")
print(boosted_grades)
grades_above_90 = find_grades_above_90(boosted_grades)
print("\nBoosted Grades Above 90:")
print(grades_above_90)
```

Output

```
Grade Categories:
Score: 85 - Grade: A
```

Score: 78 - Grade: B

Score: 92 - Grade: A

Score: 45 - Grade: C

Score: 33 - Grade: F

Score: 67 - Grade: B

Score: 88 - Grade: A

Score: 41 - Grade: C

Boosted Grades:

[89.25, 81.9, 96.6, 47.25, 34.65, 70.35, 92.4, 43.05]

Boosted Grades Above 90:

[96.6, 92.4]

5.Tuple & Set

```
books = (
    ("To Kill a Mockingbird", "Harper Lee", 1960),
    ("1984", "George Orwell", 1949),
    ("The Great Gatsby", "F. Scott Fitzgerald", 1925),
)
tags = {"classic", "dystopian", "novel", "literature"}
second_book_author = books[1][1]
print("Author of the second book:", second_book_author)
```

```
new_book = ("Brave New World", "Aldous Huxley", 1932)
books = books + (new_book,)
print("\nUpdated books tuple:", books)
title, author, year = books[2]
print("\nDetails of the third book:")
print("Title:", title)
print("Author:", author)
print("Year:", year)

print("\nBook Titles:")
for book in books:
    print(book[0])
tags.add("sci-fi")
print("\nUpdated tags set:", tags)
tags.remove("novel")
print("\nTags set after removing 'novel':", tags)
```

Output

Author of the second book: George Orwell
Updated books tuple: (('To Kill a Mockingbird', 'Harper Lee', 1960), ('1984', 'George Orwell', 1949), ('The Great Gatsby', 'F. Scott Fitzgerald', 1925), ('Brave New World', 'Aldous Huxley', 1932))

Details of the third book:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
Year: 1925

Book Titles:
To Kill a Mockingbird
1984
The Great Gatsby
Brave New World

Updated tags set: {'classic', 'novel', 'literature', 'sci-fi', 'dystopian'}

Tags set after removing 'novel': {'classic', 'literature', 'sci-fi', 'dystopian'}