..... ASSIGNMENT .....

Course Code : CSE221

Course Title : Object Oriented Programming II

Submitted to:

Teacher Name : Ms. Nasima Islam Bithi

Department : CSE

Daffodil International University.

Submitted by:

Name : TITAS SARKER

ID : 0242220005101**864**

Section : 63_N

Department : CSE

Date of Submission: 22 .09 .2024

01.Dictionary

```
# 1. Dictionary

def create_and_update_courses():
  # Create a dictionary to store courses
  courses = {
      "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
        courses["CSE102"]["Instructor"] = "Dr. Bob Jr."

        # Add a new course
        courses["CSE104"] = {
            "Course name": "Algorithms",
            "Credits": 4,
            "Instructor": "Dr. Dave"
        }

        # Remove the course CSE101
        del courses["CSE101"]

        # Loop through the dictionary and print course details
        for course_code, course_details in courses.items():
            print(f"Course Code: {course_code}")
            print(f"Course Name: {course_details['Course name']}")
            print(f"Credits: {course_details['Credits']}")
            print(f"Instructor: {course_details['Instructor']}")
```

```
            print()

    if __name__ == "__main__":
        create_and_update_courses()

```

Output:

```
Course Code: CSE102
Course Name: Data Structures
Credits: 4
Instructor: Dr. Bob Jr.

Course Code: CSE103
Course Name: Database Systems
Credits: 3
Instructor: Dr. Carol

Course Code: CSE104
Course Name: Algorithms
Credits: 4
Instructor: Dr. Dave
```

02. String

```
# 2. String
def process_string():
    sentence = "Learning Python is fun and rewarding."

    # a. Extract the substring "Python is fun" using negative slicing
    substring = sentence[-28:-14]
    print(f"Extracted substring: {substring}")

    # b. Modify the original string by replacing "rewarding" with "exciting"
        modified_sentence = sentence.replace("rewarding", "exciting")
        print(f"Modified sentence: {modified_sentence}")

        # c. Insert " Keep practicing!" after "exciting"
        position = modified_sentence.find("exciting") + len("exciting")
        final_sentence = modified_sentence[:position] + " Keep practicing!" + modified_sentence[position:]
        print(f"Sentence after insertion: {final_sentence}")

        # d. Capitalize the first letter of each word in the final output
```

```
        capitalized_sentence = final_sentence.title()
        print(f"Capitalized final sentence: {capitalized_sentence}")

    # Call the function to see the result
    process_string()
```

Output:

```
Extracted substring: Python is fun
Modified sentence: Learning Python is fun and exciting.
Sentence after insertion: Learning Python is fun and exciting Keep practicing!.
Capitalized final sentence: Learning Python Is Fun And Exciting Keep Practicing!.
```

03. List

```
#03. List

customers = ["Alice", "Bob", "Charlie", "David", "Eve"]

    # a. Access the third customer in the list
     print(customers[2])

    # b. Change the name of the second customer to "Ben"
     customers[1] = "Ben"

     # c. Add a new customer named "Frank" to the end of the list
     customers.append("Frank")

     # d. Remove the customer "David" from the list
     customers.remove("David")

     # e. Sort the customer names alphabetically and print the final list
     customers.sort()
     print(customers)
```

Output:

```
Charlie
['Alice', 'Ben', 'Charlie', 'Eve', 'Frank']
```

04. Control Flow

```
1. # 04
2. grades = [85, 78, 92, 45, 33, 67, 88, 41]
3.
4. # a. Categorize each student's grade
5. print("Grade Categories:")
6. for grade in grades:
7.     if grade > 80:
8.         print(f"Score: {grade} - Grade: A")
9.     elif grade >= 60:
10.            print(f"Score: {grade} - Grade: B")
11.        elif grade >= 40:
12.            print(f"Score: {grade} - Grade: C")
13.        else:
14.            print(f"Score: {grade} - Grade: F")
15.
16.     # b. Boost grades by 5%
17.     def boost_grades(grade):
18.         return grade * 1.05
19.
20.     boosted_grades = list(map(boost_grades, grades))
21.     print("\nBoosted Grades:")
22.     print(boosted_grades)
23.
24.     # c. Find boosted grades above 90
25.     above_90 = list(filter(lambda grade: grade > 90,
    boosted_grades))
26.     print("\nBoosted Grades Above 90:")
27.     print(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, 96.60000000000001, 47.25, 34.65, 70.35000000000001, 92.4, 43.050000000000004]

Boosted Grades Above 90:
[96.60000000000001, 92.4]
```

5. Tuple & Set

```
1.
2. # Given Initial data
3. books = (
4.     ("To Kill a Mockingbird", "Harper Lee", 1960),
5.     ("1984", "George Orwell", 1949),
6.     ("The Great Gatsby", "F. Scott Fitzgerald", 1925)
7. )
8. tags = {"classic", "dystopian", "novel", "literature"}
9.
10.      # a. Access the second book's author from the books tuple...
11.      print("The author of the second book is:", books[1][1])
12.
13.      # b. Add a new record for "Brave New World" by Aldous Huxley,
   published in 1932
14.      # we need a new tuple with the new book added.
15.      new_book = ("Brave New World", "Aldous Huxley", 1932)
16.      books = books + (new_book,)  # Concatenate the new book as a
   tuple
17.      print("\nUpdated books tuple:")
18.      print(books)
19.
20.      # c. Unpack the details of the third book
21.      title, author, year = books[2]
22.      print("\nDetails of the third book:")
23.      print(f"Title: {title}")
24.      print(f"Author: {author}")
25.      print(f"Year: {year}")
26.
27.      # d. Loop through the books tuple and print each book's title
28.      print("\nBook Titles:")
29.      for book in books:
30.          print(book[0])
31.
32.      # e. Add a new tag sci-fi
33.      tags.add("sci-fi")
34.      print("\nUpdated tags set with 'sci-fi':")
35.      print(tags)
36.
37.      # f. Remove the tag novel
38.      tags.remove("novel")
39.      print("\nUpdated tags set after removing 'novel':")
40.      print(tags)
```

Output:

```
The author of the second book is: 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 with 'sci-fi':
{'dystopian', 'classic', 'novel', 'literature', 'sci-fi'}

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