# 

**………………….…… 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 :** 024222000510**1864**

**Section :** 63\_N

**Department :** CSE

**Date of Submission:**  22 .09 .2024

**01.Dictionary**

1. # 1. Dictionary
2. def create\_and\_update\_courses():
3. # Create a dictionary to store courses
4. courses = {
5. "CSE101": {
6. "Course name": "Introduction to Programming",
7. "Credits": 3,
8. "Instructor": "Dr. Alice"
9. },
10. "CSE102": {
11. "Course name": "Data Structures",
12. "Credits": 4,
13. "Instructor": "Dr. Bob"
14. },
15. "CSE103": {
16. "Course name": "Database Systems",
17. "Credits": 3,
18. "Instructor": "Dr. Carol"
19. }
20. }
21. # Update the instructor's name for CSE102
22. courses["CSE102"]["Instructor"] = "Dr. Bob Jr."
23. # Add a new course
24. courses["CSE104"] = {
25. "Course name": "Algorithms",
26. "Credits": 4,
27. "Instructor": "Dr. Dave"
28. }
29. # Remove the course CSE101
30. del courses["CSE101"]
31. # Loop through the dictionary and print course details
32. for course\_code, course\_details in courses.items():
33. print(f"Course Code: {course\_code}")
34. print(f"Course Name: {course\_details['Course name']}")
35. print(f"Credits: {course\_details['Credits']}")
36. print(f"Instructor: {course\_details['Instructor']}")
37. print()
38. if \_\_name\_\_ == "\_\_main\_\_":
39. create\_and\_update\_courses()

**Output:**

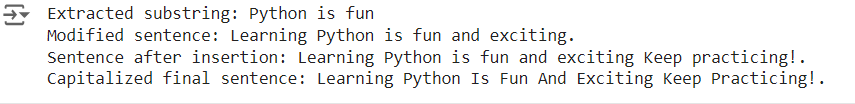
**A screenshot of a computer

Description automatically generated**

**02. String**

1. # 2. String
2. def process\_string():
3. sentence = "Learning Python is fun and rewarding."
4. # a. Extract the substring "Python is fun" using negative slicing
5. substring = sentence[-28:-14]
6. print(f"Extracted substring: {substring}")
7. # b. Modify the original string by replacing "rewarding" with "exciting"
8. modified\_sentence = sentence.replace("rewarding", "exciting")
9. print(f"Modified sentence: {modified\_sentence}")
10. # c. Insert " Keep practicing!" after "exciting"
11. position = modified\_sentence.find("exciting") + len("exciting")
12. final\_sentence = modified\_sentence[:position] + " Keep practicing!" + modified\_sentence[position:]
13. print(f"Sentence after insertion: {final\_sentence}")
14. # d. Capitalize the first letter of each word in the final output
15. capitalized\_sentence = final\_sentence.title()
16. print(f"Capitalized final sentence: {capitalized\_sentence}")
17. # Call the function to see the result
18. process\_string()

**Output:**

****

**03. List**

1. #03. List
2. customers = ["Alice", "Bob", "Charlie", "David", "Eve"]
3. # a. Access the third customer in the list
4. print(customers[2])
6. # b. Change the name of the second customer to "Ben"
7. customers[1] = "Ben"
8. # c. Add a new customer named "Frank" to the end of the list
9. customers.append("Frank")
10. # d. Remove the customer "David" from the list
11. customers.remove("David")
12. # e. Sort the customer names alphabetically and print the final list
13. customers.sort()
14. print(customers)

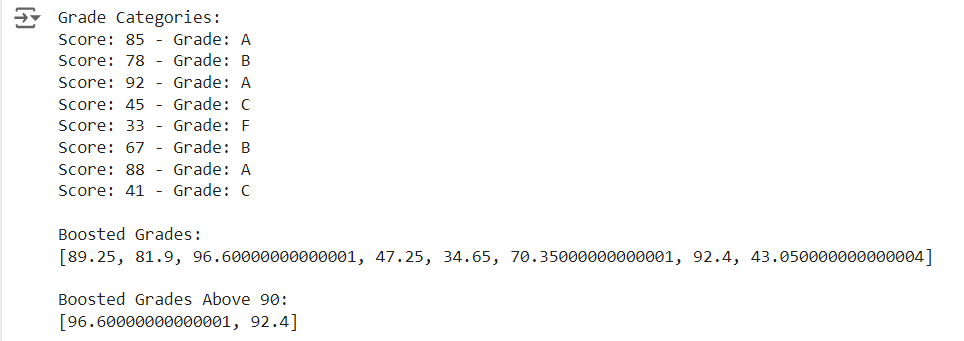
**Output:**

****

**04. Control Flow**

1. # 04
2. grades = [85, 78, 92, 45, 33, 67, 88, 41]
3. # a. Categorize each student's grade
4. print("Grade Categories:")
5. for grade in grades:
6. if grade > 80:
7. print(f"Score: {grade} - Grade: A")
8. elif grade >= 60:
9. print(f"Score: {grade} - Grade: B")
10. elif grade >= 40:
11. print(f"Score: {grade} - Grade: C")
12. else:
13. print(f"Score: {grade} - Grade: F")
14. # b. Boost grades by 5%
15. def boost\_grades(grade):
16. return grade \* 1.05
17. boosted\_grades = list(map(boost\_grades, grades))
18. print("\nBoosted Grades:")
19. print(boosted\_grades)
20. # c. Find boosted grades above 90
21. above\_90 = list(filter(lambda grade: grade > 90, boosted\_grades))
22. print("\nBoosted Grades Above 90:")
23. print(above\_90)

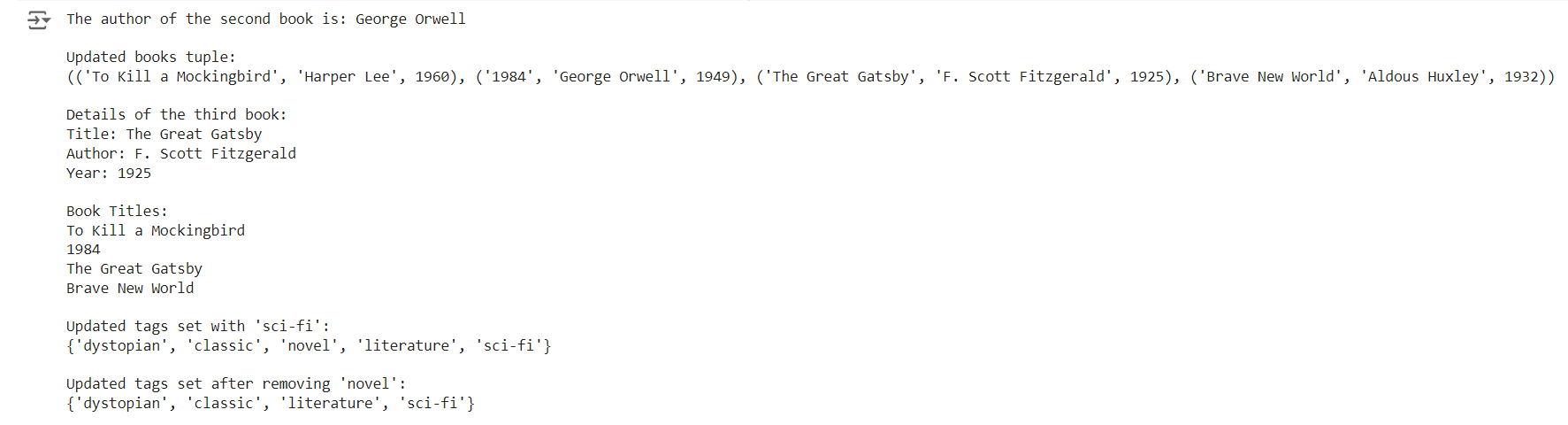
**Output:**

****

**5. Tuple & Set**

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. # a. Access the second book's author from the books tuple...
10. print("The author of the second book is:", books[1][1])
11. # b. Add a new record for "Brave New World" by Aldous Huxley, published in 1932
12. # we need a new tuple with the new book added.
13. new\_book = ("Brave New World", "Aldous Huxley", 1932)
14. books = books + (new\_book,)  # Concatenate the new book as a tuple
15. print("\nUpdated books tuple:")
16. print(books)
17. # c. Unpack the details of the third book
18. title, author, year = books[2]
19. print("\nDetails of the third book:")
20. print(f"Title: {title}")
21. print(f"Author: {author}")
22. print(f"Year: {year}")
23. # d. Loop through the books tuple and print each book's title
24. print("\nBook Titles:")
25. for book in books:
26. print(book[0])
27. # e. Add a new tag sci-fi
28. tags.add("sci-fi")
29. print("\nUpdated tags set with 'sci-fi':")
30. print(tags)
31. # f. Remove the tag novel
32. tags.remove("novel")
33. print("\nUpdated tags set after removing 'novel':")
34. print(tags)

**Output:**

****