PYTHON IS FOR EVERYONE

Tutorial 8:

PYTHON PROGRAMMING - DICTIONARIES AND DICTIONARY OPERATIONS IN GOOGLE COLAB

Jeff Gentry

@www.linkedin.com/in/jefferycharlesgentry

Objectives

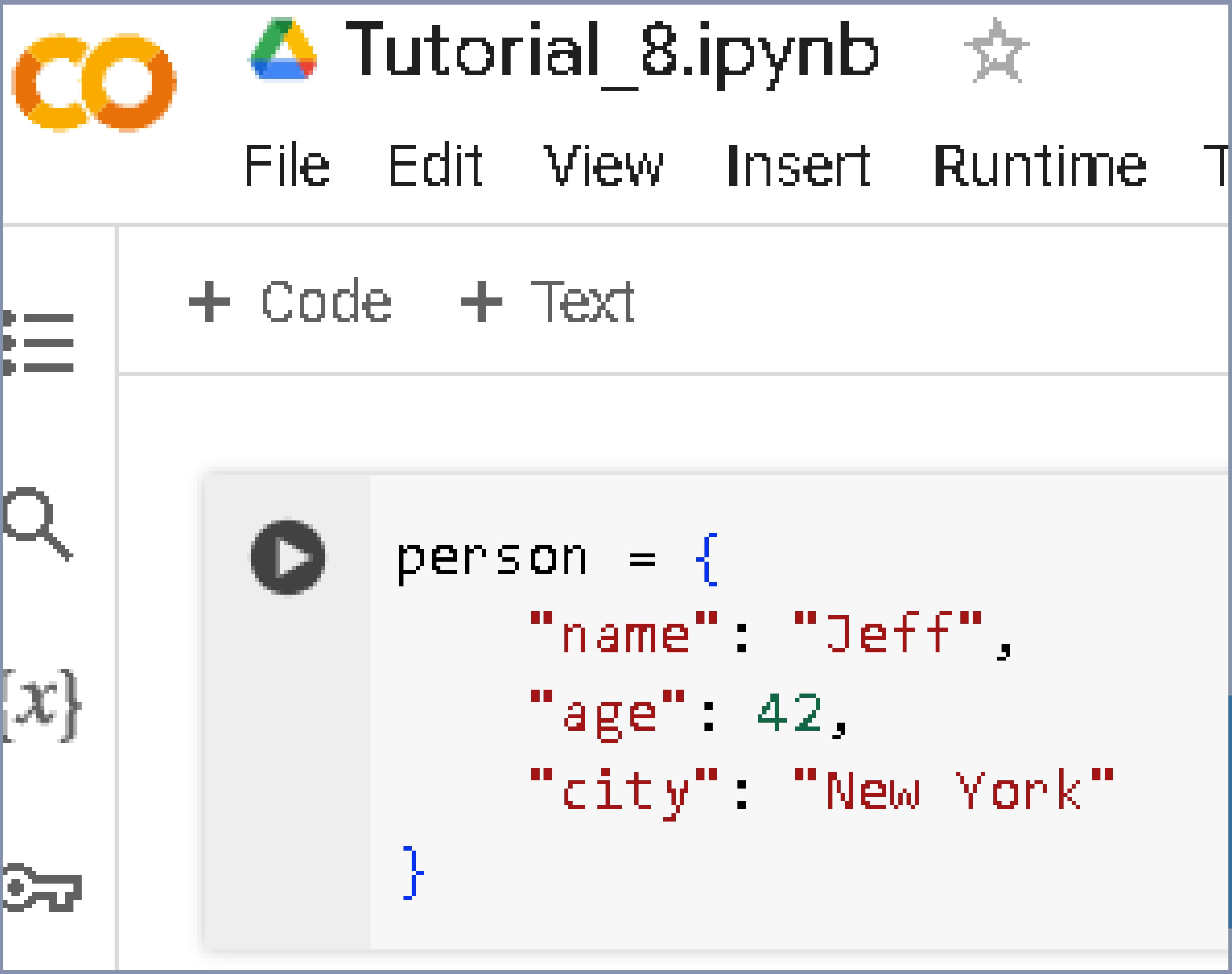
- Understand what dictionaries are and how to create them.
- Learn how to access, modify, and manipulate dictionary elements.
- Explore common dictionary methods and operations.
- Practice working with dictionaries.

What is a Dictionary?

A dictionary is a collection of key-value pairs. Each key is unique and is used to access its corresponding value. Dictionaries are unordered, meaning the items do not have a defined order.

Creating a Dictionary

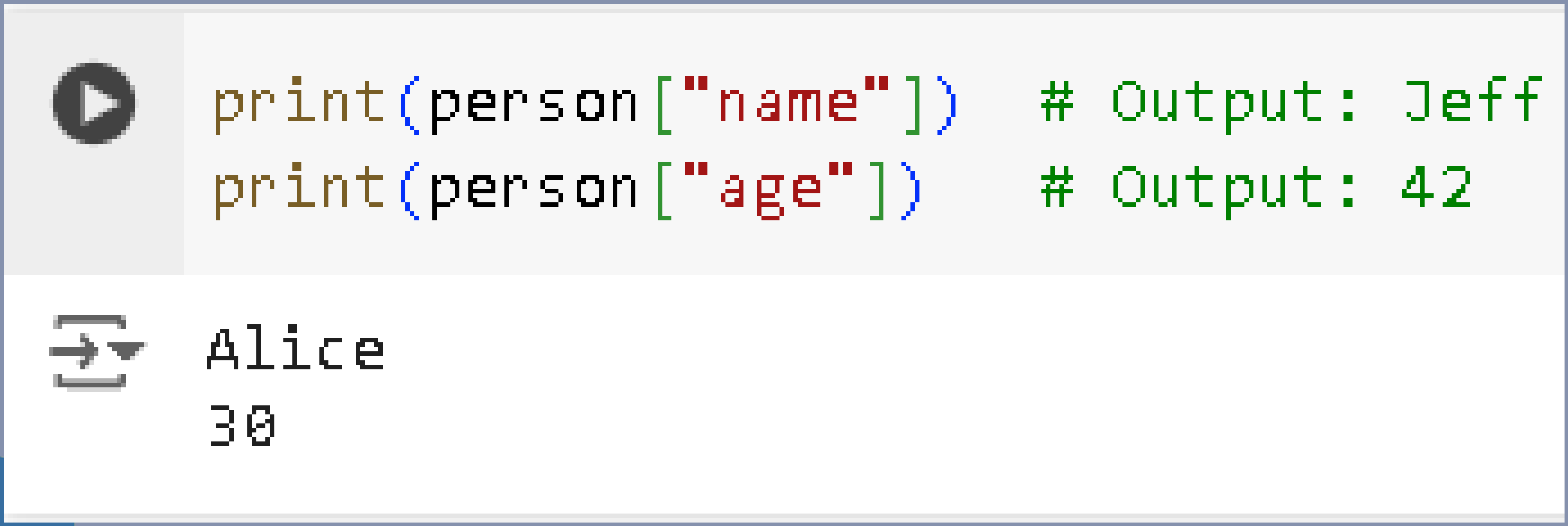
You can create a dictionary by placing key-value pairs inside curly braces “{}”, with a colon “:” separating each key from its value.



```
person = {
    "name": "Jeff",
    "age": 42,
    "city": "New York"
}
```

Accessing Dictionary Elements

You can access the value associated with a specific key using square brackets “[]”.

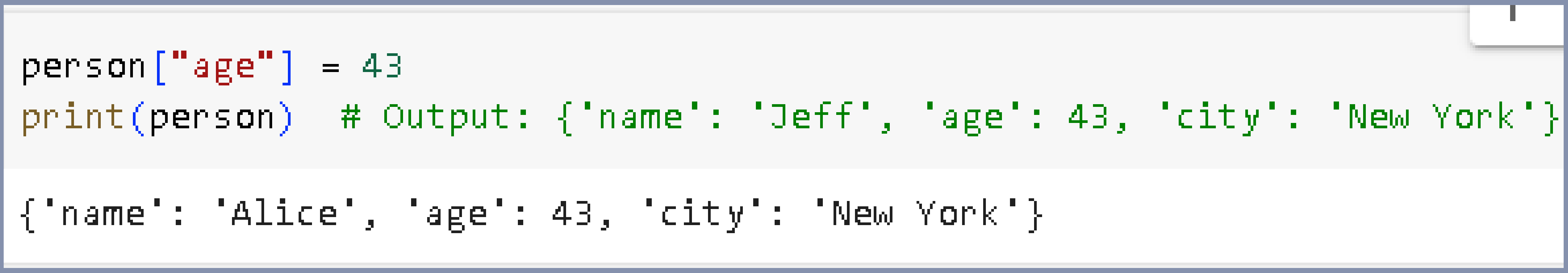


```
print(person["name"])  # Output: Jeff
print(person["age"])   # Output: 42
```

```
Alice
30
```

Modifying Dictionary Elements

You can change the value of a specific key by accessing it directly.

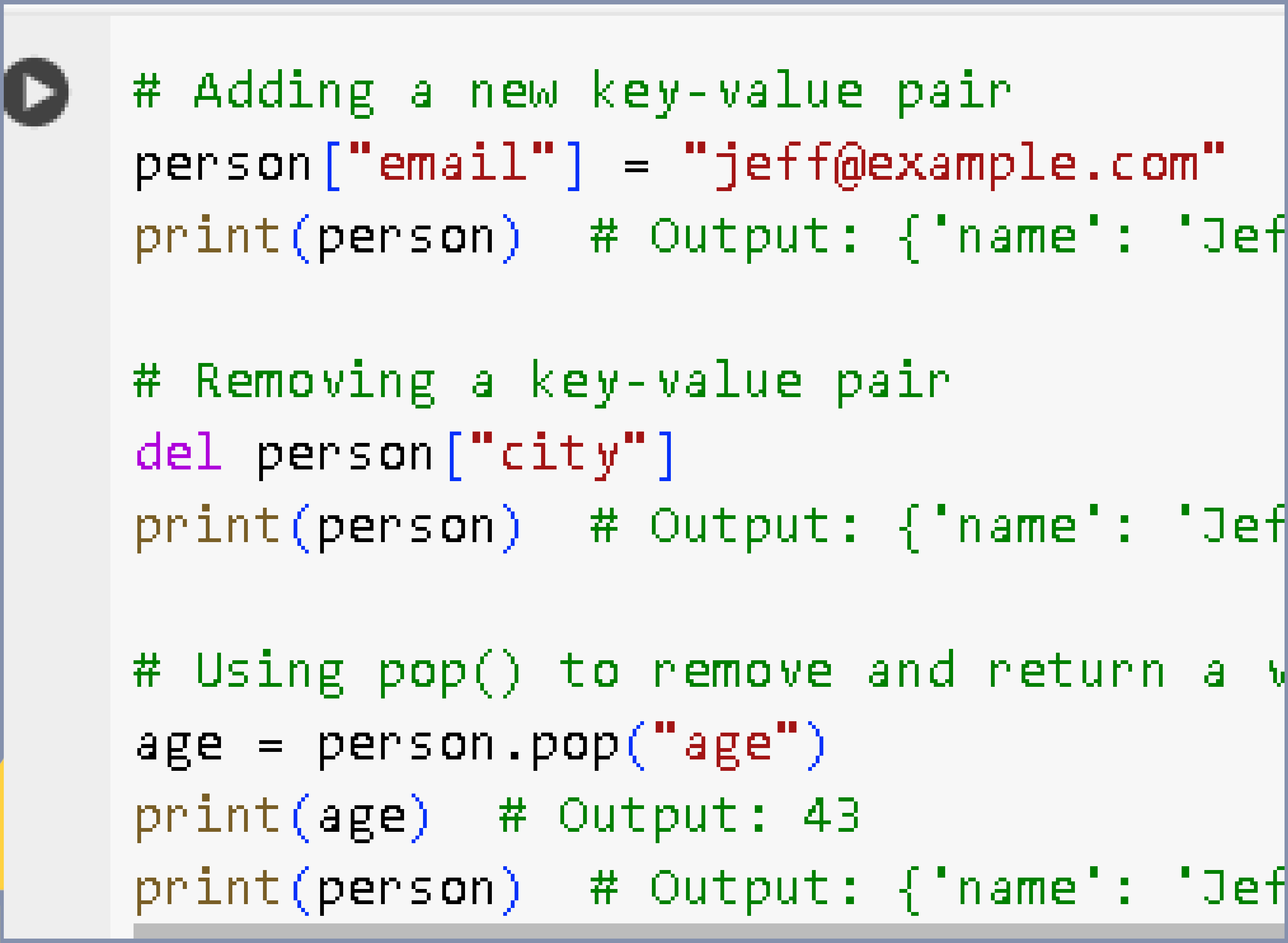


```
person["age"] = 43
print(person)  # Output: {'name': 'Jeff', 'age': 43, 'city': 'New York'}
```

```
{'name': 'Alice', 'age': 43, 'city': 'New York'}
```

Adding and Removing Elements

You can add new key-value pairs to a dictionary or remove existing ones using the "del" statement or the "pop()" method.

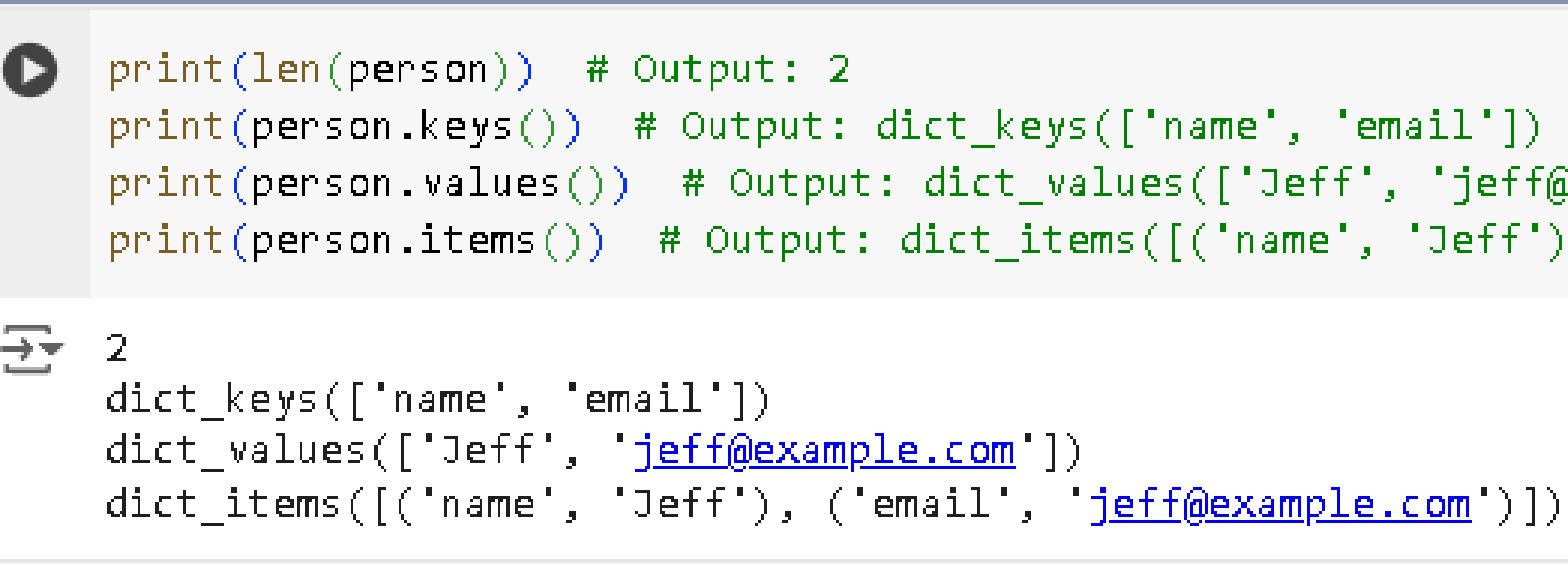


```
# Adding a new key-value pair
person["email"] = "jeff@example.com"
print(person)  # Output: {'name': 'Jef

# Removing a key-value pair
del person["city"]
print(person)  # Output: {'name': 'Jef

# Using pop() to remove and return a v
age = person.pop("age")
print(age)  # Output: 43
print(person)  # Output: {'name': 'Jef
```

Common Dictionary Methods

- len(dict): Returns the number of key-value pairs in the dictionary.
- dict.keys(): Returns a view object that displays a list of all the keys in the dictionary.
- dict.values(): Returns a view object that displays a list of all the values in the dictionary.
- dict.items(): Returns a view object that displays a list of the dictionary's key-value tuple pairs.

Common Dictionary Methods

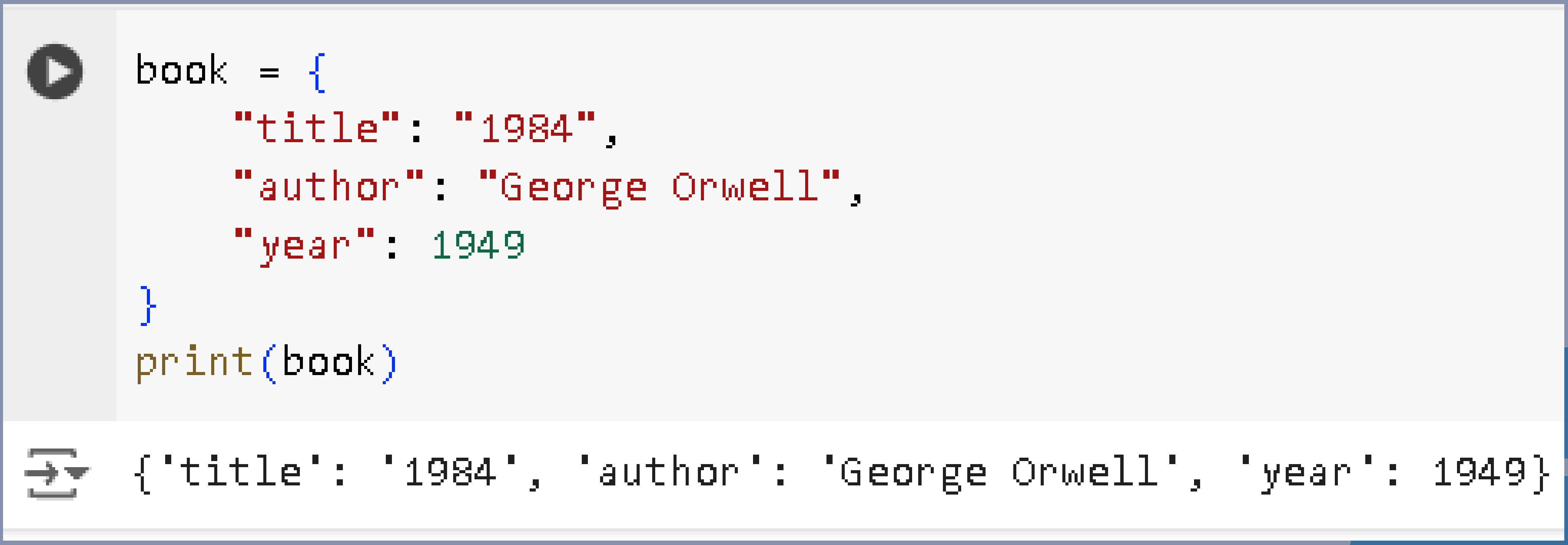


```
print(len(person))  # Output: 2
print(person.keys())  # Output: dict_keys(['name', 'email'])
print(person.values())  # Output: dict_values(['Jeff', 'jeff@
print(person.items())  # Output: dict_items([('name', 'Jeff')
```

```
2
dict_keys(['name', 'email'])
dict_values(['Jeff', 'jeff@example.com'])
dict_items([('name', 'Jeff'), ('email', 'jeff@example.com')])
```

Practice Exercises

Create a Dictionary: Create a dictionary to store information about a book (title, author, year published) and print it.

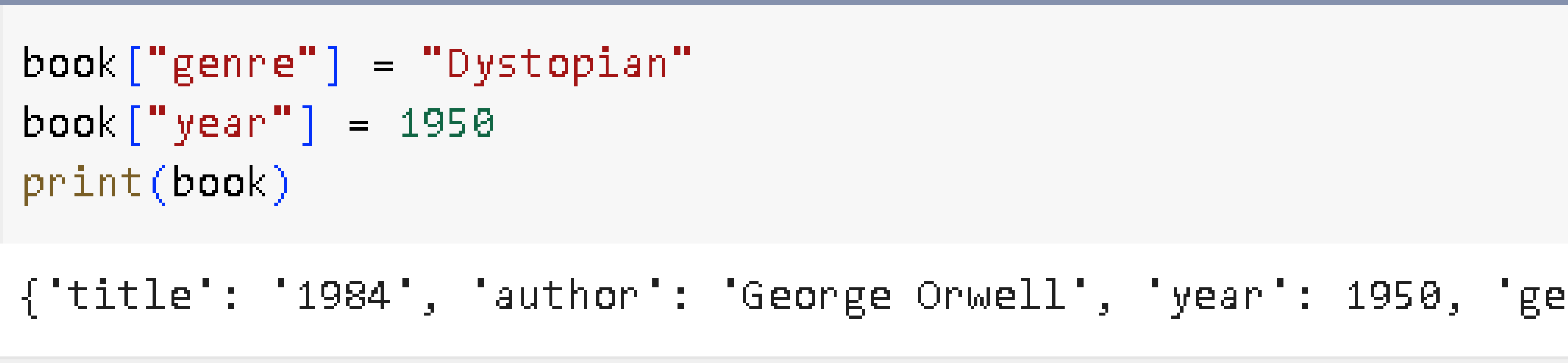


```python
book = {
    "title": "1984",
    "author": "George Orwell",
    "year": 1949
}
print(book)
```

```
{'title': '1984', 'author': 'George Orwell', 'year': 1949}
```

Practice Exercises

Modify the Dictionary: Add a new key-value pair for the genre of the book and then change the year published.

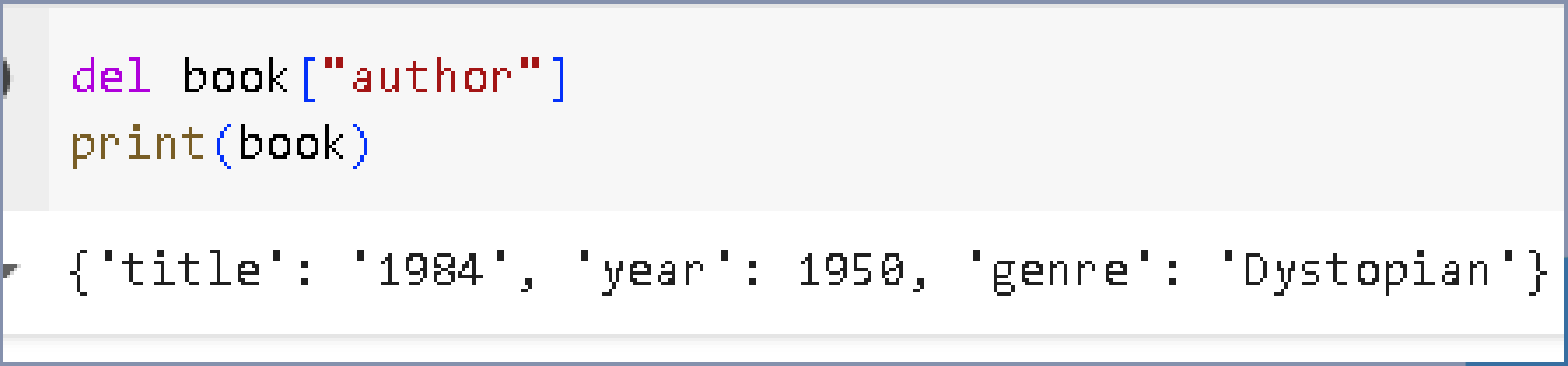


```python
book["genre"] = "Dystopian"
book["year"] = 1950
print(book)
```

```
{'title': '1984', 'author': 'George Orwell', 'year': 1950, 'ge
```

Practice Exercises

Remove a Key-Value Pair: Write a program that removes the author from the book dictionary and prints the updated dictionary.

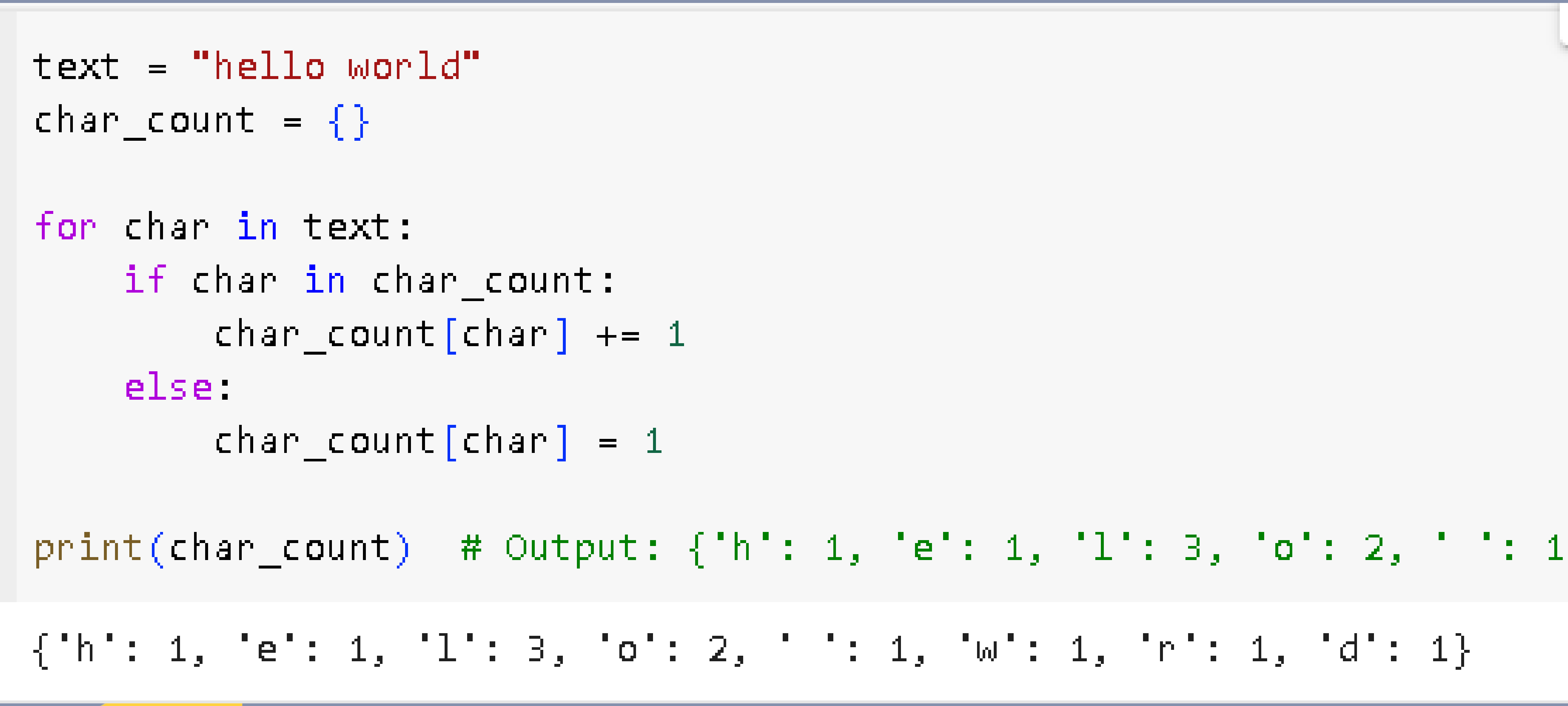


```python
del book["author"]
print(book)
```

```
{'title': '1984', 'year': 1950, 'genre': 'Dystopian'}
```

Practice Exercises

Count Occurrences: Write a program that counts the occurrences of each character in a string and stores the results in a dictionary.

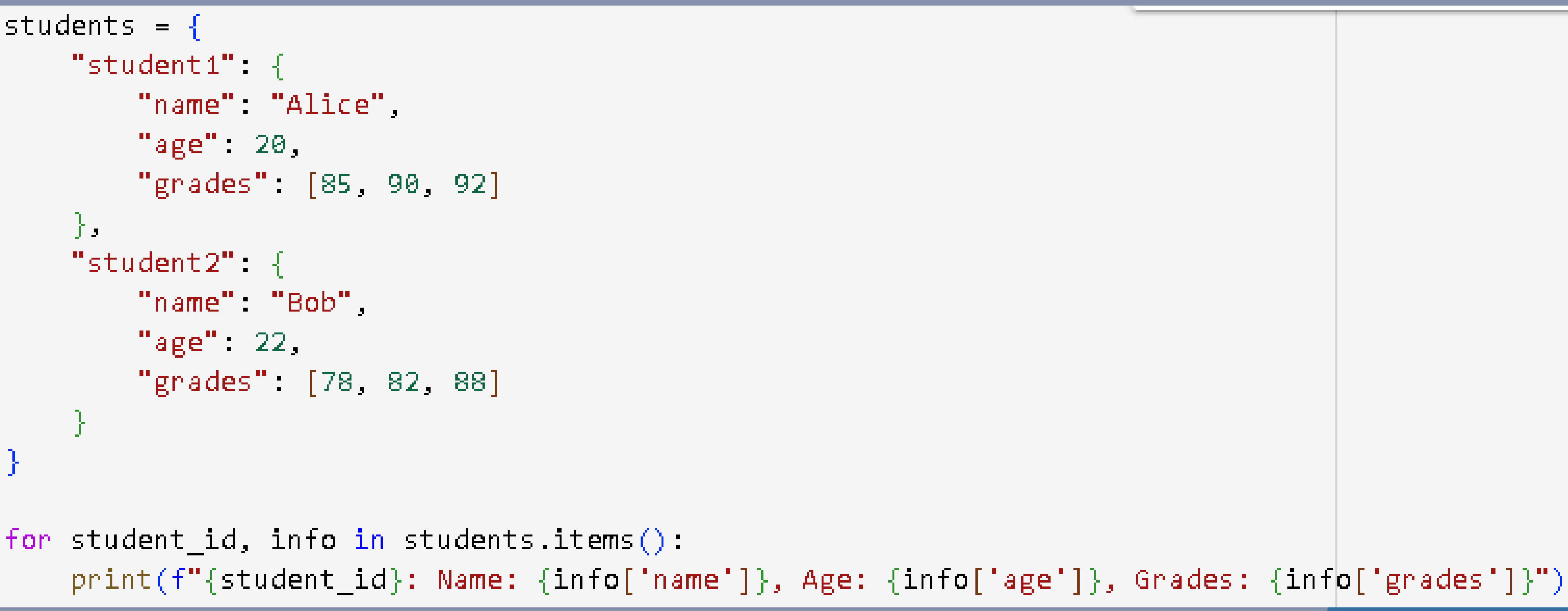


```python
text = "hello world"
char_count = {}

for char in text:
    if char in char_count:
        char_count[char] += 1
    else:
        char_count[char] = 1

print(char_count)  # Output: {'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1
```

```
{'h': 1, 'e': 1, 'l': 3, 'o': 2, ' ': 1, 'w': 1, 'r': 1, 'd': 1}
```

Practice Exercises

Nested Dictionaries: Create a nested dictionary to store information about multiple students (name, age, and grades) and print the information for each student.



```python
students = {
    "student1": {
        "name": "Alice",
        "age": 20,
        "grades": [85, 90, 92]
    },
    "student2": {
        "name": "Bob",
        "age": 22,
        "grades": [78, 82, 88]
    }
}

for student_id, info in students.items():
    print(f"{student_id}: Name: {info['name']}, Age: {info['age']}, Grades: {info['grades']}")
```

Conclusion

In this tutorial, you learned about dictionaries in Python, including how to create, access, modify, and manipulate them. You explored common dictionary methods and practiced working with dictionaries through various exercises. Dictionaries are a powerful data structure that allows you to store and retrieve data efficiently using key-value pairs.

Next Steps

In tutorial 9, we will look at how to use string manipulation and cover various string methods and formatting techniques in Python.

FOLLOW ME

for more tips you didn’t know you needed

Jeff Gentry

@www.linkedin.com/in/jefferycharlesgentry