Lists and Dictionaries - MAD 102 Week 8 Notes

Hia Al Saleh

October 23rd, 2024

Contents

1 Nested Lists

Lists can contain other lists, known as nested lists. For example:

```
crews = [['Mal', 'Washburne', 'Zoe'], ['Han', 'Chewie'], [
    'Kirk', 'Spock', 'McCoy']]
print(crews[1][1])  # Outputs 'Chewie'
```

Can use a for loop, with nested for loop, to iterate over the entire contents

```
crews = [['Mal', 'Washburne', 'Zoe'], ['Han', 'Chewie'], [
    'Kirk', 'Spock', 'McCoy']]
for position, crew in enumerate(crews):
        print('=' * 20)
        print(f'Crew #{position + 1}')
        print('=' * 20)
        for member in crew:
            print(member)
```

2 List Slicing

Slicing can be used to extract parts of a list. Syntax:

$$\text{list}[\text{startIndex} : \text{endIndex}]$$

Example:

```
numbers = [1, 2, 3, 4, 5]
sliced = numbers[:3]  # [1, 2, 3]
```

3 List Comprehensions

List comprehensions allow for modifying elements in a list:

```
numbers = [1, 2, 3, 4, 5]
new_list = [x * 2 for x in numbers]  # [2, 4, 6, 8, 10]
```

4 Dictionaries

Dictionaries store key-value pairs:

```
fruit_prices = {'apple': 2, 'banana': 1, 'orange': 3}
print(fruit_prices['apple'])  # Outputs 2
```

5 Procedural Programming

Up to this point, we've focused on procedural programming, executing steps in sequence with conditional statements and loops.

6 Object Oriented Programming

Object-Oriented Programming (OOP) is a paradigm that focuses on **objects**, containing both attributes and behaviors. Examples of objects include a *person*, a *car*, and a *bank account*.

6.1 Attributes and Behaviors

Attributes represent the data stored in an object, while behaviors are actions that an object can perform. For example, a dog can *bark*, *run*, and *eat*.

6.2 Classes and Objects

A **class** is a blueprint for creating objects, containing attributes and methods. Here's an example:

```
class Dog:
    def __init__(self, name):
        self.name = name
    def bark(self):
        print(f"{self.name} says Woof!")
```

7 Class Exercises

7.1 Favorite Animals

Task:

- Ask the user for a name and their favorite animal.
- Keep asking until they quit.
- Display the list of names entered.
- Display the list of animals entered.
- Display names and their favorite animal.

7.1.1 Python Code

The following Python code implements the *Favorite Animals* task.

```
def favorite_animals():
    names = []
    animals = []

    while True:
        name = input("Enter your name (or 'quit' to stop):
            ")
        if name.lower() == 'quit':
            break
        animal = input(f"What's {name}'s favorite animal?:
            ")
        names.append(name)
        animals.append(animal)

    print("\nList of Names:", names)
    print("List of Favorite Animals:", animals)
    print("\nNames and their favorite animals:")
    for i in range(len(names)):
        print(f"{names[i]}'s favorite animal is {animals[i
            ]}")
favorite_animals()
```

7.2 Mean - Median - Mode

Task:

- Ask the user to enter a series of numbers.
- Calculate the mean, median, and mode of the numbers.
- Use separate functions for each calculation.

7.2.1 Python Code

The following Python code implements the *Mean, Median, and Mode* task.

```
import statistics

def calculate_mean_median_mode():
    numbers = list(map(float, input("Enter numbers
        separated by space: ").split()))

    mean = statistics.mean(numbers)
    median = statistics.median(numbers)
    try:
        mode = statistics.mode(numbers)
    except statistics.StatisticsError:
        mode = "No mode"

    print(f"Mean: {mean}")
    print(f"Median: {median}")
    print(f"Mode: {mode}")
calculate_mean_median_mode()
```