Python

Class 5

Introduction to Python

Azmain Adel

August 25, 2024

01.

Review of Previous Class

Review Topics

- String
- Set
- Tuple

Solution to Problem 1

Python program to find number of vowels in a given string.

```
def count_vowels(string):
    vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
    vowel_count = 0

    for char in string:
        if char in vowels:
            vowel_count += 1

    return vowel_count

input_string = input("Enter a string: ")
result = count_vowels(input_string)
print(f"The number of vowels in the string is: {result}")
```

Solution to Problem 2

Python program to list unique characters with their count in a string

Input: “Hello” Output: h=1, e=1, l=2, o=1

```
def count_unique_chars(string):
    char_count = {}

    for char in string.lower():
        if char.isalnum():
            char_count[char] = char_count.get(char, 0) + 1

    result = ", ".join([f"{char}={count}" for char, count in char_count.items()])
    return result

input_string = input("Enter a string: ")
output = count_unique_chars(input_string)
print(output)
```

Solution to Problem 3

Program to find common elements in two lists with the help of set operations

```python
def find_common_elements(list1, list2):
    set1 = set(list1)
    set2 = set(list2)
    common_elements = list(set1.intersection(set2))
    return common_elements

# Example usage
l1 = [1, 2, 3]
l2 = [3, 4]

result = find_common_elements(l1, l2)
print(result)
```

02.

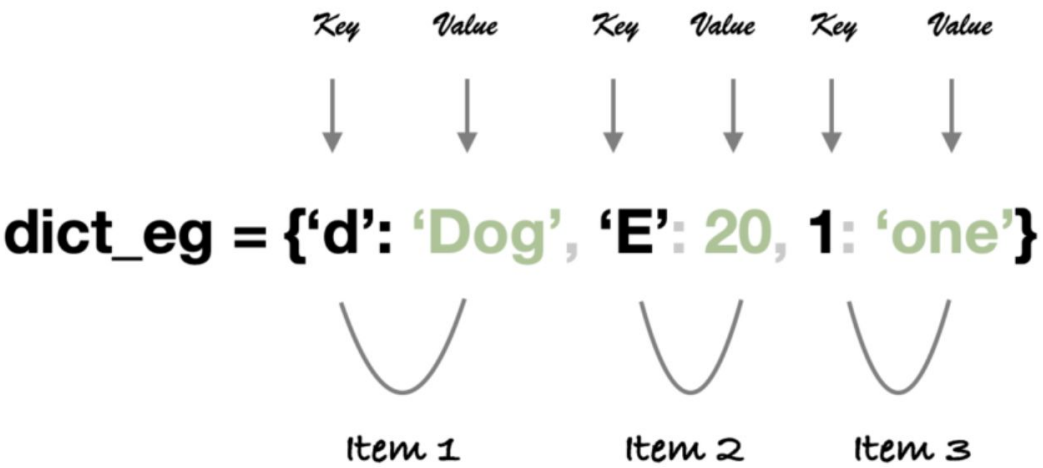
Dictionary

Dictionaries in Python

- Data type that stores data in key-value pairs.
- Each key in a dictionary is **unique** and maps to a value.

```
d1 = {
 "Fruit":["Mango","Banana"],
 "Flower":["Rose", "Lotus"]
}
```

Dictionary in Python



Dictionary Operations

- Access:
  - my_dict[“key_name”]
  - my_dict.get(“key_name”, default=None)
- Add/Modify:
  - my_dict[“key_name”] = 45
  - my_dict.update({“key_name”: 45})
- Remove:
  - del my_dict[“key_name”]
  - my_dict.pop(“key_name”)

Dictionary Operations

- View Items:
  - my_dict.keys()
  - my_dict.values()
- Iterate:
  - for key, value in my_dict.items():

More Dictionary Methods

- dict.clear()
- dict.copy()
- dict.has_key(key)
- dict.setdefault(key, default=None)

03.

File Handling

Working with Files

1. Open a file
2. Read/Write on the file
3. Close

Opening Files

file = open("filename", "mode")

File Opening Modes

- **r** for reading
- **r+** opens for reading and writing (cannot truncate a file)
- **w** for writing
- **w+** for writing and reading (can truncate a file)
- **rb** for reading a binary file. The file pointer is placed at the beginning of the file.
- **rb+** reading or writing a binary file
- **wb+** writing a binary file
- **a+** opens for appending
- **ab+** Opens a file for both appending and reading in binary.

Reading Files

- **read()** – Reads the entire file
- **readline()** – Reads one line at a time
- **readlines()** – Reads all lines into a list

Reading Files

```
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

```
with open("example.txt", "r") as file:
    line = file.readline()
    while line:
        print(line, end='')
        line = file.readline()
```

```
with open("example.txt", "r") as file:
    lines = file.readlines()
    for line in lines:
        print(line, end='')
```

Writing Files

- **write()** – Writes string to file
- **writelines()** – Writes all strings as lines to file

Writing Files

```
with open("foo.txt", "w") as file:
   file.write("Hello, World!")
   print ("Content added Successfully!!")
```

```
lines = ["First line\n", "Second line\n", "Third line\n"]
with open("example.txt", "w") as file:
   file.writelines(lines)
   print ("Content added Successfully!!")
```

Closing Files

- **file.close()**
- Opening with **with()** automatically closes the file

04.

OOP in Python

OOP Recap

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

OOP Recap

Encapsulation	Abstraction	Inheritance	Polymorphism
When an object only exposes the selected information.	Hides complex details to reduce complexity.	Entities can inherit attributes from other entities.	Entities can have more than one form.

Class and Objects

```
# defining class
class Smartphone:
    # constructor
    def __init__(self, device, brand):
        self.device = device
        self.brand = brand

    # method of the class
    def description(self):
        return f"{self.device} of {self.brand} supports Android 14"

# creating object of the class
phoneObj = Smartphone("Smartphone", "Samsung")
print(phoneObj.description())
```

05.

Recap and Q&A

Open floor for questions and clarifications

06.

To-do at Home

Self-study

Recap your OOP knowledge

Thank you.

azmainadel47@gmail.com
linkedin.com/in/azmainadel
azmainadel.com