CODE AND SUMMARY OF CLASS 2

Sahir Ahmed Sheikh
Saturday (2 – 5)

Teachers:
Muhammad Bilal And Ali Aftab Sheikh

Code And Summary Of Class 2 – Saturday (2 – 5) | Quarter 3

In today's session, we delved into Python String Methods, Data Structures in Python, Python Operators, Python Memory Management, the Random Module, and Conditional Statements. Additionally, our instructor informed us that we would be coding in **Google Colab** during this class and would transition to a code editor in the next session.

We began by searching for **Google Colab** in the browser and clicking on the second link to access the platform. After opening **Google Colab**, we created a new notebook by selecting **"Create a new notebook."** The first step after creating the notebook was naming the file, so I named it **"Class 2."**

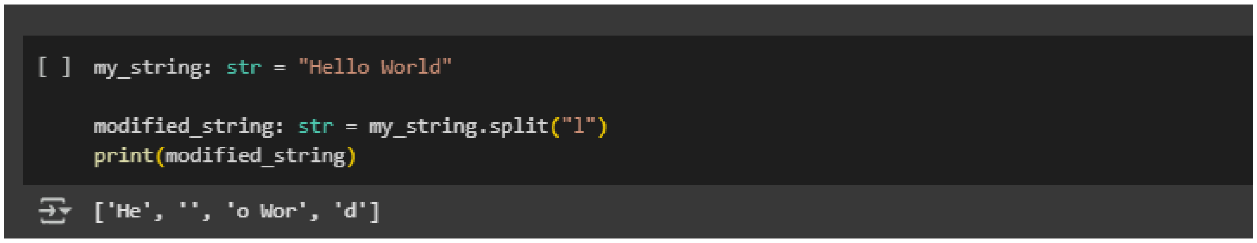
Our teacher explained that the **.ipynb** extension, visible in the filename, is specific to **Jupyter Notebook**. If we want the same interface in our code editor, we simply need to install the **Jupyter Notebook extension**. Lastly, the instructor advised us to **install Python on our laptops before the next class** to ensure a smooth learning experience without any technical difficulties.
Class Code: Class Notebook

Introduction to String Methods

1. Split() Method

- The split() method is used to break a string into multiple parts based on a specified delimiter.
- By default, it splits at spaces, but any character or substring can be used as a delimiter.
- The method returns a list containing the separated elements.

Example:



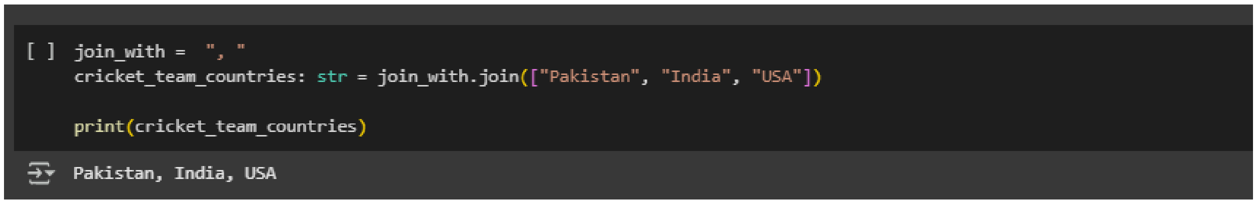
```
my_string: str = "Hello World"

modified_string: str = my_string.split("l")
print(modified_string)
```

```
['He', '', 'o Wor', 'd']
```

2. Join() Method

- The join() method combines the elements of a list into a single string, using a specified separator.
- This is useful when converting lists back into readable text formats.

Example:



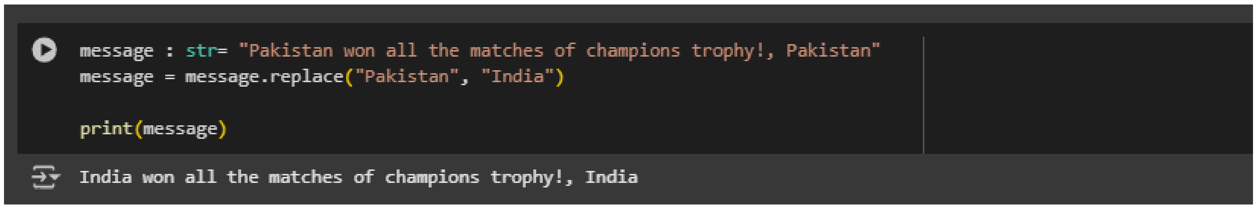
```
[ ] join_with = ", "
    cricket_team_countries: str = join_with.join(["Pakistan", "India", "USA"])

    print(cricket_team_countries)
```

```
Pakistan, India, USA
```

3. Replace() Method

- The replace() method replaces a specified substring with another substring in a given string.
- It does not modify the original string but returns a new string with the replacements.

Example:



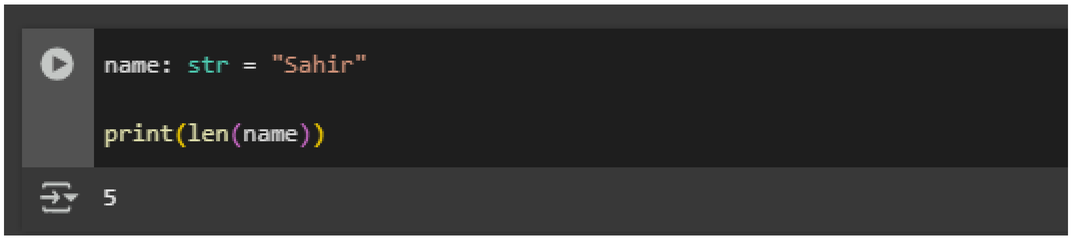
```
message : str= "Pakistan won all the matches of champions trophy!, Pakistan"
message = message.replace("Pakistan", "India")

print(message)
```

```
India won all the matches of champions trophy!, India
```

4. Len() Method

- The len() function is used to determine the number of characters in a string or elements in an iterable like lists or dictionaries.
- It counts all characters, including spaces.

Example:



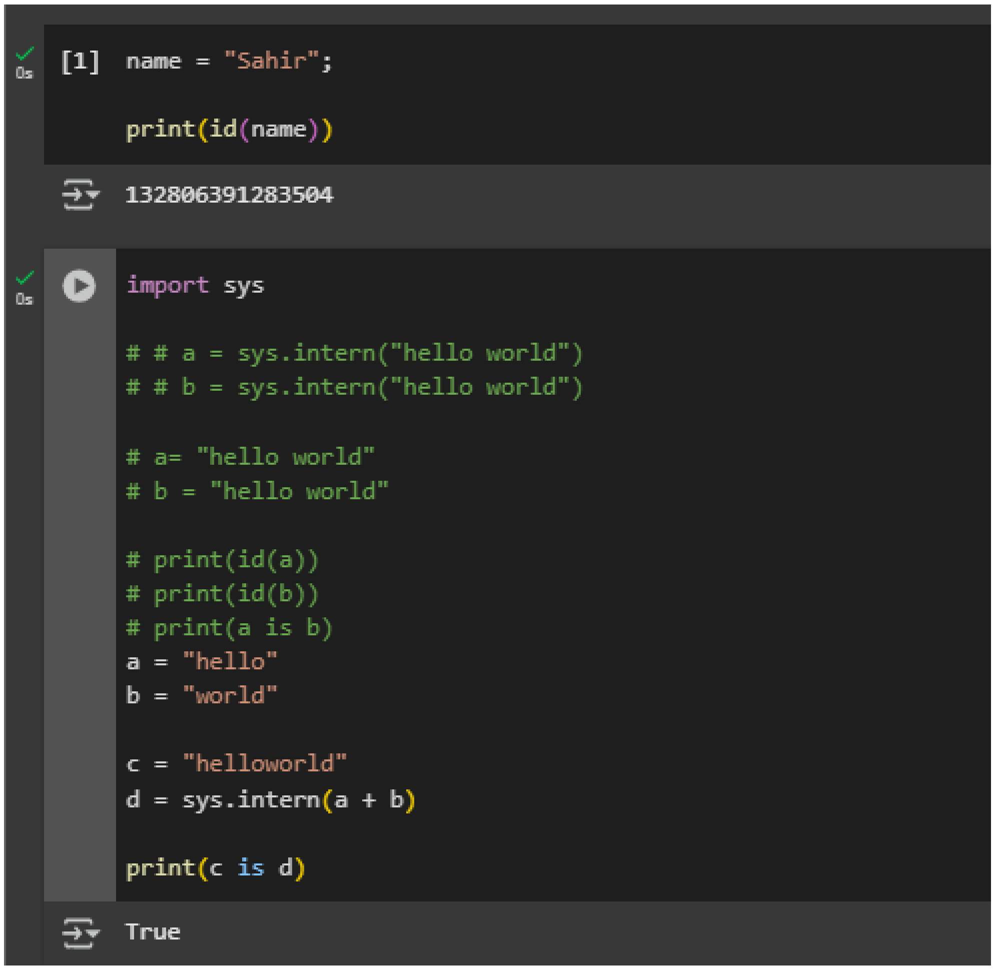
```
name: str = "Sahir"

print(len(name))
```

```
5
```

5. String Interning & ID() Function

- Python optimizes memory usage by storing identical strings in a single memory location, a process known as **string interning**.
- The id() function returns the memory address of a variable, allowing us to check if two variables refer to the same object.

Example:



```
[1] name = "Sahir";

    print(id(name))

132806391283504
```

```
import sys

# # a = sys.intern("hello world")
# # b = sys.intern("hello world")

# a= "hello world"
# b = "hello world"

# print(id(a))
# print(id(b))
# print(a is b)
a = "hello"
b = "world"

c = "helloworld"
d = sys.intern(a + b)

print(c is d)

True
```

Data Structures in Python

1. Lists

- Lists are ordered collections that can store multiple data types.
- They are **mutable**, meaning we can modify their elements.
- Lists allow **indexing and slicing** for easy element access.

- Common operations include adding, removing, and modifying elements.

2. Slice

- Slicing allows us to extract a portion of a list or string by specifying a start and end index.
- Negative indices can be used to access elements from the end of a list.

3. Append() Method

- The append() method adds an element to the **end** of a list.
- It modifies the original list.

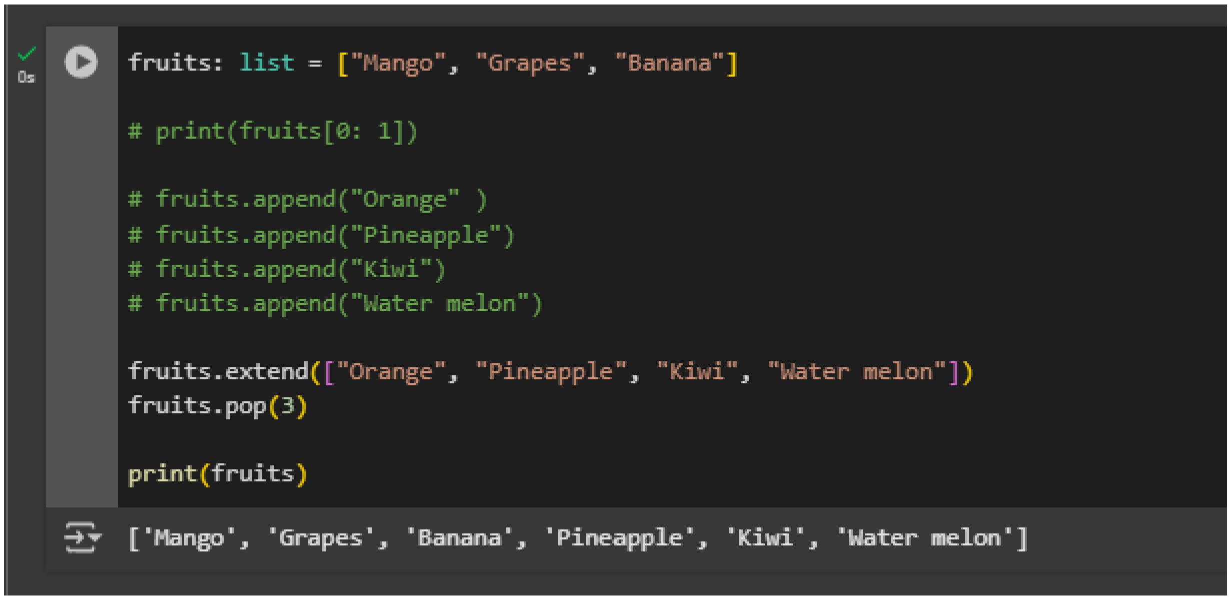
4. Extend() Method

- The extend() method adds multiple elements to a list at once.
- Unlike append(), it takes an iterable (e.g., another list) and adds each element separately.

5. Pop() Method

- The pop() method removes an element from the list.
- If an index is provided, it removes the element at that index; otherwise, it removes the last element.

Example:



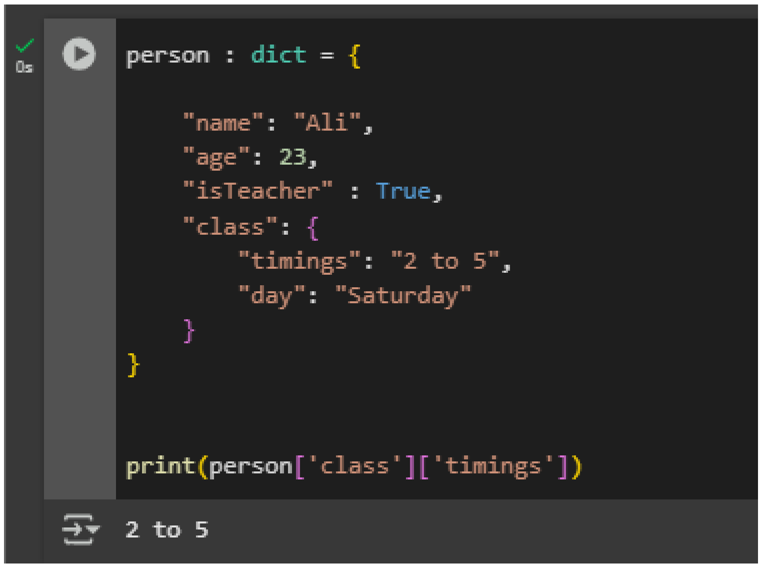
```
fruits: list = ["Mango", "Grapes", "Banana"]

# print(fruits[0: 1])

# fruits.append("Orange" )
# fruits.append("Pineapple")
# fruits.append("Kiwi")
# fruits.append("Water melon")

fruits.extend(["Orange", "Pineapple", "Kiwi", "Water melon"])
fruits.pop(3)

print(fruits)
```

```
['Mango', 'Grapes', 'Banana', 'Pineapple', 'Kiwi', 'Water melon']
```

6. Dictionaries

- A dictionary stores data in **key-value** pairs.
- Each key must be unique, and values can be of any data type.
- Dictionaries allow fast lookups and modifications.

Example:



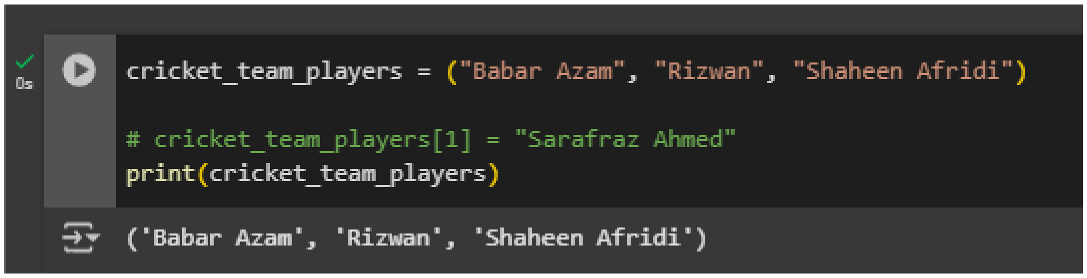
```
person : dict = {

    "name": "Ali",
    "age": 23,
    "isTeacher" : True,
    "class": {
        "timings": "2 to 5",
        "day": "Saturday"
    }
}


print(person['class']['timings'])
```

```
2 to 5
```

7. Tuples

- Tuples are similar to lists but **immutable**, meaning their elements cannot be changed after creation.
- They use parentheses `()` instead of square brackets `[]`.
- Tuples are useful for storing fixed collections of data.

Example:



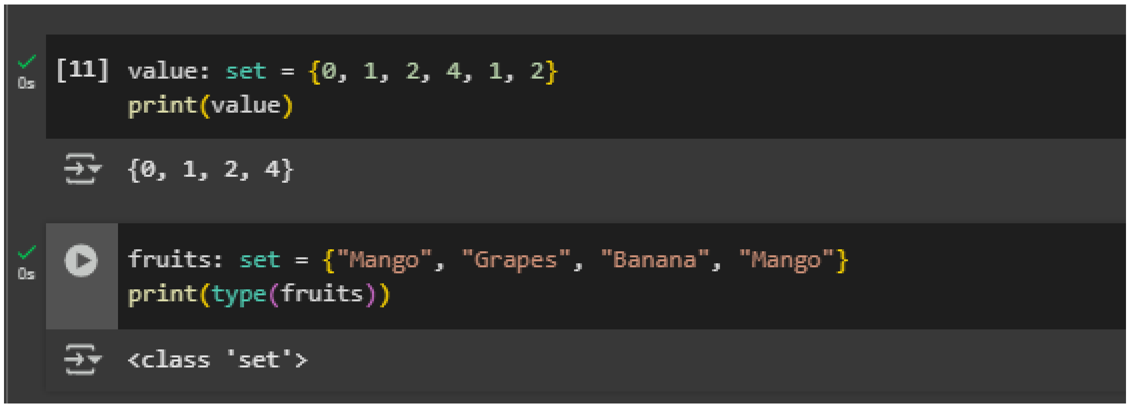
```
cricket_team_players = ("Babar Azam", "Rizwan", "Shaheen Afridi")

# cricket_team_players[1] = "Sarafraz Ahmed"
print(cricket_team_players)
```

```
('Babar Azam', 'Rizwan', 'Shaheen Afridi')
```

8. Sets

- A set is an **unordered** collection that **only stores unique values**.
- Duplicate values are automatically removed.
- Sets are useful for eliminating duplicate entries and performing mathematical operations like unions and intersections.

Example:



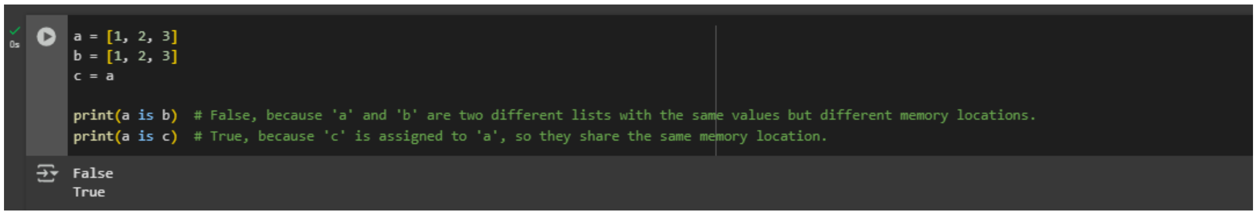
```
[11] value: set = {0, 1, 2, 4, 1, 2}
     print(value)
```
```
{0, 1, 2, 4}
```
```
fruits: set = {"Mango", "Grapes", "Banana", "Mango"}
print(type(fruits))
```
```
<class 'set'>
```

Python Operators

1. IS Operator

- The is operator checks whether two variables **point to the same memory location**.
- It does not compare values but references.

Example:

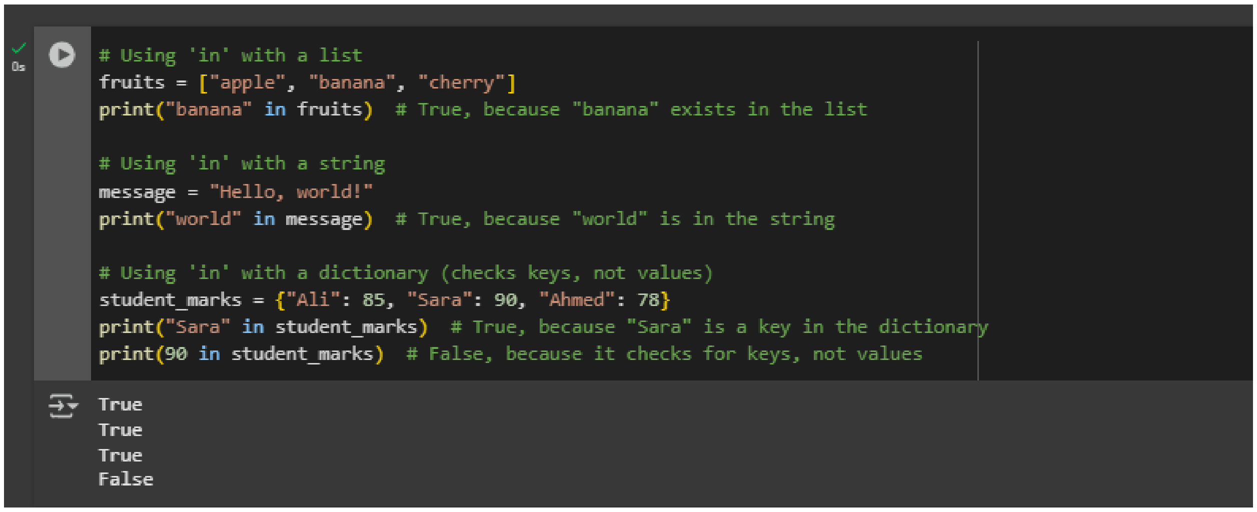


```
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a is b)  # False, because 'a' and 'b' are two different lists with the same values but different memory locations.
print(a is c)  # True, because 'c' is assigned to 'a', so they share the same memory location.
```
```
False
True
```

2. IN Operator

- The in operator checks if a value exists within a sequence (e.g., strings, lists, tuples, dictionaries).

Example:

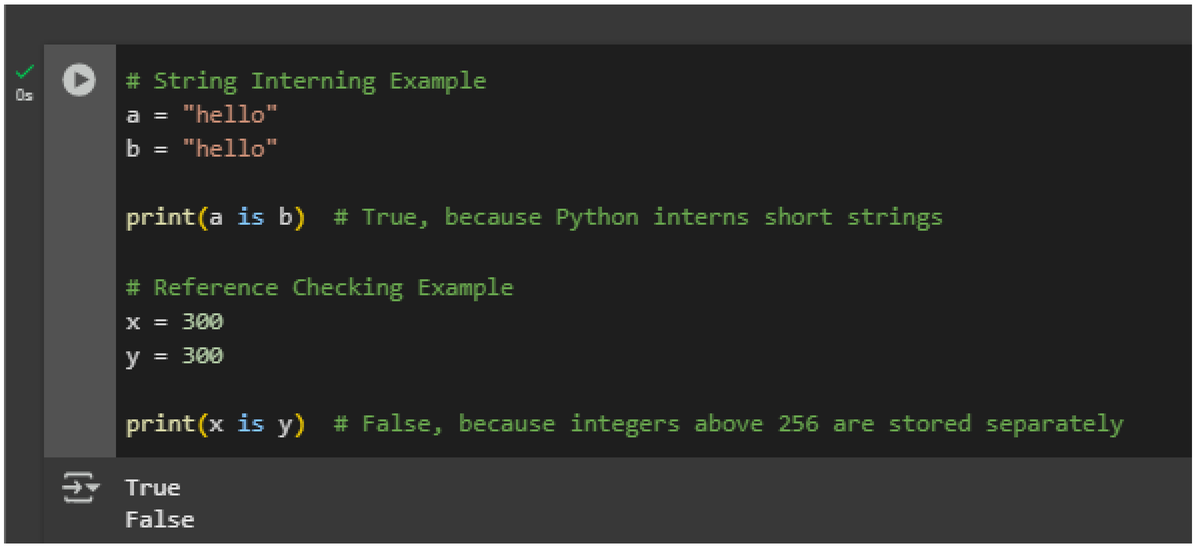


```
# Using 'in' with a list
fruits = ["apple", "banana", "cherry"]
print("banana" in fruits)  # True, because "banana" exists in the list

# Using 'in' with a string
message = "Hello, world!"
print("world" in message)  # True, because "world" is in the string

# Using 'in' with a dictionary (checks keys, not values)
student_marks = {"Ali": 85, "Sara": 90, "Ahmed": 78}
print("Sara" in student_marks)  # True, because "Sara" is a key in the dictionary
print(90 in student_marks)  # False, because it checks for keys, not values
```

```
True
True
True
False
```

Python Memory Management

- **String Interning:** Python stores identical strings in a shared memory location to save memory.
- **Reference Checking:** Using id(var1) == id(var2), we can verify if two variables share the same memory address.



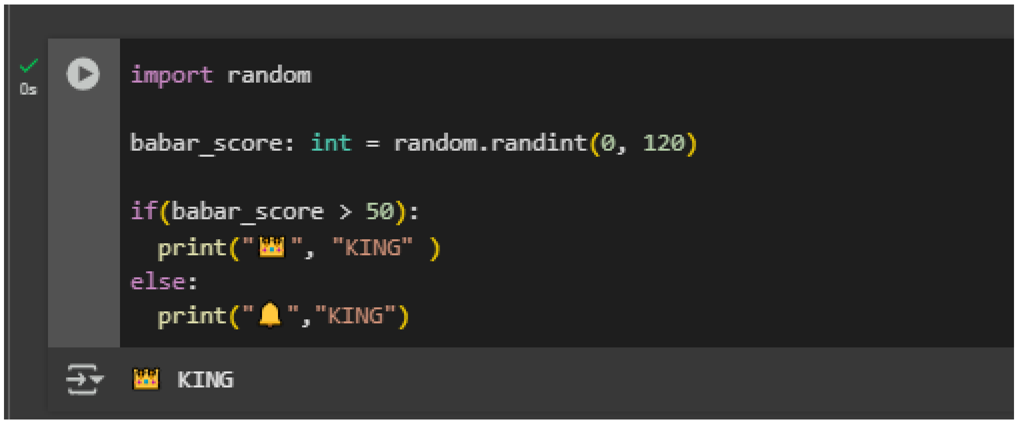
```
# String Interning Example
a = "hello"
b = "hello"

print(a is b)  # True, because Python interns short strings

# Reference Checking Example
x = 300
y = 300

print(x is y)  # False, because integers above 256 are stored separately
```

```
True
False
```

Random Module & Conditional Statements

- **Generating random numbers:** The random.randint() function generates a random integer within a given range.
- **If-Else Conditions:** Used to perform actions based on conditions in the program.

Example:



```
import random

babar_score: int = random.randint(0, 120)

if(babar_score > 50):
  print("👑", "KING" )
else:
  print("🔔","KING")
```

```
👑 KING
```

Assignments

✓ **Assignment 1:** Modern AI Python Steps Revise the first seven steps and learn step eight from Sir Zia's repository: Learn Modern AI with Python - GitHub Repo

✓ **Assignment 2:** Unit Converter Project Work on the Unit Converter project from the repository: Unit Converter - GitHub Repo

✓ **Assignment 3:** Machine Learning vs. Deep Learning vs. Generative AI Research and prepare a PowerPoint presentation on the differences between Machine Learning, Deep Learning, and Generative AI. Upload the presentation on LinkedIn.

📅 **Deadline:** Before the next class

📌 **Submission:** Upload all assignments on LinkedIn Submit all class assignments using this form: Assignment Submission Form

Thank You for Reading!

Hope you understood Class 2 well.

"Education is the most powerful weapon which you can use to change the world." *– Nelson Mandela*