2.3-Strings النصوص

In Python, strings are sequences of characters enclosed within single quotes (' '), double quotes (" "), or triple quotes (''' ''' or """ """). Strings are one of the most commonly used data types in Python for representing text.

Let's go through various operations and techniques for working with Python strings, along with explanations and examples.

1. Creating Strings انشاء النصوص

Strings can be created by enclosing characters in quotes.

Example:

Python Code

```
# Creating strings
string1 = "Hello, World!"  # Double quotes
string2 = 'Python is fun!'  # Single quotes
string3 = '''This is a multi-line string using triple quotes.'''  # Triple quotes

print(string1)
print(string2)
print(string3)
```

Output:

```
Hello, World!
Python is fun!
This is a multi-line string using triple quotes.
```

2. String Immutability ثبات النص

Strings in Python are immutable, meaning that once a string is created, it cannot be changed. However, you can create a new string based on modifications of the original string.

Example:

Python Code

```
my_string = "Hello"
# Trying to change the first character (will raise an error)
# my_string[0] = 'h'  # Uncommenting this line will raise a TypeError

# Reassigning a new value
my_string = "hello"
print(my_string)
```

Output:

```
hello
```

3. Accessing Characters in a String الوصول الى احرف النص

Strings are indexed, meaning you can access specific characters by their index. Python uses **zero-based indexing**, so the first character is at index 0.

Example:

```
Python Code
my_string = "Python"
print(my_string[0])  # First character
print(my_string[1])  # Second character
print(my_string[-1]) # Last character
```

Output:

```
P
y
n
```

Strings in Python are **immutable**, meaning you cannot modify them directly. However, you can create a new string by performing operations on the original string.

Example:

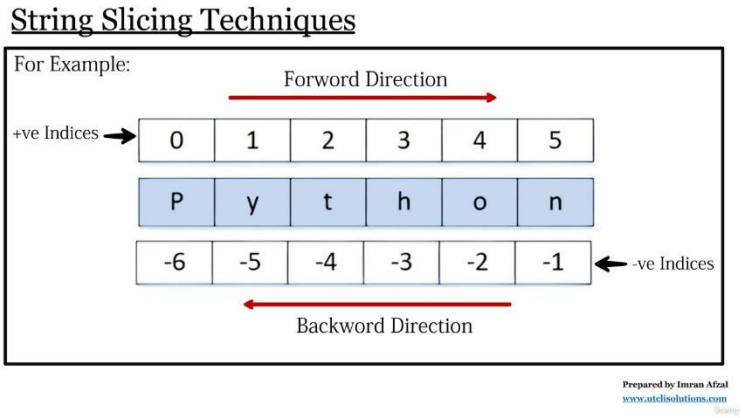
```
Python Code
my_string = "Hello, World!"
# Attempting to modify a string (this will raise an error)
# my_string[0] = 'h'  # Uncommenting this will raise a TypeError

# Reassign a modified string
my_string = "hello" + my_string[5:]
print(my_string)
```

Output:

```
hello, World!
```

4. String Slicing تقطيع النصوص

You can slice strings to extract a portion of the string using the syntax string[start:end]. The end index is excluded.



Example:

Python Code

```
my_string = "Hello, Python!"
print(my_string[0:5])   # Characters from index 0 to 4
print(my_string[7:])    # Characters from index 7 to the end
print(my_string[:5])    # Characters from the start to index 4
print(my_string[::2])   # Characters with a step of 2 (every second character)
```

Output:

```
Hello
Python!
Hello
Hlo yhn
```

5. String Concatenation **دمج النصوص**

You can concatenate strings using the + operator.

Example:

Python Code

```
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name
print(full_name)
```

Output:

```
John Doe
```

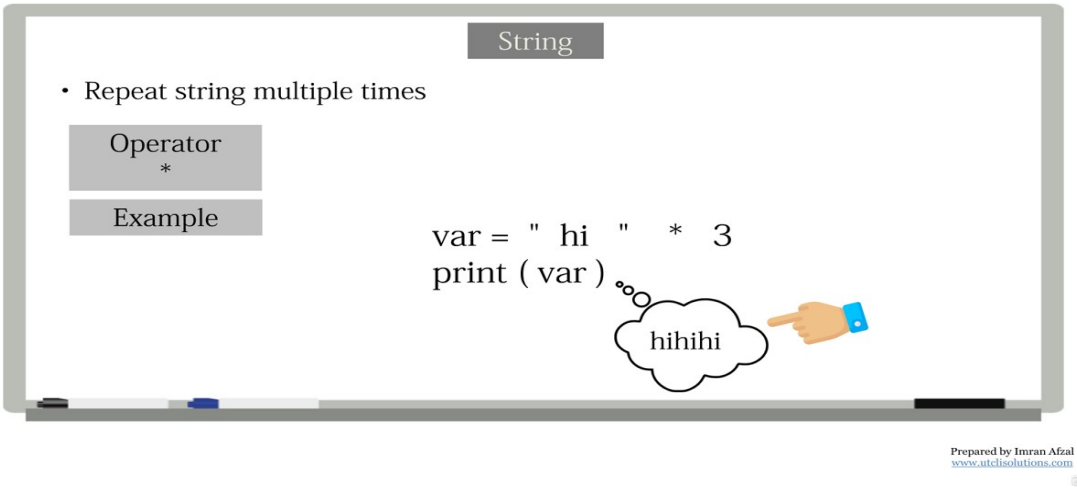
Explanation:

- The + operator joins two strings together.
- A space (" ") is added between first_name and last_name.

6. String Repetition **تكرار النص**

You can repeat a string multiple times using the * operator.

Introduction to Strings



Example:

Python Code
```
word = "Hello"
repeated_word = word * 3
print(repeated_word)
```
Output:

HelloHelloHello

Explanation:

- The * operator repeats the string word three times.

7. String Formatting (f-strings) تنسيق النص باستخدامf

Python allows you to embed variables or expressions inside strings using **f-strings** (formatted string literals).

Example:

Python Code
```
name = "Alice"
age = 25
greeting = f"Hello, my name is {name} and I am {age} years old."
print(greeting)
```
Output:

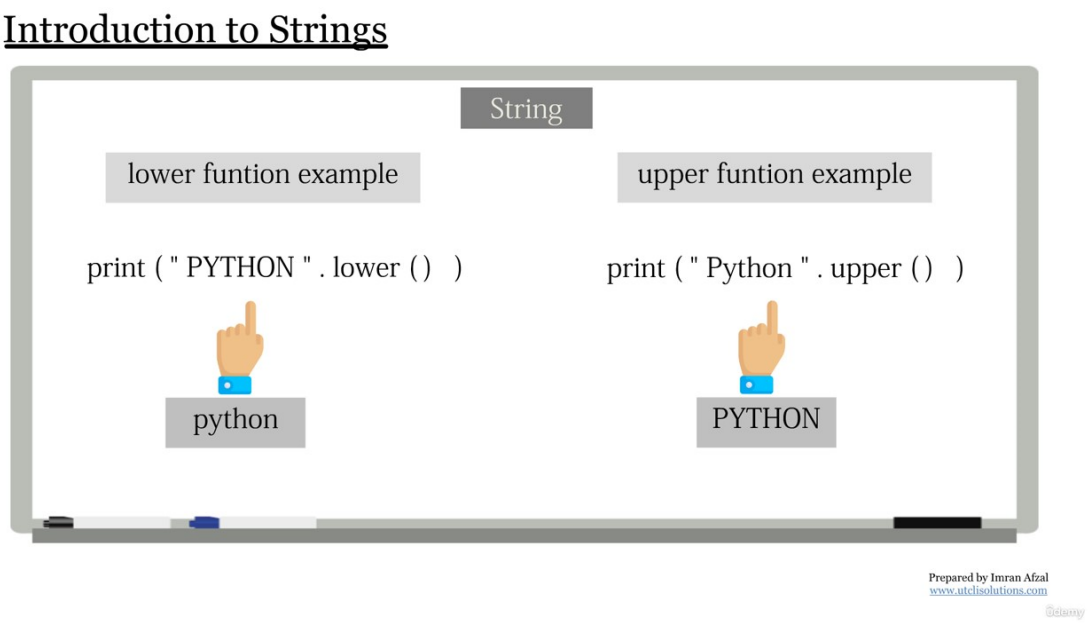
Hello, my name is Alice and I am 25 years old.

8. String Methods دوال النصوص

Python provides several built-in methods to manipulate and process strings.

8.1. lower() and upper() دالة الاحرف الصغيرة و الكبيرة

- **lower()**: Converts all characters to lowercase.
- **upper()**: Converts all characters to uppercase.



-

Example:

```
Python Code
my_string = "Hello, Python!"
print(my_string.lower())  # Converts to lowercase
print(my_string.upper())  # Converts to uppercase
```

Output:

```
hello, python!
HELLO, PYTHON!
```

8.2. strip() دالة إزالة المسافات بين الطرفين

- **strip()**: Removes leading and trailing whitespace from a string.
-

Example:

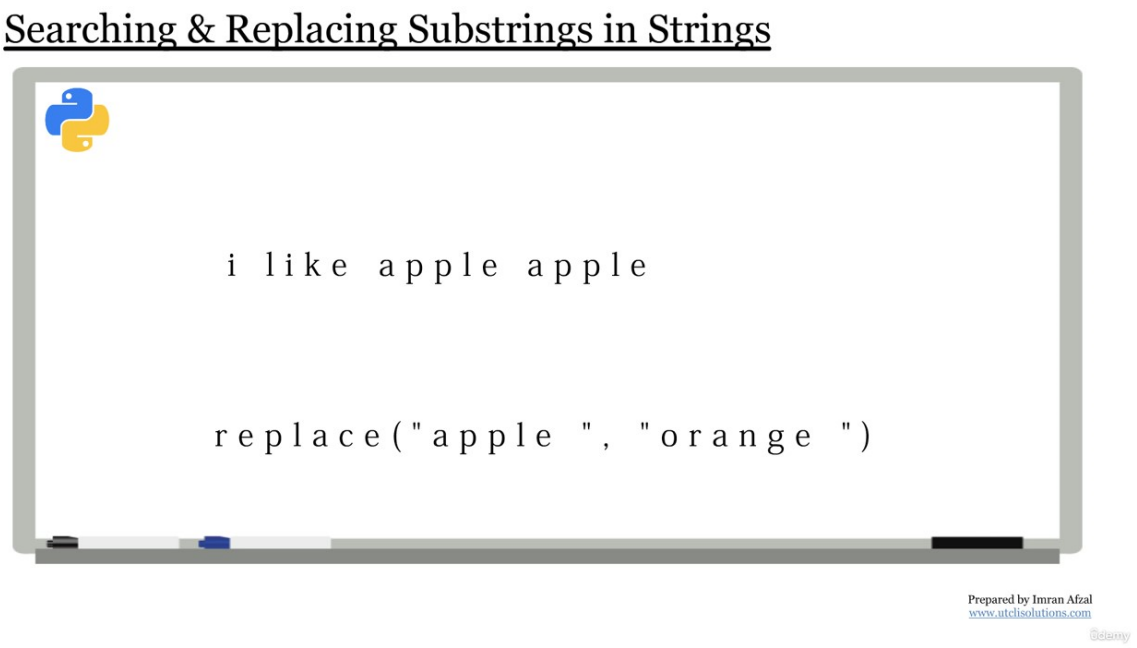
```
Python Code
my_string = "   Hello, World!   "
print(my_string.strip())  # Removes whitespace from both ends
```

Output:

```
Hello, World!
```

8.3. replace() دالة التعويض

- **replace(old, new)**: Replaces all occurrences of old substring with new.



Example:

```
Python Code
my_string = "I love programming in Java."
new_string = my_string.replace("Java", "Python")
print(new_string)
```

Output:

```
I love programming in Python.
```

8.4. split() and join() دوال التقسم و الدمج

- **split()**: Splits a string into a list based on a delimiter.
- **join()**: Joins a list of strings into a single string.

Example:

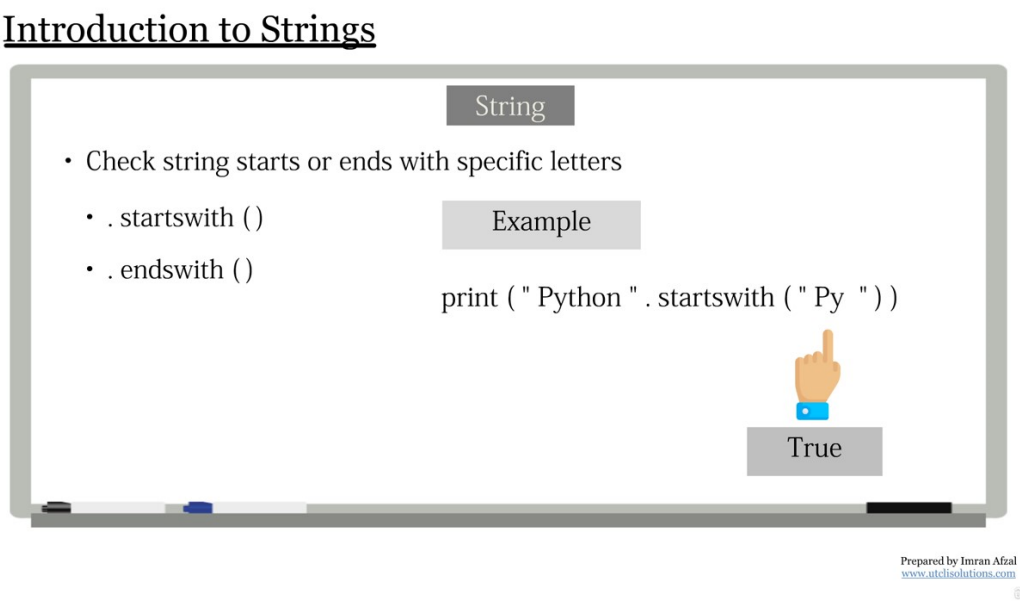
```
Python Code
my_string = "apple,banana,cherry"
fruits = my_string.split(",")  # Split the string by commas
print(fruits)

# Joining list of fruits back into a string
joined_string = " and ".join(fruits)
print(joined_string)
```

Output:

```
['apple', 'banana', 'cherry']
apple and banana and cherry
```

8.5. startswith() and endswith() دوال يبدأ ب و ينتهي ب

- **startswith(prefix)**: Returns True if the string starts with the given prefix.
- **endswith(suffix)**: Returns True if the string ends with the given suffix.



Example:

```
Python Code
my_string = "Hello, Python!"

print(my_string.startswith("Hello"))  # True
print(my_string.endswith("!"))        # True
print(my_string.startswith("Python")) # False
```

Output:

```
True
True
False
```

8.6. find() دالة البحث في النص

- **find(substring)**: Returns the index of the first occurrence of the substring. If not found, it returns -1.

Example:

```
Python Code
my_string = "I love Python programming."
index = my_string.find("Python")
print("Index of 'Python':", index)

index = my_string.find("Java")
print("Index of 'Java':", index)  # Substring not found
```

Output:

```
Index of 'Python': 7
```

Index of 'Java': -1

9. String Length طول النص

You can find the length of a string using the len() function.

Example:

Python Code

```
my_string = "Hello, World!"
length = len(my_string)
print("Length of the string:", length)
```

Output:

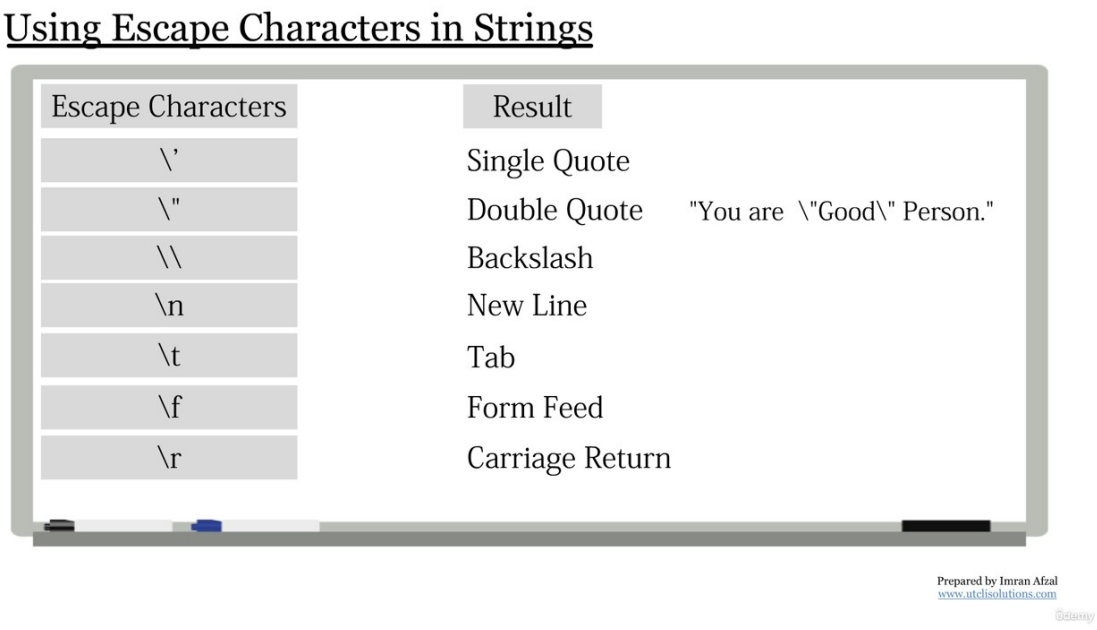
Length of the string: 13

Explanation:

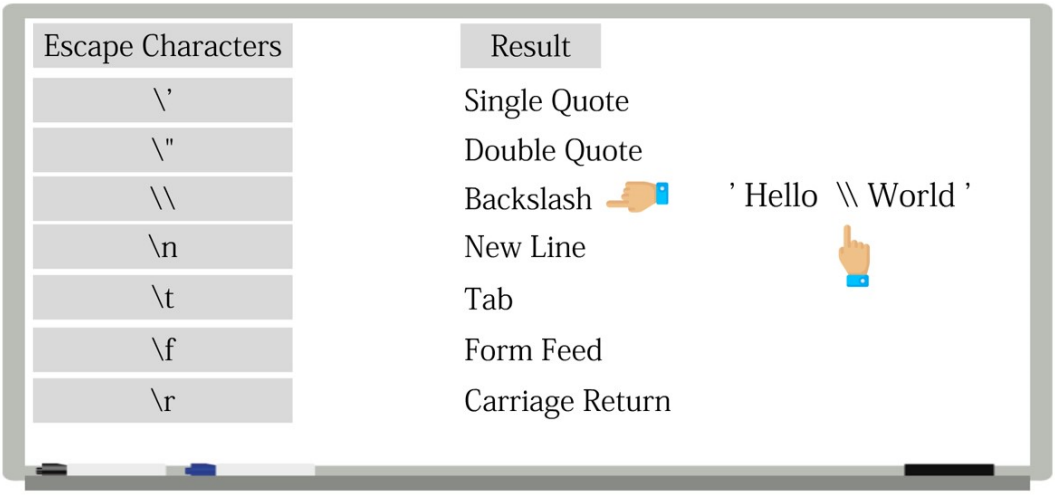
- The len() function returns the number of characters in the string, including spaces and punctuation.

10. Escape Characters(الأحرف الخاصة) أحرف الهروب

Escape characters allow you to include special characters in strings, such as newlines or quotation marks. Escape sequences start with a backslash (\).



Using Escape Characters in Strings



Example:

Python Code

```
# Using escape characters
string_with_newline = "Hello,\nWorld!"
string_with_tab = "Hello,\tWorld!"
string_with_quote = "She said, \"Python is great!\""

print(string_with_newline)
print(string_with_tab)
print(string_with_quote)
```

Output:

```
Hello,
World!
Hello,	World!
She said, "Python is great!"
```

Explanation:

- \n inserts a new line.
- \t inserts a tab.
- \" allows double quotes to be included in the string.

11. String Comparison مقارنة النصوص

You can compare strings using comparison operators (==, !=, <, >, etc.).

Example:

Python Code

```
string1 = "apple"
string2 = "banana"

print(string1 == string2)  # False
print(string1 != string2)  # True
```

print(string1 < string2)   # True (lexicographical comparison)

Output:

False
True
True

12. Multi-line Stringsنصوص متعددة الأسطر

Multi-line strings can be created using triple quotes (''' ''' or """ """).

Example:

python
Code
multi_line_string = """This is a
multi-line string
in Python."""
print(multi_line_string)

Output:

This is a
multi-line string
in Python.

Explanation:

- Triple quotes allow strings to span multiple lines.

REGULAR EXPRESIONالتعبيرات النمطية

What are Regular Expressions?

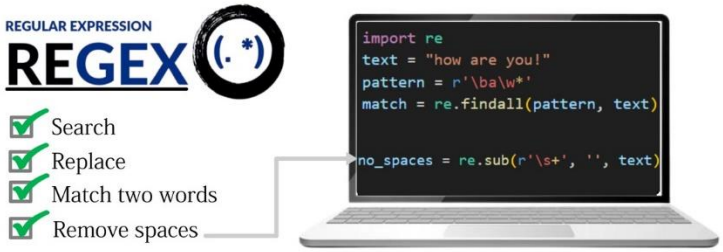
Regular expressions are **text patterns** used for searching, matching, and manipulating text. They're commonly used for validation, extraction, and text processing.

Basic Example

```python
import re

# Simple example to find a word in text
text = "Hello, my name is Ahmed and I live in Cairo"
pattern = r"Ahmed"

result = re.search(pattern, text)
```

```python
if result:
    print("Found name:", result.group())
else:
    print("Name not found")
```

String Manipulation Using Regular Expressions



Prepared by Imran Afzal
www.utclisolutions.com