String :

Indexing and Slicing in Python Strings

Indexing and Slicing Overview:

- **Indexing:**
  - Accesses a single character in a string using its position (index).
  - Indexes start from 0 for the first character and can be negative for counting from the end.
  - Example: `string[0]` gives the first character.
- **Slicing:**
  - Extracts a substring from the string using a range of indices.
  - The syntax is `string[start:stop:step]`, where:
    - `start` is the index where the slice begins (inclusive).
    - `stop` is the index where the slice ends (exclusive).
    - `step` specifies the stride between indices.
  - Example: `string[1:4]` gives characters from index 1 to 3.

Differences:

- **Indexing** retrieves individual characters, while **slicing** retrieves a substring.
- Indexing uses a single index value, whereas slicing uses start, stop, and step indices.
- Indexing is useful for accessing a specific character, while slicing is ideal for extracting parts of a string.

When to Use:

- **Indexing**: When you need a specific character, like extracting the first or last character.

- **Slicing**: When you need a part of the string, like extracting a word or phrase.

Practice Questions

1. **Indexing Practice:**
    - Given the string `s = "Hello, World!"`, what is the result of `s[7]`?
    - How can you access the last character of the string `s = "Python"` using indexing?
2. **Slicing Practice:**
    - For the string `s = "Programming"`, extract the substring "gram" using slicing.
    - How would you retrieve the string "World" from `s = "Hello, World!"`?
3. **Indexing and Slicing Combined:**
    - In the string `s = "Data Science"`, use slicing to get the word "Data" and indexing to get the first letter of "Science".
4. **Negative Indexing and Slicing:**
    - Given `s = "OpenAI"`, use negative indexing to get the character 'I'.
    - Using the same string `s = "OpenAI"`, extract the substring "pen" using negative slicing.
5. **Advanced Slicing:**
    - How can you reverse the string `s = "Python"` using slicing?
    - Given the string `s = "123456789"`, use slicing to get every second character starting from the second character.

Answers to Practice Questions:

1. **Indexing Practice:**
    - `s[7]` results in `'W'`.

- The last character can be accessed with `s[-1]`, resulting in `'n'`.

2. **Slicing Practice:**
   - The substring "gram" can be extracted using `s[3:7]`.
   - The string "World" can be retrieved with `s[7:12]`.
3. **Indexing and Slicing Combined:**
   - `s[:4]` gives "Data", and `s[5]` gives 'S'.
4. **Negative Indexing and Slicing:**
   - `s[-1]` results in 'l'.
   - The substring "pen" can be obtained with `s[-5:-2]`.
5. **Advanced Slicing:**
   - Reverse the string with `s[::-1]`, resulting in `"nohtyP"`.
   - Every second character starting from the second can be extracted with `s[1::2]`, giving `"2468"`.

Complex Questions on Indexing and Slicing

1. **Extract Alternating Characters and Reverse:**
   - Given the string `s = "abcdefghijklm"`, write a code snippet to:
     - Extract every second character starting from the first character.
     - Reverse the resulting substring.

   Answer:

   - Extracting every second character: `s[::2]` results in `"acegikm"`.
   - Reversing the substring: `s[::2][::-1]` results in `"mkgieca"`.
2. **Nested Slicing:**
   - With `s = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"`, extract every third letter starting from the second letter and then reverse this result.

Answer:

- First, slice to get every third letter starting from the second: `s[1::3]` gives `"BDFHJLNPRTXVZ"`.
- Then reverse: `s[1::3][::-1]` results in `"ZVXTRPNLJHFD"`.

3. **Manipulating Strings with Negative Indices:**
4.
   - For the string `s = "PythonIsFun"`, use slicing to:
     - Extract the substring "Is".
     - Replace "Is" with "Was" using slicing and concatenation.

   Answer:

   - Extract "Is": `s[6:8]` results in `"Is"`.
   - Replace "Is" with "Was": `s[:6] + "Was" + s[8:]` results in `"PythonWasFun"`.

5. **Complex Slicing with Steps:**
   - Given `s = "1234567890"`, how can you:
     - Extract the odd-indexed characters.
     - Then reverse the extracted substring and extract every second character.

   Answer:

   - Odd-indexed characters: `s[1::2]` results in `"24680"`.
   - Reverse and extract every second character: `s[1::2][::-1][::2]` results in `"086"`.

6. **Using Indexing and Slicing Together:**
   - For the string `s = "WelcomeToPythonProgramming"`, how can you:
     - Extract "Python" and "Programming" separately.
     - Then create a new string combining these two with a space in between.

Methods in string :

1. String Methods

Case Conversion

- `.lower()`: Converts all characters to lowercase.

  ```
  pythonCopy code
  result = "Hello".lower()
  print(result)  # Output: hello
  ```

- `.upper()`: Converts all characters to uppercase.

  ```
  pythonCopy code
  result = "Hello".upper()
  print(result)  # Output: HELLO
  ```

- `.capitalize()`: Converts the first character to uppercase and the rest to lowercase.

  ```
  pythonCopy code
  result = "hello world".capitalize()
  print(result)  # Output: Hello world
  ```

- `.title()`: Converts the first character of each word to uppercase.

```
pythonCopy code
result = "hello world".title()
print(result)  # Output: Hello World
```

- `.swapcase()`: Swaps uppercase characters to lowercase and vice versa.

```
pythonCopy code
result = "Hello World".swapcase()
print(result)  # Output: hELLO wORLD
```

String Checking

- `.isalnum()`: Returns `True` if all characters are alphanumeric.

```
pythonCopy code
result = "Hello123".isalnum()
print(result)  # Output: True
```

- `.isalpha()`: Returns `True` if all characters are alphabetic.

```
pythonCopy code
result = "Hello".isalpha()
print(result)  # Output: True
```

- `.isdigit()`: Returns `True` if all characters are digits.

```
pythonCopy code
result = "123".isdigit()
print(result)  # Output: True
```

- `.islower()`: Returns `True` if all characters are lowercase.

```
pythonCopy code
result = "hello".islower()
print(result)  # Output: True
```

- `.isupper()`: Returns `True` if all characters are uppercase.

```
pythonCopy code
result = "HELLO".isupper()
print(result)  # Output: True
```

- `.isspace()`: Returns `True` if all characters are whitespace.

```
pythonCopy code
result = "   ".isspace()
print(result)  # Output: True
```

String Manipulation

- `.strip()`: Removes leading and trailing whitespace.

```
pythonCopy code
result = "   Hello   ".strip()
print(result)  # Output: Hello
```

- `.lstrip()`: Removes leading whitespace.

```
pythonCopy code
result = "   Hello   ".lstrip()
```

```
print(result)  # Output: Hello
```

- `.rstrip()`: Removes trailing whitespace.

```
pythonCopy code
result = "   Hello   ".rstrip()
print(result)  # Output:    Hello
```

- `.replace(old, new)`: Replaces occurrences of a substring with another.

```
pythonCopy code
result = "Hello World".replace("World", "Python")
print(result)  # Output: Hello Python
```

- `.split(separator)`: Splits the string into a list using a specified separator.

```
pythonCopy code
result = "Hello World".split()
print(result)  # Output: ['Hello', 'World']
```

- `.join(iterable)`: Joins elements of an iterable into a string, using the string as a separator.

```
pythonCopy code
result = "-".join(["Hello", "World"])
print(result)  # Output: Hello-World
```

Searching

- `.find(substring)`: Returns the index of the first occurrence of the substring.

```
pythonCopy code
result = "Hello World".find("World")
print(result)  # Output: 6
```

- `.rfind(substring)`: Returns the index of the last occurrence of the substring.

```
pythonCopy code
result = "Hello World World".rfind("World")
print(result)  # Output: 12
```

- `.index(substring)`: Similar to `.find()`, but raises a `ValueError` if the substring is not found.

```
pythonCopy code
result = "Hello World".index("World")
print(result)  # Output: 6
```

- `.rindex(substring)`: Similar to `.rfind()`, but raises a `ValueError` if the substring is not found.

```
pythonCopy code
result = "Hello World World".rindex("World")
print(result)  # Output: 12
```

Formatting

- `.format(*args, **kwargs)`: Formats the string using placeholders.

```
pythonCopy code
result = "Hello, {}!".format("World")
```

```
print(result)  # Output: Hello, World!
```

- `f-strings`: A newer way to format strings, introduced in Python 3.6.

```
pythonCopy code
name = "World"
result = f"Hello, {name}!"
print(result)  # Output: Hello, World!
```

2. Built-in Functions

- `len()`: Returns the length of the string.

```
pythonCopy code
result = len("Hello")
print(result)  # Output: 5
```

- `str()`: Converts an object to a string.

```
pythonCopy code
result = str(123)
print(result)  # Output: '123'
```

- `ord()`: Returns the Unicode code point of a single character.

```
pythonCopy code
result = ord('A')
print(result)  # Output: 65
```

- `chr()`: Returns the character corresponding to a Unicode code point.

```
pythonCopy code
result = chr(65)
print(result)  # Output: 'A'
```