STRINGS

Indexing and Slicing

Indexing

Indexing is how you access individual characters in a string by their position. Python strings are **zero-indexed**, meaning the first character is at index `0`. You can also use **negative indexing**, where `-1` refers to the last character, `-2` to the second-to-last, and so on.

Python
```
my_string = "Python"

# Positive indexing
print(my_string[0]) # Output: P

# Negative indexing
print(my_string[-1]) # Output: n
```
Accessing an index that is out of range will cause an `IndexError`.

Slicing

Slicing lets you extract a portion of a string (a "substring") by specifying a range. The syntax is `string[start:end:step]`.

- **start:** The index where the slice begins (inclusive). If omitted, it defaults to the beginning.
- **end:** The index where the slice ends (exclusive). If omitted, it defaults to the end.
- **step:** An optional number indicating the interval between characters. The default is `1`. A step of `-1` can be used to reverse the string.

Python
```
my_string = "Programming"

# Extract from index 3 up to (but not including) index 7
print(my_string[3:7])   # Output: gram

# Extract from index 4 to the end
print(my_string[4:])    # Output: ramming

# Extract from the beginning up to index 4
print(my_string[:4])    # Output: Prog

# Get every second character
print(my_string[::2])   # Output: Pormig

# Reverse the string
print(my_string[::-1])  # Output: gnimmargorP
```
Slicing with an out-of-bounds index doesn't cause an error; it simply returns the portion of the string that is within bounds.

Common String Operations and Methods

Case Conversion

- .upper(): Converts the entire string to uppercase.
- .lower(): Converts the entire string to lowercase.
- .title(): Capitalizes the first letter of each word.
- .capitalize(): Capitalizes only the first letter of the entire string.
- .swapcase(): Swaps the case of all letters (upper becomes lower and vice versa).

Python
```
text = "hello WORLD"

print(text.upper())      # Output: HELLO WORLD
print(text.lower())      # Output: hello world
print(text.title())      # Output: Hello World
print(text.capitalize()) # Output: Hello world
```

Joining and Splitting

- **Concatenation (+)**: Joins two or more strings together.
- **Repetition (*)**: Repeats a string a specified number of times.
- .split(delimiter): Splits a string into a list of smaller strings based on a delimiter. If no delimiter is provided, it splits by whitespace.

Python
```
greeting = "Hello"
name = "World"

# Concatenation
print(greeting + ", " + name + "!") # Output: Hello, World!

# Repetition
print(greeting * 3) # Output: HelloHelloHello

# Splitting
sentence = "this is a sentence"
print(sentence.split()) # Output: ['this', 'is', 'a', 'sentence']
```

Finding and Replacing

- .replace(old, new): Returns a new string where all occurrences of old are replaced with new.
- .find(substring): Returns the starting index of the first occurrence of a substring. Returns -1 if the substring is not found.
- .index(substring): Similar to .find(), but raises a ValueError if the substring is not found.
- .count(substring): Returns the number of times a substring appears in the string.

Python
```
sentence = "I like apples, apples are great."

print(sentence.replace("apples", "oranges")) # Output: I like oranges, oranges are great.
print(sentence.find("apples"))               # Output: 7
print(sentence.count("apples"))              # Output: 2
```

Trimming and Formatting

- `.strip(chars)`: Removes leading and trailing characters. By default, it removes whitespace.
- **f-strings**: A modern and easy way to format strings by embedding expressions inside string literals, prefixed with an `f`.

```python
# Stripping whitespace
text = "   some text   "
print(text.strip()) # Output: some text

# Formatting with an f-string
name = "Alice"
age = 30
print(f"My name is {name} and I am {age} years old.") # Output: My name is Alice and I am 30 years old.
```

Checking Content (Boolean Methods)

These methods check the string's content and return `True` or `False`.

- `.startswith(prefix)`: Checks if the string begins with a specific prefix.
- `.isalpha()`: `True` if all characters are letters.
- `.isnumeric()`: `True` if all characters are numeric.
- `.isalnum()`: `True` if all characters are letters or numbers.
- `.isupper()` / `.islower()`: `True` if all letters are uppercase or lowercase, respectively.

```python
text1 = "Python"
text2 = "123"

print(text1.isalpha())   # Output: True
print(text2.isnumeric()) # Output: True
print(text1.isupper())   # Output: False
```