Python Strings

Complete Interview Guide

Break, Continue, Pass & String Operations

With Code Examples and Outputs

1. Break, Continue, Pass Statements

Control flow statements that alter loop execution: break exits the loop immediately, continue skips the current iteration and moves to the next, pass is a null statement that does nothing (placeholder for future code).

```
# Example 1: break in for loop
for i in range(1, 11):
    if i == 5:
        break
    print(i, end=' ')
print('\nLoop stopped')
```

```
Output: 1 2 3 4
Loop stopped
```

```
# Example 2: continue in for loop
for i in range(1, 6):
    if i == 3:
        continue
    print(i, end=' ')
```

```
Output: 1 2 4 5
```

```
# Example 3: pass as placeholder
for i in range(3):
    pass  # TODO: implement later
print('Nothing executed')
```

```
Output: Nothing executed
```

```
# Example 4: break in while loop
count = 1
while count <= 10:
    if count == 6:
        break
    print(count, end=' ')
    count += 1
```

```
Output: 1 2 3 4 5
```

```
# Example 5: continue to skip even numbers
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(i, end=' ')
```

```
Output: 1 3 5 7 9
```

```
# Example 6: break in nested loop
for i in range(1, 4):
    for j in range(1, 4):
        if j == 2:
            break
        print(f'({i},{j})', end=' ')
    print()
```

```
Output: (1,1)
(2,1)
(3,1)
```

```
# Example 7: Search and break
numbers = [10, 20, 30, 40, 50]
target = 30
for num in numbers:
    if num == target:
        print(f'Found: {num}')
        break
else:
    print('Not found')
```

```
Output: Found: 30
```

```
# Example 8: continue with string processing
text = 'Python123'
for char in text:
    if char.isdigit():
        continue
    print(char, end='')
```

```
Output: Python
```

```
# Example 9: pass in function definition
def future_function():
    pass  # Will implement later

future_function()
print('Function called')
```

```
Output: Function called
```

```
# Example 10: Multiple conditions with break
for i in range(1, 21):
    if i % 3 == 0 and i % 5 == 0:
        print(f'Found: {i}')
        break
    print(i, end=' ')
```

```
Output: 1 2 4 5 7 8 10 11 13 14
Found: 15
```

2. Create a Python String

Strings are sequences of characters enclosed in quotes (single, double, or triple). They are immutable, meaning once created, they cannot be changed. Strings support various operations and methods for text manipulation.

```
# Example 1: Single quotes
str1 = 'Hello World'
print(str1)
print(type(str1))
```

```
Output: Hello World
```

```
# Example 2: Double quotes
str2 = "Python Programming"
print(str2)
```

```
Output: Python Programming
```

```
# Example 3: Triple quotes (multiline)
str3 = '''This is a
multiline
string'''
print(str3)
```

```
Output: This is a
multiline
string
```

```
# Example 4: Empty string
empty = ''
print(f'Length: {len(empty)}')
print(f'Is empty: {len(empty) == 0}')
```

```
Output: Length: 0
Is empty: True
```

```
# Example 5: String with quotes inside
str4 = "He said, 'Hello!'"
print(str4)
```

```
Output: He said, 'Hello!'
```

```
# Example 6: Escape characters
str5 = 'Line1\nLine2\tTabbed'
print(str5)
```

```
Output: Line1
Line2 Tabbed
```

```
# Example 7: Raw string (ignores escape)
path = r'C:\Users\name\folder'
print(path)
```

```
Output: C:\Users\name\folder
```

```
# Example 8: String concatenation
first = 'Hello'
last = 'World'
full = first + ' ' + last
print(full)
```

```
Output: Hello World
```

```
# Example 9: String repetition
str6 = 'Ha' * 3
print(str6)
```

```
Output: HaHaHa
```

```
# Example 10: String from other types
num = 123
str7 = str(num)
print(f'Value: {str7}, Type: {type(str7)}')
```

```
Output: Value: 123, Type:
```

3. Indexing / Negative Indexing

String indexing accesses individual characters. Positive indexing starts from 0 (left), negative indexing starts from -1 (right). Index out of range raises IndexError.

```
# Example 1: Positive indexing
text = 'Python'
print(f'First char: {text[0]}')
print(f'Third char: {text[2]}')
```

```
Output: First char: P
Third char: t
```

```
# Example 2: Negative indexing
text = 'Python'
print(f'Last char: {text[-1]}')
print(f'Second last: {text[-2]}')
```

```
Output: Last char: n
Second last: o
```

```
# Example 3: Get first and last
word = 'Programming'
print(f'First: {word[0]}, Last: {word[-1]}')
```

```
Output: First: P, Last: g
```

```
# Example 4: Index all characters
text = 'Hello'
for i in range(len(text)):
    print(f'Index {i}: {text[i]}')
```

```
Output: Index 0: H
Index 1: e
Index 2: l
Index 3: l
Index 4: o
```

```
# Example 5: Negative index demonstration
text = 'ABCDE'
print(f'{text[-1]} = {text[len(text)-1]}')
print(f'{text[-3]} = {text[len(text)-3]}')
```

```
Output: E = E
C = C
```

```
# Example 6: Check character at position
email = 'user@example.com'
if email[4] == '@':
    print('@ found at index 4')
```

```
Output: @ found at index 4
```

```
# Example 7: Access middle character
word = 'Python'
mid = len(word) // 2
print(f'Middle char: {word[mid]}')
```

```
Output: Middle char: h
```

```
# Example 8: Compare characters
str1 = 'Apple'
str2 = 'Apricot'
if str1[0] == str2[0]:
    print(f'Both start with: {str1[0]}')
```

```
Output: Both start with: A
```

```
# Example 9: Extract extension from filename
filename = 'document.pdf'
ext_start = filename.index('.')
print(f'Extension: {filename[ext_start:]}')
```

```
Output: Extension: .pdf
```

```
# Example 10: Reverse using negative index
text = 'Hello'
reversed_text = ''
for i in range(1, len(text) + 1):
    reversed_text += text[-i]
print(reversed_text)
```

```
Output: olleH
```

4. Slicing

Slicing extracts a portion of a string using [start:end:step]. Start is inclusive, end is exclusive. Negative indices and steps allowed. Essential for string manipulation.

```
# Example 1: Basic slicing
text = 'Python Programming'
print(text[0:6])  # Characters 0 to 5
print(text[7:18])  # Characters 7 to 17
```

```
Output: Python
Programming
```

```
# Example 2: Omit start or end
text = 'Hello World'
print(text[:5])   # From start to 5
print(text[6:])   # From 6 to end
```

```
Output: Hello
World
```

```
# Example 3: Negative indices in slicing
text = 'Python'
print(text[-6:-3])  # Pyt
print(text[-3:])    # hon
```

```
Output: Pyt
hon
```

```
# Example 4: Step in slicing
text = 'Python'
print(text[::2])   # Every 2nd char
print(text[1::2])  # Every 2nd from index 1
```

```
Output: Pto
yhn
```

```
# Example 5: Reverse string using slicing
text = 'Hello'
reversed_text = text[::-1]
print(reversed_text)
```

```
Output: olleH
```

```
# Example 6: Extract substring
email = 'user@example.com'
at_pos = email.index('@')
username = email[:at_pos]
domain = email[at_pos+1:]
print(f'User: {username}, Domain: {domain}')
```

```
Output: User: user, Domain: example.com
```

```
# Example 7: Get every nth character
text = 'ABCDEFGHIJ'
print(text[::3])  # Every 3rd character
```

```
Output: ADGJ
```

```
# Example 8: Middle portion
text = 'Programming'
mid_start = len(text) // 4
mid_end = 3 * len(text) // 4
print(text[mid_start:mid_end])
```

```
Output: ogram
```

```
# Example 9: Remove first and last
text = '"Hello"'
cleaned = text[1:-1]
print(cleaned)
```

```
Output: Hello
```

```
# Example 10: Palindrome check using slicing
word = 'radar'
if word == word[::-1]:
    print(f'{word} is palindrome')
else:
    print(f'{word} is not palindrome')
```

```
Output: radar is palindrome
```

5. Edit & Delete a String

Strings are immutable in Python - you cannot change them in place. To 'edit', create a new string. You can delete entire string variables using del, but cannot delete individual characters.

```
# Example 1: Strings are immutable
text = 'Hello'
# text[0] = 'h'  # This raises TypeError
text = 'h' + text[1:]  # Create new string
print(text)
```

```
Output: hello
```

```
# Example 2: Replace character
word = 'Python'
new_word = word.replace('P', 'J')
print(f'Original: {word}')
print(f'Modified: {new_word}')
```

```
Output: Original: Python
Modified: Jython
```

```
# Example 3: Delete entire string
text = 'Hello World'
del text
# print(text)  # NameError: name 'text' not defined
print('String deleted')
```

```
Output: String deleted
```

```
# Example 4: Update string by concatenation
greeting = 'Hello'
greeting = greeting + ' World'
print(greeting)
```

```
Output: Hello World
```

```
# Example 5: Remove character by slicing
text = 'Python'
# Remove 'y' at index 1
new_text = text[:1] + text[2:]
print(new_text)
```

```
Output: Pthon
```

```
# Example 6: Update using join
text = 'Hello World'
words = text.split()
words[0] = 'Hi'
new_text = ' '.join(words)
print(new_text)
```

```
Output: Hi World
```

```
# Example 7: Clear string (reassign empty)
name = 'Python'
name = ''
print(f'Length: {len(name)}')
```

```
Output: Length: 0
```

```
# Example 8: Insert character
text = 'Pyton'
# Insert 'h' at index 3
new_text = text[:3] + 'h' + text[3:]
print(new_text)
```

```
Output: Python
```

```
# Example 9: Multiple replacements
text = 'Hello World Hello'
text = text.replace('Hello', 'Hi')
print(text)
```

```
Output: Hi World Hi
```

```
# Example 10: Remove spaces
text = '  Python  '
text = text.strip()
print(f'|{text}|')
```

```
Output: |Python|
```

6. Operations on a String

String operations include concatenation (+), repetition (*), membership (in/not in), comparison (==, !=, <, >), length (len()), and iteration. These are fundamental operations.

```
# Example 1: Concatenation
first = 'Hello'
last = 'World'
full = first + ' ' + last
print(full)
```

```
Output: Hello World
```

```
# Example 2: Repetition
str1 = 'Python '
repeated = str1 * 3
print(repeated)
```

```
Output: Python Python Python
```

```
# Example 3: Membership - in
text = 'Python Programming'
if 'Python' in text:
    print('Found')
else:
    print('Not found')
```

```
Output: Found
```

```
# Example 4: Membership - not in
text = 'Hello World'
if 'Java' not in text:
    print('Java not present')
```

```
Output: Java not present
```

```
# Example 5: String length
text = 'Python'
length = len(text)
print(f'Length: {length}')
```

```
Output: Length: 6
```

```
# Example 6: String comparison
str1 = 'apple'
str2 = 'banana'
if str1 < str2:
    print(f'{str1} comes before {str2}')
```

```
Output: apple comes before banana
```

```
# Example 7: Iterate through string
text = 'Python'
for char in text:
    print(char, end=' ')
```

```
Output: P y t h o n
```

```
# Example 8: Check substring
email = 'user@example.com'
if '@' in email and '.' in email:
    print('Valid email format')
```

```
Output: Valid email format
```

```
# Example 9: Count occurrences manually
text = 'hello world'
char = 'l'
count = 0
for c in text:
    if c == char:
        count += 1
print(f"'{char}' appears {count} times")
```

```
Output: 'l' appears 3 times
```

```
# Example 10: String multiplication for patterns
pattern = '* ' * 5
print(pattern)
```

```
Output: * * * * *
```

7. Common String Functions

Python provides built-in string methods for common operations: len() for length, type() for type checking, str() for conversion, ord() for ASCII value, chr() for character from ASCII, and more.

```
# Example 1: len() - string length
text = 'Python Programming'
print(f'Length: {len(text)}')
```

```
Output: Length: 18
```

```
# Example 2: type() - check data type
text = 'Hello'
print(type(text))
```

```
Output:
```

```
# Example 3: str() - convert to string
num = 123
str_num = str(num)
print(f'{str_num} is {type(str_num)}')
```

```
Output: 123 is
```

```
# Example 4: ord() - ASCII value
char = 'A'
ascii_val = ord(char)
print(f'ASCII of {char}: {ascii_val}')
```

```
Output: ASCII of A: 65
```

```
# Example 5: chr() - character from ASCII
ascii_val = 65
char = chr(ascii_val)
print(f'Character: {char}')
```

```
Output: Character: A
```

```
# Example 6: min() - smallest character
text = 'Python'
print(f'Min: {min(text)}')
```

```
Output: Min: P
```

```
# Example 7: max() - largest character
text = 'Python'
print(f'Max: {max(text)}')
```

```
Output: Max: y
```

```
# Example 8: sorted() - sort characters
text = 'python'
sorted_chars = sorted(text)
print(sorted_chars)
```

```
Output: ['h', 'n', 'o', 'p', 't', 'y']
```

```
# Example 9: reversed() - reverse string
text = 'Hello'
reversed_text = ''.join(reversed(text))
print(reversed_text)
```

```
Output: olleH
```

```
# Example 10: any() and all() with conditions
text = 'Python123'
has_digit = any(c.isdigit() for c in text)
all_alpha = all(c.isalpha() for c in text)
print(f'Has digit: {has_digit}')
print(f'All alphabetic: {all_alpha}')
```

```
Output: Has digit: True
All alphabetic: False
```

8. Capitalize/Title/Upper/Lower/Swapcase Functions

Case conversion methods: capitalize() - first char uppercase; title() - first char of each word uppercase; upper() - all uppercase; lower() - all lowercase; swapcase() - swap case.

```
# Example 1: capitalize() - first char uppercase
text = 'python programming'
print(text.capitalize())
```

```
Output: Python programming
```

```
# Example 2: title() - title case
text = 'python programming language'
print(text.title())
```

```
Output: Python Programming Language
```

```
# Example 3: upper() - all uppercase
text = 'hello world'
print(text.upper())
```

```
Output: HELLO WORLD
```

```
# Example 4: lower() - all lowercase
text = 'PYTHON PROGRAMMING'
print(text.lower())
```

```
Output: python programming
```

```
# Example 5: swapcase() - swap case
text = 'Python Programming'
print(text.swapcase())
```

```
Output: pYTHON pROGRAMMING
```

```
# Example 6: Case-insensitive comparison
str1 = 'Python'
str2 = 'PYTHON'
if str1.lower() == str2.lower():
    print('Strings are equal (case-insensitive)')
```

```
Output: Strings are equal (case-insensitive)
```

```
# Example 7: Format name properly
name = 'john DOE'
formatted = name.title()
print(f'Formatted: {formatted}')
```

```
Output: Formatted: John Doe
```

```
# Example 8: Convert to lowercase for processing
email = 'User@Example.COM'
email = email.lower()
print(email)
```

```
Output: user@example.com
```

```
# Example 9: First letter of sentence
sentence = 'hello. how are you?'
sentences = sentence.split('. ')
formatted = '. '.join([s.capitalize() for s in sentences])
print(formatted)
```

```
Output: Hello. How are you?
```

```
# Example 10: Username normalization
username = 'JohnDoe123'
normalized = username.lower()
print(f'Username: {normalized}')
```

```
Output: Username: johndoe123
```

9. Count/Find/Index Functions

Search methods: count() - count occurrences; find() - find first occurrence (returns -1 if not found); index() - like find() but raises ValueError if not found; rfind() and rindex() search from right.

```
# Example 1: count() - count occurrences
text = 'hello world hello'
count = text.count('hello')
print(f'"hello" appears {count} times')
```

```
Output: "hello" appears 2 times
```

```
# Example 2: count() with substring
text = 'Python Programming'
count = text.count('o')
print(f'"o" appears {count} times')
```

```
Output: "o" appears 2 times
```

```
# Example 3: find() - find position
text = 'Python Programming'
pos = text.find('Programming')
print(f'Found at index: {pos}')
```

```
Output: Found at index: 7
```

```
# Example 4: find() - not found
text = 'Python'
pos = text.find('Java')
print(f'Position: {pos}')
```

```
Output: Position: -1
```

```
# Example 5: index() - find position
text = 'Hello World'
pos = text.index('World')
print(f'"World" at index: {pos}')
```

```
Output: "World" at index: 6
```

```
# Example 6: rfind() - find from right
text = 'hello world hello'
pos = text.rfind('hello')
print(f'Last occurrence at: {pos}')
```

```
Output: Last occurrence at: 12
```

```
# Example 7: Find with start and end
text = 'Python Programming Python'
pos = text.find('Python', 10)  # Start from index 10
print(f'Found at: {pos}')
```

```
Output: Found at: 19
```

```
# Example 8: Count in specific range
text = 'abcabcabc'
count = text.count('abc', 3, 9)  # From index 3 to 9
print(f'Count: {count}')
```

```
Output: Count: 2
```

```
# Example 9: Validate string contains substring
url = 'https://www.example.com'
if url.find('https') != -1:
    print('Secure URL')
else:
    print('Not secure')
```

```
Output: Secure URL
```

```
# Example 10: Extract domain from email
email = 'user@example.com'
at_pos = email.index('@')
domain = email[at_pos+1:]
print(f'Domain: {domain}')
```

```
Output: Domain: example.com
```

10. endswith() / startswith() Functions

These methods check string prefixes and suffixes: startswith() checks if string starts with specified prefix; endswith() checks if string ends with specified suffix. Both return Boolean and support tuple of strings and optional start/end positions.

```
# Example 1: startswith() basic
text = 'Python Programming'
if text.startswith('Python'):
    print('Starts with Python')
```

```
Output: Starts with Python
```

```
# Example 2: endswith() basic
filename = 'document.pdf'
if filename.endswith('.pdf'):
    print('PDF file')
```

```
Output: PDF file
```

```
# Example 3: startswith() with tuple
text = 'Hello World'
if text.startswith(('Hello', 'Hi')):
    print('Greeting found')
```

```
Output: Greeting found
```

```
# Example 4: endswith() with tuple
filename = 'image.jpg'
if filename.endswith(('.jpg', '.png', '.gif')):
    print('Image file')
```

```
Output: Image file
```

```
# Example 5: Case-sensitive check
text = 'Python'
print(text.startswith('python'))  # False
print(text.lower().startswith('python'))  # True
```

```
Output: False
True
```

```
# Example 6: Check URL protocol
url = 'https://www.example.com'
if url.startswith('https://'):
    print('Secure connection')
elif url.startswith('http://'):
    print('Unsecure connection')
```

```
Output: Secure connection
```

```
# Example 7: File extension validation
filename = 'script.py'
if filename.endswith('.py'):
    print('Python file')
else:
    print('Not a Python file')
```

```
Output: Python file
```

```
# Example 8: startswith() with position
text = 'Hello World'
if text.startswith('World', 6):
    print('World starts at index 6')
```

```
Output: World starts at index 6
```

```
# Example 9: Filter files by extension
files = ['doc.txt', 'image.jpg', 'data.csv', 'notes.txt']
text_files = [f for f in files if f.endswith('.txt')]
print(text_files)
```

```
Output: ['doc.txt', 'notes.txt']
```

```
# Example 10: Check command prefix
command = '/start game'
if command.startswith('/'):
    cmd = command[1:].split()[0]
    print(f'Command: {cmd}')
```

```
Output: Command: start
```

11. format() Function

The format() method formats strings by replacing placeholders {} with values. Supports positional and keyword arguments, formatting specifications for numbers, alignment, padding, and more. Modern alternative is f-strings (Python 3.6+).

```
# Example 1: Basic format()
name = 'Alice'
age = 25
text = 'Name: {}, Age: {}'.format(name, age)
print(text)
```

```
Output: Name: Alice, Age: 25
```

```
# Example 2: Positional arguments
text = '{0} is {1} years old. {0} is a student.'.format('Bob', 20)
print(text)
```

```
Output: Bob is 20 years old. Bob is a student.
```

```
# Example 3: Keyword arguments
text = '{name} scored {marks} marks'.format(name='John', marks=95)
print(text)
```

```
Output: John scored 95 marks
```

```
# Example 4: Number formatting
pi = 3.14159
text = 'Pi: {:.2f}'.format(pi)
print(text)
```

```
Output: Pi: 3.14
```

```
# Example 5: Integer formatting with leading zeros
num = 42
text = 'Number: {:05d}'.format(num)
print(text)
```

```
Output: Number: 00042
```

```
# Example 6: Alignment - left, right, center
text1 = '{:<10}'.format('Left')
text2 = '{:>10}'.format('Right')
text3 = '{:^10}'.format('Center')
print(f'|{text1}|')
print(f'|{text2}|')
print(f'|{text3}|')
```

```
Output: |Left      |
|     Right|
|  Center  |
```

```
# Example 7: Percentage formatting
ratio = 0.856
text = 'Success rate: {:.1%}'.format(ratio)
print(text)
```

```
Output: Success rate: 85.6%
```

```
# Example 8: Thousand separator
value = 1234567
text = 'Amount: {:,}'.format(value)
print(text)
```

```
Output: Amount: 1,234,567
```

```
# Example 9: F-string alternative (modern)
name = 'Python'
version = 3.11
text = f'{name} version {version}'
print(text)
```

```
Output: Python version 3.11
```

```
# Example 10: Binary, octal, hex formatting
num = 255
print('Decimal: {0:d}'.format(num))
print('Binary: {0:b}'.format(num))
print('Octal: {0:o}'.format(num))
print('Hex: {0:x}'.format(num))
```

```
Output: Decimal: 255
Binary: 11111111
Octal: 377
Hex: ff
```

12. isalnum/isalpha/isdigit/isidentifier Functions

Character classification methods: isalnum() - alphanumeric; isalpha() - alphabetic; isdigit() - digits; isidentifier() - valid Python identifier; isnumeric(), isdecimal() for numeric checks. All return Boolean.

```
# Example 1: isalnum() - alphanumeric
text1 = 'Python3'
text2 = 'Python 3'
print(f'{text1}.isalnum(): {text1.isalnum()}')
print(f'{text2}.isalnum(): {text2.isalnum()}')
```

```
Output: Python3.isalnum(): True
Python 3.isalnum(): False
```

```
# Example 2: isalpha() - alphabetic only
text1 = 'Python'
text2 = 'Python3'
print(f'{text1}.isalpha(): {text1.isalpha()}')
print(f'{text2}.isalpha(): {text2.isalpha()}')
```

```
Output: Python.isalpha(): True
Python3.isalpha(): False
```

```
# Example 3: isdigit() - digits only
text1 = '12345'
text2 = '123.45'
print(f'{text1}.isdigit(): {text1.isdigit()}')
print(f'{text2}.isdigit(): {text2.isdigit()}')
```

```
Output: 12345.isdigit(): True
123.45.isdigit(): False
```

```
# Example 4: isidentifier() - valid variable name
var1 = 'my_var'
var2 = '2nd_var'
print(f'{var1}.isidentifier(): {var1.isidentifier()}')
print(f'{var2}.isidentifier(): {var2.isidentifier()}')
```

```
Output: my_var.isidentifier(): True
2nd_var.isidentifier(): False
```

```
# Example 5: Validate username
username = 'user123'
if username.isalnum():
    print('Valid username')
else:
    print('Username can only contain letters and numbers')
```

```
Output: Valid username
```

```
# Example 6: Check if all characters are letters
name = 'John'
if name.isalpha():
    print('Valid name (letters only)')
else:
    print('Name contains non-letter characters')
```

```
Output: Valid name (letters only)
```

```
# Example 7: Validate PIN code
pin = '123456'
if pin.isdigit() and len(pin) == 6:
    print('Valid PIN')
else:
    print('Invalid PIN')
```

```
Output: Valid PIN
```

```
# Example 8: Check if string is valid identifier
variable = 'my_variable_1'
if variable.isidentifier():
    print(f'{variable} can be used as variable name')
```

```
Output: my_variable_1 can be used as variable name
```

```
# Example 9: Password strength check
password = 'Pass123'
has_alpha = any(c.isalpha() for c in password)
has_digit = any(c.isdigit() for c in password)
if has_alpha and has_digit:
    print('Password contains letters and digits')
```

```
Output: Password contains letters and digits
```

```
# Example 10: Filter alphanumeric characters
text = 'Hello@World#2024!'
filtered = ''.join([c for c in text if c.isalnum()])
print(filtered)
```

```
Output: HelloWorld2024
```

13. split() & join() Functions

split() divides a string into a list using a delimiter (default: whitespace). join() combines list elements into a string with specified separator. These are complementary operations essential for text processing.

```
# Example 1: split() with default delimiter
text = 'Python is awesome'
words = text.split()
print(words)
```

```
Output: ['Python', 'is', 'awesome']
```

```
# Example 2: split() with specific delimiter
date = '2024-02-15'
parts = date.split('-')
print(parts)
```

```
Output: ['2024', '02', '15']
```

```
# Example 3: split() with maxsplit
text = 'one,two,three,four'
parts = text.split(',', 2)
print(parts)
```

```
Output: ['one', 'two', 'three,four']
```

```
# Example 4: join() with space
words = ['Python', 'is', 'fun']
sentence = ' '.join(words)
print(sentence)
```

```
Output: Python is fun
```

```
# Example 5: join() with custom separator
words = ['Python', 'Java', 'C++']
text = ' | '.join(words)
print(text)
```

```
Output: Python | Java | C++
```

```
# Example 6: CSV parsing
csv_line = 'John,25,USA'
data = csv_line.split(',')
name, age, country = data
print(f'Name: {name}, Age: {age}, Country: {country}')
```

```
Output: Name: John, Age: 25, Country: USA
```

```
# Example 7: Reverse words in sentence
sentence = 'Hello World Python'
words = sentence.split()
reversed_sentence = ' '.join(reversed(words))
print(reversed_sentence)
```

```
Output: Python World Hello
```

```
# Example 8: Count words
text = 'Python programming is fun and powerful'
word_count = len(text.split())
print(f'Word count: {word_count}')
```

```
Output: Word count: 6
```

```
# Example 9: Split multiline text
text = '''Line 1
Line 2
Line 3'''
lines = text.split('\n')
for i, line in enumerate(lines, 1):
    print(f'{i}: {line}')
```

```
Output: 1: Line 1
2: Line 2
3: Line 3
```

```
# Example 10: Create path from parts
path_parts = ['home', 'user', 'documents', 'file.txt']
path = '/'.join(path_parts)
print(path)
```

```
Output: home/user/documents/file.txt
```

14. replace() Function

The replace() method returns a copy of the string with all occurrences of a substring replaced by another. Syntax: str.replace(old, new, count). If count is specified, only that many occurrences are replaced. Original string remains unchanged (immutable).

```
# Example 1: Basic replace
text = 'Hello World'
new_text = text.replace('World', 'Python')
print(new_text)
```

```
Output: Hello Python
```

```
# Example 2: Replace all occurrences
text = 'cat and cat and cat'
new_text = text.replace('cat', 'dog')
print(new_text)
```

```
Output: dog and dog and dog
```

```
# Example 3: Replace with count
text = 'one one one one'
new_text = text.replace('one', 'two', 2)
print(new_text)
```

```
Output: two two one one
```

```
# Example 4: Remove substring (replace with empty)
text = 'Python Programming Language'
new_text = text.replace(' Programming', '')
print(new_text)
```

```
Output: Python Language
```

```
# Example 5: Case-sensitive replacement
text = 'Python python PYTHON'
new_text = text.replace('Python', 'Java')
print(new_text)
```

```
Output: Java python PYTHON
```

```
# Example 6: Replace multiple spaces
text = 'Python    Programming'
new_text = text.replace('    ', ' ')
print(new_text)
```

```
Output: Python Programming
```

```
# Example 7: Remove special characters
text = 'Hello@World#2024!'
new_text = text.replace('@', '').replace('#', '').replace('!', '')
print(new_text)
```

```
Output: HelloWorld2024
```

```
# Example 8: Format phone number
phone = '1234567890'
formatted = phone.replace(phone[0:3], f'({phone[0:3]}) ')
formatted = formatted.replace(formatted[6:9], f'{formatted[6:9]}-')
print(formatted)
```

```
Output: (123) 456-7890
```

```
# Example 9: Sanitize filename
filename = 'my/file.txt'
safe_name = filename.replace('/', '_').replace('<', '').replace('>', '')
print(safe_name)
```

```
Output: my_filename.txt
```

```
# Example 10: Chain replacements
text = 'I love apples and apples are great'
new_text = text.replace('apples', 'oranges').replace('love', 'like')
print(new_text)
```

```
Output: I like oranges and oranges are great
```

15. strip() Function

Strip methods remove characters from string ends: strip() removes from both ends; lstrip() from left; rstrip() from right. Default removes whitespace. Can specify characters to remove. Useful for cleaning user input.

```
# Example 1: strip() - remove spaces
text = '   Python   '
cleaned = text.strip()
print(f'|{cleaned}|')
```

```
Output: |Python|
```

```
# Example 2: lstrip() - remove left spaces
text = '   Hello World'
cleaned = text.lstrip()
print(f'|{cleaned}|')
```

```
Output: |Hello World|
```

```
# Example 3: rstrip() - remove right spaces
text = 'Hello World   '
cleaned = text.rstrip()
print(f'|{cleaned}|')
```

```
Output: |Hello World|
```

```
# Example 4: strip() with specific characters
text = '###Python###'
cleaned = text.strip('#')
print(cleaned)
```

```
Output: Python
```

```
# Example 5: Remove multiple characters
text = '!!!Hello World!!!'
cleaned = text.strip('!')
print(cleaned)
```

```
Output: Hello World
```

```
# Example 6: Clean user input
user_input = '  john@example.com  '
email = user_input.strip()
print(f'Email: {email}')
```

```
Output: Email: john@example.com
```

```
# Example 7: Remove newlines
text = '\n\nPython\n\n'
cleaned = text.strip('\n')
print(f'|{cleaned}|')
```

```
Output: |Python|
```

```
# Example 8: Strip with multiple character types
url = 'https://example.com///'
cleaned = url.rstrip('/')
print(cleaned)
```

```
Output: https://example.com
```

```
# Example 9: Clean list of strings
names = ['  Alice  ', ' Bob ', '  Charlie  ']
cleaned = [name.strip() for name in names]
print(cleaned)
```

```
Output: ['Alice', 'Bob', 'Charlie']
```

```
# Example 10: Remove quotes from string
text = '""Hello World""'
cleaned = text.strip('"')
print(cleaned)
```

```
Output: Hello World
```