Unit 2 - Control Flow

Lesson 5 Practice

String Iteration and Character Processing

QUICK CHECK: Part 1 Review

Let's verify your understanding of character classification!

We just learned:

- Direct string iteration: `for char in word`
- Character range comparisons: `'A' <= char <= 'Z'`
- Counting different character types

Ready for 4 quick questions?

Q&A QUESTION 1: Character Range

What does this condition check?

Question Code

```
if '0' <= char <= '9':
    print("Found it!")
```

Think about it: What type of characters would make this condition True?

ANSWER 1: Character Range

Answer: This checks for digits (0-9)

Explanation

```
# This checks if character is between '0' and '9'
# Examples that would trigger:
char = '5'  # True - it's a digit
char = 'A'  # False - it's a letter
char = '!'  # False - it's punctuation
```

Characters '0' through '9' are the digit characters!

Q&A QUESTION 2: Loop Output

What will this code print?

Question Code

```
text = "Hi5"
count = 0
for char in text:
    if 'A' <= char <= 'Z':
        count += 1
print(count)
```

Think step-by-step: Go through each character in "Hi5" and count the uppercase letters.

ANSWER 2: Loop Output

Answer: The code will print 1

Step-by-step

```
text = "Hi5"
# H - uppercase letter ✓ (count = 1)
# i - lowercase letter ✗
# 5 - digit ✗
# Final count: 1
```

Only 'H' is an uppercase letter in "Hi5"

Q&A QUESTION 3: Condition Logic

How would you write a condition to check for ANY letter (upper OR lower)?

Consider these options

```
# Option 1: Using 'and'
'A' <= char <= 'Z' and 'a' <= char <= 'z'
# Option 2: Using 'or'
'A' <= char <= 'Z' or 'a' <= char <= 'z'
# Option 3: Single range
'A' <= char <= 'z'
```

Which option is correct and why?

ANSWER 3: Condition Logic

Correct Answer: Option 2 `'A' <= char <= 'Z' or 'a' <= char <= 'z'`

Why the others are wrong

```
# Option 1: Uses 'and' - impossible! Can't be both upper AND lower
# Option 3: Includes symbols between Z and a (like [ \ ] ^ _ `)
# Option 2: Correctly uses 'or' - either upper OR lower
```

Use 'or' to check multiple conditions where only one needs to be true!

Q&A QUESTION 4: String Iteration

Compare these two approaches for going through a string:

Method A vs Method B

```
# Method A
for char in text:
    print(char)
# Method B
for i in range(len(text)):
    print(text[i])
```

What are the advantages of Method A over Method B?

ANSWER 4: String Iteration

Answer: Method A has multiple advantages:

- **More readable** - directly states what you want (each character)
- **Prevents index errors** - no risk of wrong index calculations
- **Cleaner code** - fewer moving parts, less to go wrong

Comparison

```
# Method A - clean and safe
for char in text:
    print(char)
# Method B - more complex, risk of errors
for i in range(len(text)):
    print(text[i])  # Could fail if index wrong
```

Use direct iteration when you only need the characters, not positions!

PART 2: QUICK PRACTICE

What will this code output?

Practice Code

```
text = "Hi123"
for char in text:
    if 'A' <= char <= 'Z':
        print(f"{char} is UPPER")
    elif 'a' <= char <= 'z':
        print(f"{char} is lower")
    elif '0' <= char <= '9':
        print(f"{char} is digit")
```

Trace through: H-i-1-2-3

ANSWER: Character Classification

Output:

```
H is UPPER
i is lower
1 is digit
2 is digit
3 is digit
```

Each character gets classified into its proper category!

PART 3: YOUR TURN

Vowel Counter

Create a program that:

- Goes through each character in a string
- Counts vowels (a, e, i, o, u) - both upper and lowercase
- Test with: "Hello World"

2 minutes - GO!

Vowel Counter: Method 1

Check Each Vowel Individually

```
text = "Hello World"
vowel_count = 0

for char in text:
    if char == 'a' or char == 'e' or char == 'i' or char == 'o' or char == 'u':
        vowel_count += 1
    elif char == 'A' or char == 'E' or char == 'I' or char == 'O' or char == 'U
        vowel_count += 1

print(f"Vowels in '{text}': {vowel_count}")
```

Vowel Counter: Method 2

Using String Membership

```
text = "Hello World"
vowel_count = 0
vowels = "aeiouAEIOU"

for char in text:
    if char in vowels:
        vowel_count += 1

print(f"Vowels in '{text}': {vowel_count}")
```

Output: Vowels in 'Hello World': 3 (e, o, o)

Using String Slicing: Get First Characters

Check Word Beginnings

```
word1 = "Python"
word2 = "programming"
first_char1 = word1[0]    # P
first_char2 = word2[0]    # p

if 'A' <= first_char1 <= 'Z':
    print(f"'{word1}' starts with uppercase")

if 'a' <= first_char2 <= 'z':
    print(f"'{word2}' starts with lowercase")
```

Using String Slicing: Get Last Characters

Check Word Endings

```
sentence = "Hello World"
last_char = sentence[-1]  # d

if 'a' <= last_char <= 'z':
    print("Sentence ends with lowercase letter")
elif 'A' <= last_char <= 'Z':
    print("Sentence ends with uppercase letter")
```

PART 4: PATTERN CHALLENGE A

What does this code do?

Challenge Code

```
text = "ABC123xyz"
for i in range(len(text)):
    if '0' <= text[i] <= '9':
        print(f"Digit at position {i}: {text[i]}")
```

ANSWER: Challenge A

Finds digits and shows their positions:

```
Digit at position 3: 1
Digit at position 4: 2
Digit at position 5: 3
```

Combines indexing with character classification!

PART 4: PATTERN CHALLENGE B

What does this code do?

Challenge Code

```python
word = "Hello"
for i in range(len(word)):
    print(f"{word[i]} at index {i} and {word[-1-i]} at index {-1-i}")
```

ANSWER: Challenge B

Shows positive and negative indexing:

```
H at index 0 and o at index -1
e at index 1 and l at index -2
l at index 2 and l at index -3
l at index 3 and e at index -4
o at index 4 and H at index -5
```

Mirror indexing from both ends!

PART 5: CODE ALONG!

Text Statistics Calculator

Let's build together:

1. Count total characters, letters, and digits
2. Find the first and last letter in a string

PART 6: YOUR TURN

Password Strength Checker

Build a basic password analyzer: - Count uppercase, lowercase, and digits - Check if password has at least one of each type - Check if password is at least 8 characters long - Print "Strong" or "Weak"
3 minutes - GO!

Password Strength: Setup

Initialize Variables

```
password = input("Enter a password: ")

uppercase_count = 0
lowercase_count = 0
digit_count = 0
```

Password Strength: Count Characters

Analyze Each Character

```
for char in password:
    if 'A' <= char <= 'Z':
        uppercase_count += 1
    elif 'a' <= char <= 'z':
        lowercase_count += 1
    elif '0' <= char <= '9':
        digit_count += 1
```

Password Strength: Check Requirements

Test Strength Criteria

```
length_ok = len(password) >= 8
has_uppercase = uppercase_count > 0
has_lowercase = lowercase_count > 0
has_digit = digit_count > 0
```

Password Strength: Final Decision

Determine Strength

```
if length_ok and has_uppercase and has_lowercase and has_digit:
    print("Password strength: STRONG")
else:
    print("Password strength: WEAK")
```

Finding Digit Positions

Advanced Text Analysis

```
text = "Python123Programming"

print("Digits found:")
for i in range(len(text)):
    if '0' <= text[i] <= '9':
        print(f"  '{text[i]}' at position {i}")
```

Counting Letter Groups

Track Consecutive Letters

```
text = "abc123def456"
letter_groups = 0
in_letter_group = False

for char in text:
    if 'A' <= char <= 'Z' or 'a' <= char <= 'z':
        if not in_letter_group:
            letter_groups += 1
            in_letter_group = True
    else:
        in_letter_group = False
print(f"Letter groups: {letter_groups}")
```

PART 7: DEBUG CHALLENGE

Find the error in this vowel counter:

Buggy Code

```
text = "Hello World"
vowels = 0

for char in text:
    if char == "aeiou":
        vowels += 1
print(f"Vowels: {vowels}")
```

What's wrong?

DEBUG SOLUTION

Problem: `char == "aeiou"` compares single character to entire string!

Correct Version

```
text = "Hello World"
vowels = 0

for char in text:
    if char in "aeiou":
        vowels += 1

print(f"Vowels: {vowels}")
```

Use 'in' to check membership, not == for entire strings!

PART 8: YOUR TURN

Word Analysis Challenge

Create a comprehensive analyzer: - Count total characters and letters only - Find longest sequence of consecutive letters - Check if sentence starts and ends with letters

Use indexing, slicing, and character iteration!

Word Analyzer: Setup

Initialize Variables

```
sentence = input("Enter a sentence: ")

total_chars = len(sentence)
letter_count = 0
current_word_length = 0
longest_word_length = 0
```

Word Analyzer: Process Characters

Analyze Each Character

```
for char in sentence:
    if 'A' <= char <= 'Z' or 'a' <= char <= 'z':
        letter_count += 1
        current_word_length += 1
        if current_word_length > longest_word_length:
            longest_word_length = current_word_length
    else:
        current_word_length = 0
```

Word Analyzer: Check Ends

Test First and Last Characters

```
starts_with_letter = False
ends_with_letter = False

if total_chars > 0:
    first_char = sentence[0]
    last_char = sentence[-1]

    if 'A' <= first_char <= 'Z' or 'a' <= first_char <= 'z':
        starts_with_letter = True

    if 'A' <= last_char <= 'Z' or 'a' <= last_char <= 'z':
        ends_with_letter = True
```

Word Analyzer: Display Results

Show Analysis Results

```
print(f"Total characters: {total_chars}")
print(f"Letters only: {letter_count}")
print(f"Longest word: {longest_word_length}")
print(f"Starts with letter: {starts_with_letter}")
print(f"Ends with letter: {ends_with_letter}")
```

What We Accomplished Today

- **Reviewed string indexing and slicing** with positive and negative indices
- **Mastered string iteration** using `for char in string` syntax
- **Used character range comparisons** to classify letters and digits
- **Built practical text analyzers** using only basic Python constructs
- **Combined multiple techniques** for comprehensive text processing

You can now analyze any text data using fundamental Python skills!

Next up: While loops and loop control - for more flexible iteration!

Practice Challenges

Try these at home:

1. **Letter Frequency:** Count each vowel (a, e, i, o, u) separately
2. **Text Reverser:** Print string backwards using negative indexing
3. **Mixed Case Detector:** Find words with both upper and lowercase

4. **Digit Extractor:** Collect all numbers from mixed text

Master character-by-character processing for powerful text analysis!

Questions? {#questions }

Key Takeaways:

- Use `word[0]` and `word[-1]` for first and last characters
- `for char in string` is cleaner than index-based loops
- Character ranges like `'A' <= char <= 'Z'` work by ASCII values
- String slicing `text[start:end]` helps analyze specific parts
- Combine techniques for comprehensive text processing

You're ready to tackle any text processing challenge!