Week 3 Asynchronous Materials Guide

String Operations and Debugging

This Week's Topics Overview

Building on Week 2's variables, you'll now learn to:

- Use important string methods: upper(), lower(), strip(), replace()
- Access parts of strings using indexing [0] and slicing [start:end]
- Format strings with f-strings for dynamic output
- Read and understand error messages
- Debug common string-related errors

Start Here: Video Tutorial

Corey Schafer Python Strings Tutorial

Python Tutorial for Beginners 2: Strings Working with Textual Data

- **Duration:** About 30 minutes
- **What you will learn:** String basics, methods, formatting, and slicing
- **Why this helps:** Shows examples with real code
- **Study approach:** Pause and code along

String Methods and Operations

W3Schools Modify Strings

https://www.w3schools.com/python/python_strings_modify.asp

- Important string methods: upper(), lower(), strip(), replace()
- Examples you can test right away

LearnPython.org Basic String Operations

https://www.learnpython.org/en/Basic_String_Operations

- Tutorial with exercises
- Covers indexing, slicing, and string methods
- Complete the exercise at the end

String Indexing and Slicing

Visual Guide: String Indexing

```
       H  e  l  l  o

    0 1 2 3 4 (positive indexes)
 -5 -4 -3 -2 -1 (negative indexes)
```

Key Point: Python starts counting at 0, not 1. Use negative numbers to count from the end.

W3Schools Slicing Strings

https://www.w3schools.com/python/python_strings_slicing.asp

- Learn [start:end] syntax with examples
- Shows different slice combinations

String Indexing and Slicing Examples

```python
# String indexing getting single characters
word = "Python"
print(word[0])    # P (first character)
print(word[1])    # y (second character)
print(word[5])    # n (last character at index 5)
print(word[2])    # t (character at index 2)

# Check length first to avoid errors
print(f"Length: {len(word)}")  # Length: 6
print(f"Last index: {len(word) - 1}")  # Last index: 5

# String slicing getting multiple characters
```

```python
sentence = "Hello World"
print(sentence[0:5])    # "Hello" (characters 0 through 4)
print(sentence[6:11])   # "World" (characters 6 through 10)
print(sentence[6:])     # "World" (from 6 to end)
print(sentence[:5])     # "Hello" (from start to 4)

# Practical slicing examples
email = "student@college.edu"
username = email[:email.index("@")]  # "student"
domain = email[email.index("@") + 1:]  # "college.edu"
print(f"Username: {username}, Domain: {domain}")

# Getting initials from a name
full_name = "Alice Johnson"
space_position = full_name.index(" ")
first_initial = full_name[0]
last_initial = full_name[space_position + 1]
print(f"Initials: {first_initial}.{last_initial}.")  # "A.J."
```

Slicing Visualization

```
"Hello World"

  H e l l o W o r l d
0 1 2 3 4 5 6 7 8 9 10
```

sentence[0:5] gives "Hello" (positions 0,1,2,3,4)

sentence[6:] gives "World" (position 6 to end)

sentence[:5] gives "Hello" (start to position 4)

F-String Formatting

W3Schools String Formatting

https://www.w3schools.com/python/python_string_formatting.asp

- Basic f-string syntax
- Shows how to put variables inside strings

Real Python F-Strings Guide

https://realpython.com/python-f-strings/

- More f-string examples
- Read when you want more details

F-String Examples

```python
# Basic f-string formatting
name = "Sarah"
age = 22
city = "Boston"

# Simple variable insertion
greeting = f"Hello, my name is {name}"
print(greeting)  # "Hello, my name is Sarah"

# Multiple variables in one string
info = f"I am {name}, I am {age} years old, and I live in {city}"
print(info)

# Using expressions inside f-strings
next_year = f"Next year I will be {age + 1} years old"
print(next_year)  # "Next year I will be 23 years old"

# Combining f-strings with string methods
message = f"Welcome {name.upper()}!"
print(message)  # "Welcome SARAH!"

# Using f-strings with user input
user_name = input("What is your name? ").strip()
user_age = int(input("How old are you? "))
result = f"Nice to meet you, {user_name}! You are {user_age} years old."
print(result)

# F-strings with calculations
price = 19.99
tax_rate = 0.08
total = f"Price: ${price}, Tax: ${price * tax_rate:.2f}, Total: ${price * (1 + tax_rate):.2f}"
print(total)
```

F-String Format Examples

```
name = "Alice"

age = 25

f"My name is {name}" gives "My name is Alice"
f"I am {age} years old" gives "I am 25 years old"
f"{name} is {age}" gives "Alice is 25"
f"Next year: {age + 1}" gives "Next year: 26"
```

Understanding Error Messages -- Do Not Skip This Section

When you start working with strings, you will see error messages. This is normal and happens to every programmer. The key is learning to read these messages instead of giving up.

Error Type 1: String Index Out of Range

This error happens when you try to access a character that does not exist in the string.

```
IndexError Traceback (most recent call last) Cell In[3], line 3 1 name = "Alice" 2
print(len(name)) # This shows 5 ----> 3 print(name[10]) # Error! Alice only has 5
characters (indexes 0-4) IndexError: string index out of range
```

How to read this error:

- **Cell In[3], line 3:** The error happened in cell 3 on line 3 of your code
- **IndexError:** You tried to access an index that does not exist
- **Fix:** Check the string length with len() before accessing indexes

Error Type 2: TypeError with String Operations

This happens when you try to do math with strings and numbers without converting first.

```
TypeError Traceback (most recent call last) Cell In[5], line 3 1 age = input("Enter
your age: ") # input() always gives you a string! 2 print(f"Age type: {type(age)}")
# This will show <class 'str'> ----> 3 result = age + 10 # Error! Can't add string
+ number TypeError: can only concatenate str (not "int") to str
```

How to read this error:

- **TypeError:** You mixed up data types
- **"can only concatenate str (not int) to str":** Python is telling you exactly what went wrong
- **Fix:** Convert the string to a number first: age = int(input("Enter your age: "))

Error Type 3: Attribute Error with String Methods

This happens when you try to use a method that does not exist or call it incorrectly.

```
AttributeError Traceback (most recent call last) Cell In[2], line 2 1 name =
"alice" ----> 2 print(name.Upper()) # Error! Python is case-sensitive
AttributeError: 'str' object has no attribute 'Upper'
```

How to read this error:

- **AttributeError:** The method name is wrong
- **'str' object has no attribute 'Upper':** There is no method called 'Upper'
- **Fix:** Use the correct method name: name.upper() (lowercase)

Error Type 4: NameError

This happens when you make a typo in a variable name or forget to define it.

```
NameError Traceback (most recent call last) Cell In[4], line 3 1 name = "Alice" 2
print("Setting up variables...") ----> 3 print(f"Hello {nme}") # Error! Typo in
variable name NameError: name 'nme' is not defined
```

How to read this error:

- **NameError:** Variable name is wrong or not defined
- **name 'nme' is not defined:** Python cannot find a variable called 'nme'
- **Fix:** Check spelling: change 'nme' to 'name'

General Debugging Tips:

- **Read the last line first:** This tells you what type of error happened

- **Look at the line number:** This tells you where to look in your code
- **Check for typos:** Most errors are simple spelling mistakes
- **Use print() to check values:** Print variables to see what they contain
- **Check data types:** Use type() to see if variables are strings, numbers, etc.

Quick Reference

Important String Methods

```python
# Basic string methods
text = "  Hello World  "
print(text.upper())        # "  HELLO WORLD  "
print(text.lower())        # "  hello world  "
print(text.strip())        # "Hello World" (removes spaces)
print(text.replace("Hello", "Hi"))  # "  Hi World  "

# Method chaining
clean_text = text.strip().upper()  # "HELLO WORLD"

# Checking string content
message = "Python Programming"
print(message.startswith("Python"))  # True
print(message.endswith("ing"))       # True
print(message.count("o"))            # 2
print(message.index("Pro"))          # 7
```

Combining All Concepts

```python
# Real example: Processing user input
full_name = input("Enter your full name: ").strip().title()
age = int(input("Enter your age: "))

# Extract first name (up to first space)
space_pos = full_name.index(" ")
first_name = full_name[:space_pos]

# Create a personalized message
message = f"Hello {first_name}! You are {age} years old."
```

```python
print(message.upper())

# Get nickname using slicing
nickname = first_name[:3]
print(f"Your nickname could be {nickname}")
```

Debugging Your Code

```python
# Always check your variables
name = input("Enter name: ")
print(f"Name: '{name}' (length: {len(name)}, type: {type(name)})")

# Check string length before slicing
print(f"First 3 letters: {name[:3]}")

# Convert input before math to avoid TypeError
age_text = input("Enter age: ")
print(f"As text: {age_text} (type: {type(age_text)})")
age_number = int(age_text)
print(f"As number: {age_number} (type: {type(age_number)})")

# Use print() to debug step by step
text = "hello world"
print(f"Original: {text}")
print(f"Uppercase: {text.upper()}")
print(f"After strip: '{text.strip()}'")
print(f"Length: {len(text)}")
```