Python Week Recap

Everything We've Learned So Far

Part 1: The Basics

Variables, Input/Output, and Basic Operations

What We've Built Up To

In the past few days you've both learned and built a ton, **8 assignments in total!**

By the end of this week, you'll be able to build:

- Smart calculators
- Decision-making programs
- Programs that repeat tasks
- Reusable code with functions
- Interactive games

Let's review how we got here!

Variables: Storing Information

Variables are containers for data:

```
# Different types of data
name = "Alice"          # String
age = 25                # Integer
temperature = 98.6      # Float
is_student = True       # Boolean

# Variables can change
score = 0
score = score + 10
print(f"Score: {score}")
```

```
Score: 10
```

Every variable is made of three things:

- **Name**: How you refer to it (e.g., `name`, `age`)
- **Value**: The data it holds (e.g., `"Alice"`, `25`)
- **Type**: The kind of data (e.g., `str`, `int`)

Input and Output: Talking to Users

```
# Output: Display information
print("Welcome to Python!")

# Input: Get information
name = input("What's your name? ")
age = int(input("What's your age? "))

# Combine them
print(f"Hello {name}, you are {age} years old!")
```

Things to remember:

- `print` displays text to the user
- `input` gets user input (always returns a string)
  - When you need a number, convert it with `int()` or `float()`

Basic Math Operations

Python is a powerful calculator:

```python
# Basic operations
total = 10 + 5        # Addition
difference = 10 - 5   # Subtraction
product = 10 * 5      # Multiplication
quotient = 10 / 5     # Division (always float!)

# Special operations
floor = 10 // 3       # Floor division: 3
remainder = 10 % 3    # Modulus: 1
power = 2 ** 3        # Exponentiation: 8

print(f"10 // 3 = {floor}, 10 % 3 = {remainder}")
```

```
10 // 3 = 3, 10 % 3 = 1
```

Part 2: Making Decisions

Conditionals and Logic

If Statements: Making Choices

Programs need to make decisions based on conditions:

```
temperature = 85

if temperature > 90:
    print("It's very hot!")
elif temperature > 70:
    print("Nice weather!")
elif temperature > 50:
    print("A bit cool")
else:
    print("It's cold!")
```

```
Nice weather!
```

Things to remember:

- Indentation matters! Code inside `if` blocks must be indented all the same
- Order matters: Python checks conditions in order

Comparison Operators

How we check conditions:

```
age = 18

# All the ways to compare
print(f"age == 18: {age == 18}")  # Equal to
print(f"age != 21: {age != 21}")  # Not equal
print(f"age > 16: {age > 16}")    # Greater than
print(f"age >= 18: {age >= 18}")  # Greater or equal

# Common mistake
x = 5    # Assignment (one =)
x == 5   # Comparison (two ==)
```

```
age == 18: True
age != 21: True
age > 16: True
age >= 18: True
```

Combining Conditions

Use `and`, `or`, `not` for complex decisions:

```
age = 16
has_permit = True
is_weekend = False

# Both must be true
can_drive = age >= 16 and has_permit
print(f"Can drive: {can_drive}")

# At least one must be true
can_sleep_in = is_weekend or is_holiday
print(f"Can sleep in: {can_sleep_in}")

# Flip the boolean
must_wake_early = not can_sleep_in
```

```
Can drive: True
PythonError: Traceback (most recent call last):
File "/lib/python312.zip/_pyodide/_base.py", line 597, in eval_code_async
await CodeRunner(
File "/lib/python312.zip/_pyodide/_base.py", line 411, in run_async
coroutine = eval(self.code, globals, locals)
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Part 3: Repeating Code

Loops

While Loops: Repeat While True

Keep going as long as a condition is true:

```
# Count to 5
count = 1
while count <= 5:
    print(f"Count: {count}")
    count = count + 1
```

```
Count: 1
Count: 2
Count: 3
Count: 4
Count: 5
```

This is especially useful for getting an input until it's valid:

```
while True:
    answer = input("Continue? (yes/no): ")
    if answer == 'yes' or answer == 'no':
        break
    print("Please enter yes or no")
```

For Loops: Repeat a Specific Number of Times

When you know how many times to repeat:

```
# Count from 0 to 4
for i in range(5):
    print(f"i = {i}")

# Count from 1 to 5
for num in range(1, 6):
    print(f"Number: {num}")

# Calculate running total
total = 0
for price in range(7):
    total = total + price
print(f"Total: ${total}")
```

```
i = 0
i = 1
i = 2
i = 3
i = 4
Number: 1
Number: 2
```

When to Use Which Loop?

Use `while` when:

- You don't know how many times to repeat
- Waiting for user input
- Checking a condition that might change

```
# While: unknown repetitions
while input("Again? ") == "yes":
    print("Going again!")
```

Use `for` when:

- You know exactly how many times
- Counting up or down
- Processing a sequence (we'll get to this soon)

```
for day in range(7):
    print(f"Day {day + 1}")
```

```
Day 1
Day 2
Day 3
Day 4
Day 5
Day 6
Day 7
```

Part 4: Functions

Writing Reusable Code

Functions: Package Your Code

Instead of repeating code, write it once:

```python
# Define a function
def greet(name):
    print(f"Hello, {name}!")
    print("Welcome to Python!")

# Use it multiple times
greet("Alice")
greet("Bob")

# Functions can return values
def add(a, b):
    return a + b

result = add(5, 3)
print(f"5 + 3 = {result}")
```

```
Hello, Alice!
Welcome to Python!
Hello, Bob!
Welcome to Python!
5 + 3 = 8
```

Function Patterns We've Learned

```
# Get input with validation
def get_positive_number():
    while True:
        num = float(input("Enter positive number: "))
        if num > 0:
            return num
        print("Must be positive!")

# Check conditions
def is_passing_grade(score):
    return score >= 60

# Process and return
def calculate_tip(bill, percent=20):
    return bill * (percent / 100)
```

Part 5: Putting It All Together

Building Complete Programs

Program Structure

```
def get_grade():
    """Get a valid grade from user"""
    while True:
        grade = float(input("Enter grade (0-100): "))
        if 0 <= grade <= 100:
            return grade
        print("Invalid grade!")

def calculate_letter(score):
    """Convert number to letter grade"""
    if score >= 90:
        return 'A'
    elif score >= 80:
        return 'B'
    elif score >= 70:
        return 'C'
    else:
        return 'F'

# Main program
grade = get_grade()
letter = calculate_letter(grade)
print(f"Your grade: {letter}")
```

Common Patterns: Menu Systems

```
def show_menu():
    print("\n=== CALCULATOR ===")
    print("1. Add")
    print("2. Subtract")
    print("3. Quit")

def calculator():
    while True:
        show_menu()
        choice = input("Choose: ")
        if choice == '3':
            print("Goodbye!")
            break
        elif choice == '1' or choice == '2':
            a = float(input("First number: "))
            b = float(input("Second number: "))
            if choice == '1':
                print(f"Result: {a + b}")
            else:
                print(f"Result: {a - b}")

calculator()
```

Common Patterns: Game Loop

```
import random

def guessing_game():
    secret = random.randint(1, 100)
    attempts = 0

    print("I'm thinking of a number 1-100!")

    while True:
        guess = int(input("Your guess: "))
        attempts = attempts + 1

        if guess == secret:
            print(f"Correct! Took {attempts} tries!")
            break
        elif guess < secret:
            print("Too low!")
        else:
            print("Too high!")

guessing_game()
```

Common Issues to Avoid

```
# 1. Assignment vs comparison
if x = 5:    # ❌ SyntaxError
if x == 5:   # ✅ Correct

# 2. Infinite loops
while True:  # ❌ No way out!
    print("Help!")

# 3. Wrong types
age = input("Age: ")  # Returns string!
if age > 18:          # ❌ Can't compare string to int
if int(age) > 18:     # ✅ Convert first

# 4. Indentation matters
if True:
print("Hello")      # ❌ IndentationError
    print("Hello")  # ✅ Correct
```

Things You May Have Picked Up On

I didn’t teach you everything…

External Functions and Libraries

You can use functions from other files or libraries:

```
from random import randint

def roll_dice():
    return randint(1, 6)

print(f"You rolled a {roll_dice()}")
```

```
You rolled a 3
```

```
from math import sqrt

print(sqrt(16))  # 4.0
```

```
4.0
```

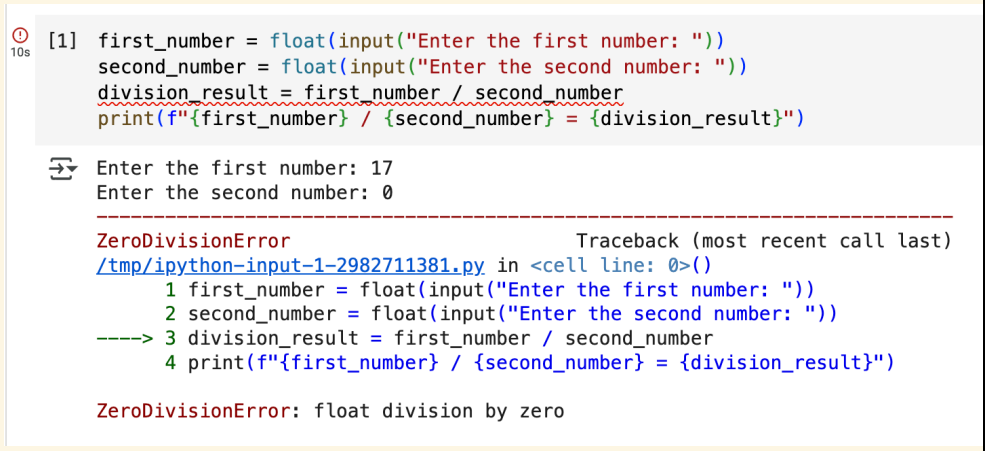
These are basic tools for now but libraries are how we borrow more advanced functionality from others. **We're going to be using A LOT of libraries in the next two weeks!**

Errors

Often we run into errors during the runtime. We call these "Exceptions".

An exception is made of three parts:

- **Type**: What kind of error it is (e.g., `ZeroDivisionError`, `ValueError`)
- **Message**: What went wrong (e.g., "division by zero")
- **Traceback**: Where it happened in the code (file name, line number



Handling Errors

It's actually pretty easy to handle errors in Python, we use a special block called `try` / `except`:

```python
def divide(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        print("Cannot divide by zero!")
        return None

a = 10
b = 0
result = divide(a, b)
print(f"{a} / {b} = {result}")
```

```
Cannot divide by zero!
10 / 0 = None
```

This will come up a little more as we introduce new libraries.

Debugging Tips

When your code doesn't work:

1. **Read error messages** - Python tells you what's wrong!
2. **Check indentation** - Python is picky about spacing
3. **Print variables** - See what's actually happening
4. **Divide the haystack!** - Don't write everything at once
5. **Rubber duck debugging** - Explain your code out loud

```
# Debug with print
print(f"DEBUG: has_key = {has_key}")
print(f"DEBUG: current_room = {current_room}")
```

Any sufficiently advanced technology is indistinguishable from magic.

~ Arthur C. Clarke