Discrete Structures!

CMPSC 102

ALLEGHENY COLLEGE

Key Questions

- How do I use **iteration** and **conditional logic** in a Python program to perform computational tasks like processing a file's contents and mathematical tasks like using Newton's method to approximate the square root of a number?

Learning Objectives

- To remember and understand some discrete mathematics and Python programming concepts, setting the stage for exploring discrete structures.

Python Programming Retrospective

- Python code is designed to be **intuitive**
- Key components of Python programming include:
    - Function calls
    - Assignment statements
    - Iteration constructs
    - Conditional logic
    - Variable creation
    - Variable computations
    - Variable output

Python Programming Retrospective

- Investigate the **syntax** and **semantics** of these components

```
x = 5
print("x + 3")
```

- Understand how to **connect** these components together in a program

```
def square(n):
return n * n

num = int(input("Enter a number: "))
print("The square is:", square(num))
```

- **Implement** Python functions to **understand** mathematical functions

A program is a sequence of statements

```
# Iteration-based program in Python

# Using a for loop to iterate over a range of numbers for num in range(1, 6):
for num in range(1, 6):
        square = num ** 2
        print(f"The square of {num} is: {square}")
```

A program is a sequence of statements

Programming parallels cooking …

- A Python program is a sequence of statements about mixing things with the rest of the ingredients … like a *recipe*
- There is a list of ingredients
- There is a sequence of events about when to use each ingredient
- Timing (run time) is important
- (Chef, waiter, guests) == (programmers, instructions, users)

Simple and compound statements – main()

Determine whether an integer is prime

(Primes can only be divided by one or themselves.)

```
def main()-> None:
        """ driver function of the program """
        # Example usage
        user_input = int(input("\t Enter a number: "))
        result = is_prime(user_input)

        if result:
                print(f"\t {user_input}: Prime number")
        else:
                print(f"\t {user_input}: Not a prime number")
# end of main()

# place at bottom of the file
main() # call the main() function
```

Simple and compound statements – is_prime()

```
def is_prime(number: int) -> bool:
        """ Determine primality: return 0 and 1"""
        # Handle special cases for 0 and 1
        if number < 2:
                return False

        # Iterate from 2 to the square root of the number
        for i in range(2, int(number**0.5) + 1):
                # If the number is divisible
                # by any value in the range, it's not prime
                if number % i == 0:
                        return False

        # If no divisors are found, the number is prime
        return True
# end of is_prime()
```

Simple and compound statements

Run the program: python3 isPrime.py or, python isPrime.py

Output from program:

```
Enter a number: 101
101: Prime number
```

- Programs contain both **simple** and **compound** statements (i.e., steps having multiple processes on one line)
- Which of these statements were simple?
- Which of these statements were compound?

Industry Standard Python – Part 1

Be like the Python professionals

- Always use Python 3 for all of your programs!
- Python2 is no longer supported…
- Add **docstrings** to your Python programs (i.e., informed comments to help others follow reasoning behind the code, including)
  - Modules
  - Classes
  - Functions
- Add comments to enhance understanding of important lines of code

Industry Standard Python – Part 2

Be like the Python professionals

- Add comments for important blocks of your program
- Use descriptive variable and function names
- The book does not always adhere to industry standards!
- All course projects will enforce these standards in GitHub Actions

Program to calculate area of a square - Function squareArea()

```
def squareArea(s: float ) -> float:
        """ determine area of square"""
        return s*s # area of square is s*s
# end of squareArea()
print(squareArea("5"))
```

What inputs *s* are acceptable?

- integers?
- floats?
- booleans?
- strings?
- imaginary numbers? $(1 + 3j)$

File: squareArea.py - Function main()

```
def main() -> None:
        sideLength = 5
        # Testing value
        print(f"Length {sideLength}")
        print(f" Area: {squareArea(sideLength)}")

        # These inputs work
        testValues_list = [2, 0, -3, 2 + 5j]

        # why will these inputs not work?
        # testValues_list = [True, "radius"]

        print(f"\n Iterating over the list.")
        for val in testValues_list: # iteration
        print(f" Length {val}, Area: {squareArea(val)}")
        # end main()

main() # call the driver function
```

File: squareArea.py - Initiation code

Run the program: python3 isPrime.py or, python isPrime.py

Output from program:
Length 5
Area: 25

Iterating over the list.
Length 2, Area: 4
Length 0, Area: 0
Length -3, Area: 9
Length (2+5j), Area: (-21+20j)