Discrete Structures!

CMPSC 102

ALLEGHENY COLLEGE

Midterm Exam Policy

Date & Time:
📅 February 19, 2025
🕜 1:30 PM – 4:20 PM

📖 **Exam Details:**
✅ 9 Questions
🚫 No Local or Online Resources Allowed
⏳ Late Submission is NOT Acceptable

📎 **Midterm Invitation Link:**
📢 Will be sent through Discord

Midterm Exam Policy

- Add-Your-Name
- Re-type the sentence "I adhered to the Allegheny College Honor Code while completing this examination."
- Honor Code

CSV Data - Files in Directories Can Store n-Tuples

- Suppose you had some data in a CSV format?
- How to do something with the data?!

- CSV data: sandbox/contacts.csv

```
tylernelson@gmail.com,Careers adviser
gregory02@medina-mayer.com,"Accountant, management"
jonesmiguel@hotmail.com,Health and safety inspector
rsanchez@yahoo.com,"Surveyor, planning and development"
hillfrank@ward-wood.com,"Scientist, physiological"
aaronhunter@gmail.com,"Surveyor, insurance"
kylebarnes@hotmail.com,Records manager
joe70@yahoo.com,Network engineer
torresjames@white.info,Electrical engineer
shawkins@watson.com,Science writer
```

Functions that Manage Tuples

File: csvreader.py

```
from os.path import exists
from logging import exception

def openCSVFile(fname_str: str) -> str:
        """loads a file, returns csv string"""
        # print("openCSVFile()")
        if not exists(fname_str): # no file found?
                print(f"\t [-] No file by that name: {fname_str}")
                exit() # end program if no file has been found.
        try:
                data_str = open(fname_str, "r").read()
        except exception:
                print("\t [-] Using correct filename?")
                return None
        # commas in this loaded file?
        if len(data_str) > 0 and "," in data_str:
                return data_str
        return None
```

Functions that Manage Tuples - iterateData

```
def iterateData(in_str: str) -> dict:
        """Function to output the data in tidy lines. Place data into dictionary."""
        contact_dict = {}
        for line in in_str.splitlines():
                # print(line)
                # get the name, located before first comma
                name_str = line[: line.find(",")]
                service_str = line[line.find(",") + 1 :].replace("'", "")
                contact_dict[name_str] = service_str
        return contact_dict
```

Functions that Manage Tuples - main()

```
def main() -> None:
        """driver function"""
        prompt_str = "\t Enter the CSV filename : "
        myFile_str = input(prompt_str)
        # print(f"\t [+] You entered file : {myFile_str}")

        myCSV_str = openCSVFile(myFile_str)
        # print(f"Main() {myCSV_str}")

        # print out in tidy lines
        myContact_dict = iterateData(myCSV_str)

        # print(f"Dictionary of names: {myContact_dict}")
        for i in myContact_dict:
                print(f"\t [+] {i} : {myContact_dict[i]}")
```

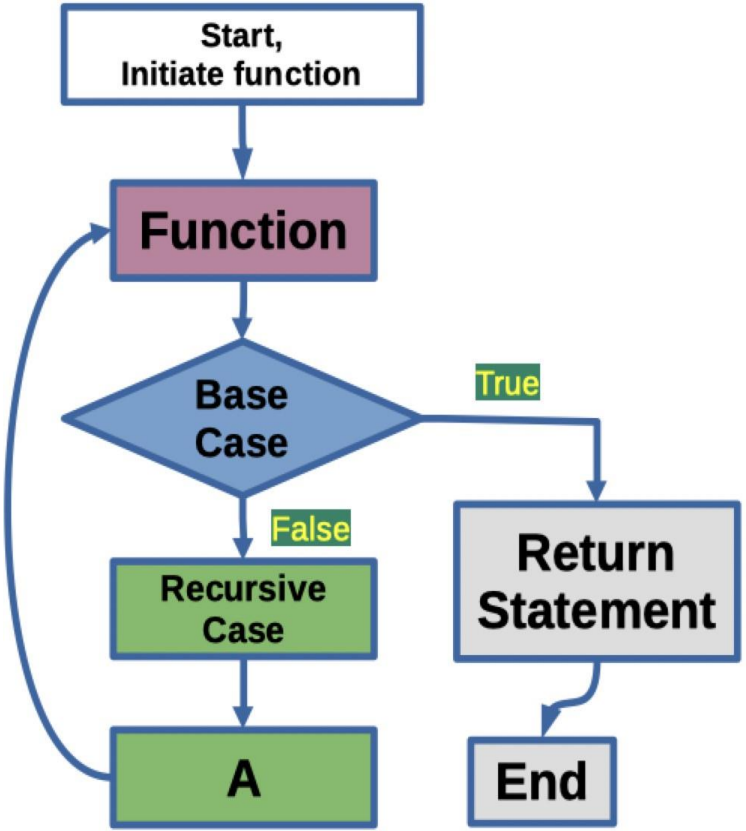
Fun Activity - Ungraded work

- Prepare Python code to implement the following code
  - Create a tuple with some data
  - Now, change the tuple into a list
  - Now, combine the list with a dictionary
  - Now, place a tuple in the dictionary
  - Print out the contents of each data structure

Virtual Environments

- How do I use virtual environments like Venv and Poetry, along with tools like Typer and other resources, to create a professional Python project?

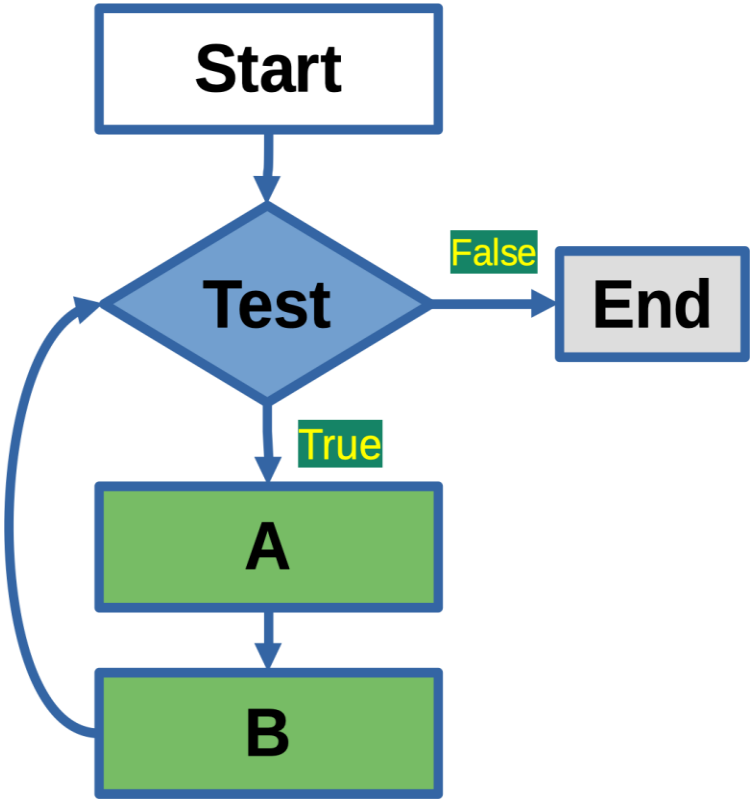
Creating Solutions



Calculating Factorials by Recursion

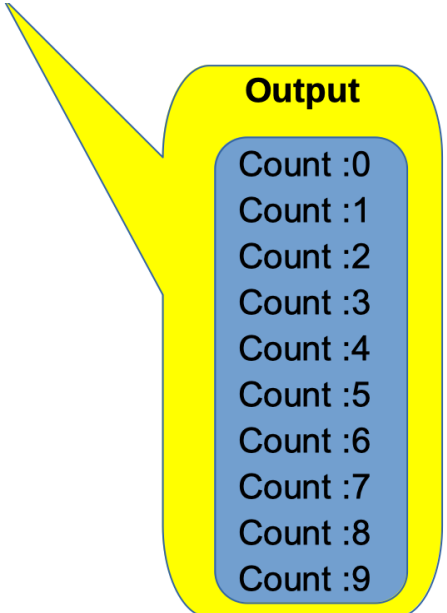
```
def factorial(number: int):
        if number == 1:
                return 1
        return number * factorial(number - 1)

num = 5
print("The factorial of " str(num)
        + " is " str(factorial(num)))
```

Loops for Iteration - A loop is a way to reuse code blocks



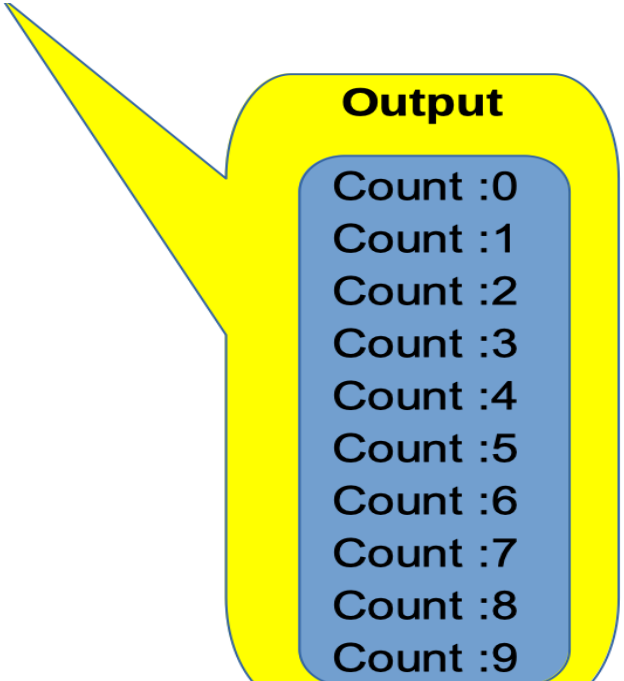
The While Loop

```
index = 0
while (index < 10): # condition
        print(f"Count :{index}")
        index += 1 # add one to index
```



The for … in range() Loop

```
for index in range(10):
        print(f"Count :{index}")
```



Square Roots - Mathematical loops to find quadruple roots

How to compute : $\sqrt{x}$?

Method

The function initializes the guess for the square root and iteratively refines it using an approximation formula until the approximation is within the specified tolerance.

The While Loop Application - Finding a Square Root

```
n = 4
guess = 1.0
while abs(n - guess*guess) > 0.0001:
        guess = guess - (guess*guess - n)/(2*guess)
        print(f"n = {n} : guess = {guess}")
```

- Iteratively **guesses** the square root until **within tolerance**
- The while loop uses ‘abs‘ for computing an absolute value
- This loop computes the root as 2.0000000929222947
- The math.sqrt(n) function confirms this approximation!
- Any questions about this way to approximate a square root?

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()
```

Data Types

```
height_int = 5
print(f" The height is: {height_int}")
print(" The height is:", height_int) # print another way

num_float = 3.14
print(f" The float variable is : {num_float}")

s_str = "Hello World"

print(" The integer is equal to: ", s_str)
```

Key Components

- **Function calls**: Granting temporary kernel-time and/or using issuing parameters to a sub-sequence of instruction in a program.

```
def greet(name):
    print(f"Hello, {name}! Welcome to Python.")

greet("Alice")
```

- **Assignment statements**: The issuing of a value to a variable or place in memory to contain the value.

```
x = 10  # First assignment (also creates x)
x = x + 5  # Reassigning a new value to x
y = x * 2  # Assigning a computed value to y
```

Key Components

- **Iteration constructs**: Structures used in computer programming to repeat the same computer code multiple times (*loops*).

```
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:
    print(fruit)
```

- **Conditional logic**: the use of logical rules in code to govern steps taken.

```
age = 18

if age >= 7:
    print("You are an adult.")
```

Key Components

- **Variable creation: The introduction of an object in memory to contain some value**

```
x = 10  # 'x' is created and assigned the value 10
name = "Alice"  # 'name' is created and assigned a string value
```

- **Variable computations**: The use of values contained in variables to create new value using an operator.

```
a = 10
b = 5
sum_result = a + b      # Addition
diff_result = a - b     # Subtraction
prod_result = a * b     # Multiplication
quotient_result = a / b  # Division (float)
mod_result = a % b      # Modulus (remainder)
exp_result = a ** b     # Exponentiation (10^5)
```

Key Components

- **Variable output:** The revealing of some value in a variable by printing or another means

```
name = "Alice"
age = 25

print(name)  # Output: Alice
print(age)   # Output: 25
```

Runnable Application: Using Python to Process and Analyze Data

```
#!/usr/bin/env python3
""" Demo program"""

myFile = [1,2,3,4,5,6,7,8,9,10]
sum = 0
count = 0
for line in myFile:
        n = int(line)
        sum += n
        count += 1
print(sum/count)
```