**Module 1: Introduction to Python (2 hours)**

#Variables do not need to be declared with any particular type, and can even change type after they have been set.

#Example:

#NOTE: This will create two variables

thisIsAnInteger = 5 # int

ThisIsAnInteger = "Five" #str

#NOTE: String variables can be declared either by using single or double quotes:

thisIsAnotherString = "Hello, World!!" # str using double string

#NOTE: Get the type of the variable using type() function

print(type(thisIsAnInteger)) // int

print(type(thisIsAString)) // str

#NOTE: Assigning multiple values in at once

x,y,z = 'Orange', 'Red', "Blue"

print(x) // Orange

print(y) // Red

print(z) // Blue

#NOTE: Assigning single value into multiple variables

X = Y = Z = "Orange"

#NOTE: The Python print() function is often used to output variables.

#In the print() function, you output multiple variables, separated by a comma:

print(X,Y,Z) ‘Orange’, ‘Orange’, ‘Orange’

**Simple code to incorporate the loop and conditional statement**

**Operators used**

% Modolus - This operator gives remainder of a division operation

== Equal – This is used to compare two different values and determine if they are equal to each other

**Other Operators that are not used in the code:**

+ Addition

- Subtraction

\* Multiplication

/ Division

// Floor division

Code

*#* *iterate 30 times, with index taking on the values from 1 to 30 (inclusive) in each iteration*

for index in range(1, 31):

    if index % 15 == 0:

        print(str(index) + ' ' + 'FizzBuzz') *#NOTE: If the number is divisible by both 3 and 5 then it will print FizzBuzz*

    elif index % 3 == 0:

        print(str(index) + ' ' + 'Fizz') *#NOTE: If the number is divisible by 3 then it will print Fizz*

    elif index % 5 == 0:

        print(str(index) + ' ' + 'Buzz') *#NOTE: If the number is divisible by 3 then it will print Buzz*

    else:

        print(index) *# Else it will only print the index*

**Module 2: Working with Functions**

Write functions for:

* 1. Mathematical operations (addition, subtraction, etc.)
  2. Unit conversion (e.g., Celsius to Fahrenheit)
  3. String manipulation (finding the length of a string, checking if a character exists)

**Code**

*#TODO: Theses are the function for getting the sum, product, difference and quotient of the two numbers*

def addition(x, y):

    sum = x + y

    return f'The sum of {x} + {y} is {sum}'

def multiplication(x, y):

    product = x \* y

    return f'The product of {x} \* {y} is {product}'

def subtraction(x, y):

    difference = x - y *#Get the absolute value of the number*

    return f'The difference of {x} - {y} is {difference}'

def division(x, y):

    quotient = x / y

    return f'The quotient of {x} / {y} is {quotient}'

*# NOTE: This function is for problem 1 Mathematical Operators*

def mathematical\_operators(operator):

    print('Mathematical Operators')

    num1 = float(input('Enter Num1: ')) *# NOTE: if you want to perform numerical operations on this input, you need to convert it to a numeric type (like int or float).*

    num2 = float(input('Enter Num2: '))

    if operator == '1':

            print(addition(num1, num2))

    elif operator == '2':

            print(subtraction(num1, num2))

    elif operator == '3':

            print(multiplication(num1, num2))

    elif operator == '4':

            print(division(num1, num2))

    else:

            print('Invalid operator')

def unit\_conversion(celsius):

    print('Unit Conversion')

    fahrenheit = (celsius \* 9/5) + 32

    return fahrenheit

def get\_percentage(part, whole):

    total\_percentage =  (float(part) / float(whole)) \* 100

    return total\_percentage

def string\_manipulation(input\_char):

    print('String Manipulation')

    string = 'Hello, World. This is a string'

    print(f'String: {string}')

    print(f'Length: {len(string)}') *#NOTE: len() is used to get the total length of the string including the spaces*

    if input\_char in string: *#NOTE: The in Keyword si used to check if a value exist in the sequence*

        return f"'{input\_char}' exists in the string: '{string}'"

    else:

        return f"'{input\_char}' does not exist in the string: '{string}'"

def module\_two\_activity(selected):

    print(selected)

    if selected == '1':

        print('Mathematical Operator')

        print('1 - Addition')

        print('2 - Subtraction')

        print('3 - Multiplication')

        print('4 - Division')

        operator = input('Enter Operator: ')

        mathematical\_operators(operator)

    elif selected == '2':

        celsius = float(input('Enter Celsius: '))

        print(f'{celsius}°C is equal to {unit\_conversion(celsius)}°F')

    elif selected == '3':

        char = input('Input a character: ')

        print(string\_manipulation(char))

    elif selected == '4':

        part = input('Part: ')

        whole = input('Whole:')

        print(f'{part} is {get\_percentage(part, whole)}% of {whole}') *#NOTE: f stands for "formatted string literals. " F-strings provide a concise and convenient way to embed expressions inside string literals, allowing you to create strings with embedded variables or expressions directly within them.*

print('Module 2 activity: Working with functions')

print('1 - Mathematical Operators')

print('2 - Unit Conversion')

print('3 - String Manipulation')

print('4 - Get Percentage')

user\_input = input('Enter your choice: ')

module\_two\_activity(user\_input)

**Module 3: Data Structures and Lists**

* Implement more advanced list methods (e.g., pop, insert, reverse).
* Explore nested lists (lists within lists) for complex data organization.

Code

list = [[3, 2, 1, 4,4], [2,3], [3,2,4,1,6]]

list[0].append(5) *#NOTE Append 5 into index 0*

print(list)

for index in range(len(list)):

    list[index].sort() *# Sort each list*

    if index == 0 and index == 2:

        list[index].remove(0) *#NOTE: Remove the current element of index 0 for list 0 and list 2*

print(list)

**Challenge: Building a Tic-Tac-Toe Game:**

* Design a simple tic-tac-toe game using lists:
  + Represent the board as a 3x3 list (nested list).
  + Implement functions to:
    - Check for valid moves (preventing out-of-bounds placements).
    - Place markers on the board (updating the list).
    - Check for winning conditions (horizontal, vertical, diagonal).
* Consider using additional functionalities like:
  + Taking turns between players.
  + Displaying the current game state on the screen.

Code

def print\_board(board):

*#print(board)*

    for row in board:

        print(" | ".join(row)) *#NOTE: Vertical Line*

        print("-" \* 9) *#NOTE Horizontal Line*

def check\_valid\_move(board, row, col):

*#print(board, row, col)*

    if 0 <= row < 3 and 0 <= col < 3: *#NOTE: Check if both rows and colmuns are within range*

        return board[row][col] == ' ' *#NOTE: checks if the specified cell on the board is empty, if not empty the move is not valid*

    return False

def place\_marker(board, row, col, player):

*#print(board, row, col, player)*

    if check\_valid\_move(board, row, col):

        board[row][col] = player *#Place the players  marker if the move is valid*

        return True

    else:

        return False

def check\_winner(board, player):

*#TODO: Check rows and columns*

    for i in range(3):

        if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j in range(3)):

            return True

*#TODO: Check diagonals*

    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3)):

        return True

    return False

def main():

    board = [[' ' for \_ in range(3)] for \_ in range(3)] *#NOTE: this defines the dimesion of the board*

    players = ['X', 'O'] *# NOTE: this will be the marker of player*

    turn = 0

    print("Welcome to Tic Tac Toe!")

    while True: *# NOTE: infinite loop, which continues until a winner is determined or the game ends in a tie*

        print\_board(board)

        print(f"Player {players[turn]}'s turn")

        print(turn)

        row = int(input("Enter row (0, 1, or 2): "))

        col = int(input("Enter column (0, 1, or 2): "))

        if place\_marker(board, row, col, players[turn]):

            if check\_winner(board, players[turn]):

                print\_board(board)

                print(f"Player {players[turn]} wins!")

                break

            elif all(board[i][j] != ' ' for i in range(3) for j in range(3)):

                print\_board(board)

                print("It's a tie!")

                break

            else:

                turn = (turn + 1) % 2 *#NOTE: set players turn*

        else:

            print("Invalid move! Try again.")

if \_\_name\_\_ == "\_\_main\_\_": *# NOTE Using if \_\_name\_\_ == "\_\_main\_\_": allows you to write code that will only be executed if the script is run directly as the main program, and not if it is imported as a module into another script.*

    main()

## ****Module 4: Working with Files (1 hour)****

1. **Reading from text files:**
   1. Resource: Python official documentation on file I/O (<https://docs.python.org/3/library/filesys.html>)
   2. Practice using the open() function with read mode ('r') to access file contents.
2. **Writing to text files:**
   1. Same resource as above
   2. Practice using the open() function with write mode ('w') to create or modify text files.
3. **Error handling with** try-except**:**
   1. Resource: W3Schools Python Try Except Tutorial (<https://www.w3schools.com/python/python_try_except.asp>)
   2. Implement try-except blocks to handle potential errors like file not found exceptions.

**Code**

def readFile(path):

    try:

        with open(path, 'r') as file:

            lines = file.readlines()  *# Use readlines() to read all lines into a list*

*# Print each line of the file*

            for line in lines:

                print(line.strip())  *# strip() removes any leading or trailing whitespace*

    except FileNotFoundError:

        print("File not found!")

    except IOError: *# NOTE: If there is any other input/output error (raises IOError), it prints "Error reading the file!".*

        print("Error reading the file!")

def writeFile(input\_text):

    global path *# NOTE: Made this variable global using the global keyword so I can also use it outside this function*

    path = 'python3-activity-WHM1.txt'

    try:

*# Open the file in write mode ('w')*

        with open(path, 'w') as file:

            file.write(f"{input\_text}\n")  *# Corrected to add input\_text without leading space*

    except IOError: *# NOTE: If there is any other input/output error (raises IOError), it prints "Error reading the file!".*

        print("Error writing to the file!")

input\_text = input('Write Here: ')

writeFile(input\_text)

readFile(path)  *# Call readFile() after writeFile()*