Practice Exercises

Control Structures - If-Else Statements, Loops

INDIVIDUAL EXERCISES (with Instructors)

❖ If Statement

Exercise 1: Check if the number is even.

→ Define the number you want to check in "Num" variable. e.g.: *Num = 6*
→ Use the modulo operator (%) to check if the number is divisible by 2. e.g.: *if Num % 2 == 0:*
→ If the remainder is 0, the number is even. Print a message indicating that. e.g.: *print('The number is even.')*
→ Final code must look something like this:

```
num = 6
if num % 2 == 0:
   print('The number is even.')
```

→ Change the number values and observe the output.

Exercise 2: Display a message if a list is empty.

→ Define the list you want to check in "my_list" variable. e.g.: *my_list = [ ]*
→ Check the length of the list e.g.: *if len(my_list) == 0:*
→ If the length is 0, the list is empty. Print a message indicating that. e.g.: *print ('list is empty')*
→ Final code must look something like this:

```
my_list = [ ]
If len(my_list) == 0:
   print('list is empty')
```

→ Change the list values and observe the output.

❖ else Statement

Exercise 1: Write a script to check if a number as odd or even.

→ Define the number you want to check in "Num" variable e.g.: *Num = 6*
→ Use the modulo operator (%) to check if the number is divisible by 2. e.g.: *if Num % 2 == 0:*
→ If the remainder is 0, the number is even. Print a message indicating that. e.g.: *print('The number is even.')*
→ If the condition in step 2 is false (i.e., the number is odd), print a message indicating that. e.g.:

```
else:
    print(' The number is odd.')
```

→ Final code must look something like this:

```
num = 6
if num % 2 == 0:
   print('The number is even.')
else:
  print('The number is odd.')
```

→ Change the number values and observe the output.

❖ elif Statement

Exercise 1: Write a script for Temp categories.

→ Define the temperature in "temp" variable. e.g.: *temp = 15.07*
→ Check if the temperature is less than or equal to 0 degrees. e.g.: *if temp <= 0:*
→ If true, it means the temperature is freezing. Print a message indicating that. e.g.: *print('Its Freezing Colddd!')*
→ Else if the temperature is less than 10 degrees, e.g.: *elif temp <= 10:*
→ If true, it means the temperature is cold. Print a message indicating that. e.g.: *print('Its Cold')*
→ You can add as many elif statements.
→ Else if the temperature is less than 20 degrees, e.g.: *elif temp <= 20:*
→ If true, it means the temperature is moderate. Print a message indicating that. e.g.: *print('Its Moderate')*
→ Else if the temperature is less than 25 degrees, e.g.: *elif temp <= 25:*
→ If true, it means the temperature is warm. Print a message indicating that. e.g.: *print('Its warm')*
→ Else none of the above conditions are met (i.e., the temperature is 35 degrees or higher), it means that the temperature is hot. Print a message indicating that. e.g.:

```
else:
   print('Its Hotttt!')
```

→ Final code must look something like this:

```
temp = 15.07
if temp <= 0:
   print('Its Freezing Colddd!')
elif temp <= 10:
   print('Its Cold')
elif temp <= 20:
  print('Its Moderate')
elif temp <= 25:
  print('Its warm')
else:
  print('Its Hotttt!')
```

→ Change the temperature values and observe the output.

❖ for Loop

Exercise 1: Print each character in a string.

→ Define the string in “string” variable. e.g.: *string = ‘Hello, World’*
→ Use for loop to iterate through each character in the string. e.g.: *for char in string:*
→ Print each character in the string during each iteration of the loop. e.g.: *print(char)*
→ Final code must look something like this:

```
string = 'Hello,World'

for char in string:
    print(char)
```

→ Change the character in string and observe the output.

Exercise 2: Create a list of squares of the first 10 natural numbers.

→ Generate a list of squares of the first 10 natural numbers by iterating each value “I” in the range from 1 to 10 (inclusive) and calculates its square i ** 2. The resulting squares are collected into a list called squares. e.g.: *squares = [i ** 2 for i in range(1, 11)]*
→ Print the list. e.g.: *print(squares)*
→ Final code must look something like this:

```
squares = [i ** 2 for i in range(1, 11)]
print(squares)
```

❖ while Loop

Exercise 1: Add numbers until sum reaches 100.

→ Start by initializing two variables “total_sum” to keep track of the sum of numbers and “current_number” to represent the current number to be added. e.g.:

```
total_sum = 0
current_number = 1
```

→ Use a while loop to add numbers until the “total_sum” reaches or exceeds 100. Inside the loop:

- Add the “current_number” to the “total_sum”.
- Increment the “current_number” by 1. e.g.

```
while total_sum < 100:
total_sum += current_number
current_number += 1
```

→ The loop will continue until the total_sum reaches or exceeds 100. After the loop, print the final total_sum.e.g.: *print(‘The sum of numbers until reaching 100 is:’, total_sum)*

→ Final code must look something like this:

```
total_sum = 0
current_number = 1

while total_sum < 100:
   total_sum += current_number
   current_number += 1

print('The sum of numbers until reaching 100 is:', total_sum)
```

❖ **Infinite Loops and break**

Exercise : Guessing game with break on correct guess.

→ Follow along the step mentioned in Control Structures.ipyb file.

DEMO

❖ **Basics of Git version control in VS Code**

NOTE: Make sure Git had been installed

→ On VS Code: Open Folder
- File > Open Folder (Ctrl+K Ctrl+O)
- Select the folder you want to open in VSCode.

→ Source Control
- View the source control panel by navigating to View > Source Control (SCM) (or use the shortcut Ctrl+Shift+G).
- This panel displays the status of files in the repository and allows you to perform version control operations.

→ Initialize Repository
- To initialize a repository, navigate to the folder you want to initialize in VSCode.
- Open the Command Palette by navigating to View > Command Palette (or use the shortcut Ctrl+Shift+P).
- Type "Git: Initialize Repository" and select the option to initialize the repository. By default, the main branch is created

→ Rename a Branch
- Open the Command Palette.
- Type "Git: Rename Branch" and select the option to rename the branch.

→ File Version Control Status
- In the source control panel, files are represented with different statuses:

U: Untracked file
A: Added file
M: Modified file

→ Commit File

- To commit changes to a file, navigate to the source control panel.
- Select the files you want to commit.
- Enter a commit message and click on the commit button (✓).

→ Create a Branch

- Open the Command Palette.
- Type "Git: Create Branch" and follow the prompts to create a new branch.

→ Diff Editor

- To view file differences, click on the file in the source control panel.
- Click on the inline view button to open the diff editor.

→ Stage Changes

- To stage changes, click on the stage changes button (+) next to the file in the source control panel.

→ Switch Branches

- View the current branch in the status bar (lower left corner).
- Click on the branch name to switch branches.

→ Merge Branch

- Navigate to Views and More Actions (...) > Branch > Merge Branch to merge a branch into the current branch.

→ Publish Branch to GitHub

- To publish a branch to GitHub, navigate to the source control panel.
- Click on the publish icon next to the branch name.

→ Clone Repository

- Open the Command Palette.
- Type "Git: Clone" and select the option to clone from a URL.
- Enter the URL of the repository you want to clone and follow the prompts.