SWE225 Introduction to Programming and Problem Solving

COLLEGE OF TECHNOLOGICAL INNOVATIONS (CTI)

Topics of Discussion

- Control Structure – Repetition
  - While
  - For
- Break and Continue Statements
- Nested Loops
- for-else Loop

Control Structure - Repetition

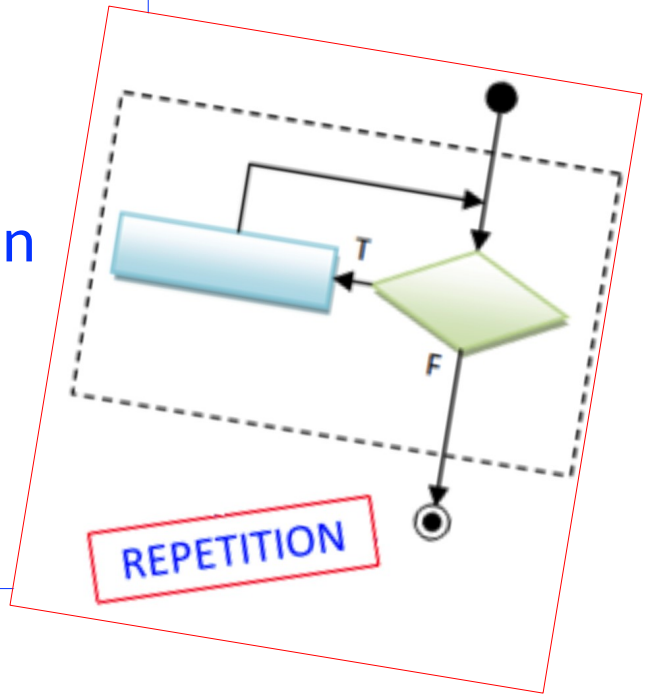
While Loop

Control Structure - Repetition

- As seen so far, program statements are executed in a sequence or they can branch based on conditions with the “if” statements.

- Repetition or Looping:
  - At times, the program is required to repeat statements for a number of times. This is called Repetition or Looping.

  - Looping alters the sequential flow of program execution by repeating statement(s) till a certain condition is met and then continues the sequential flow.



Class Work - Requirement for Loops

- Write an algorithm to add ANY 10 numbers given by the user.
- Unknown Inputs: <num1>, <num2>, <num3>, … <num10>
- Output: Sum of all 10 numbers

Algorithm

1. Start
2. Write "Enter 10 numbers"
3. Read num1, num2, num3, num4, num5, num6, num7, num8, num9, num10
4. iResult = num1+ num2 + num 3 + num4 + … + num10
5. Write iResult
6. Stop

Class Work - Requirements for executing a Loop

- **Algorithm**

1. Start
2. counter = 0
3. iResult = 0
4. Read num
5. iResult = iResult + num
6. counter = counter + 1
7. if (counter < 10) goto Step 4
8. Write iResult
9. Stop

Initialize the variable to control loop repetition

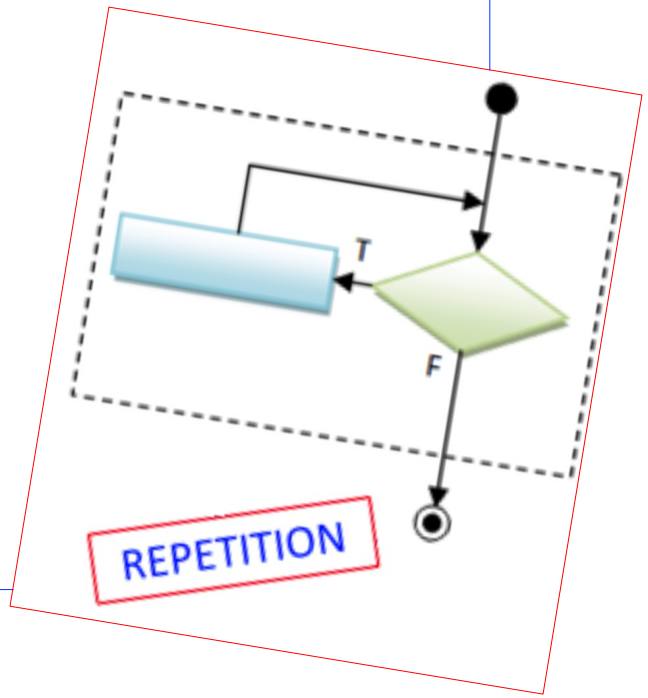
The Loop

Should the loop repeat or not!?

Counter:

iResult:

Read num:



The While Loop

- Syntax:

```
while expression:
    statement(s)
```

- `statement(s)` may be a single statement or a block of statements.
- The `expression` is a condition that evaluates to either
  - True (non-zero value) or
  - False.
- The loop iterates (repeats) while the expression evaluates to True.
- When the condition becomes false, program control passes to the line immediately following the loop statement(s).

Class Work

- Write a Python program using a while loop to find the SUM/average of ANY positive numbers given by the user.

- INPUT: ?
- OUTPUT: ?

Counter Controlled Loop

```
# While Loop
#

counter = 0                    # Initialize loop control variable

while counter < 10:            # Check the loop control variable
   ...
   counter = counter + 1       # Update the loop control variable
   ...

print("Bye")
```

Class Work

- Write a Python program to print the first 10 numbers (1 to 10)
- Python Program

```
# While Loop
#

counter = 0          # Initialize loop control variable

while counter < 10:      # Check the loop control variable
  counter = counter + 1 # Update the loop control variable
  print(counter)

print("Outside Loop")
```

Counter:

Print:

- Trace the changes in the variable **`counter`** for each iteration (repetition).

Class Work

- Python Program

```
# While Loop
#

counter = 1         # Initialize loop control variable

while counter < 10:       # Check the loop control variable
  counter = counter + 1 # Update the loop control variable
  print(counter)

print("Outside Loop")
```

- Predict the output of the above program
- Trace the changes in the variable **`counter`** for each iteration (repetition).

Class Work

- Python Program

```
# While Loop
#

counter = 0        # Initialize loop control variable

while counter < 10:      # Check the loop control variable
  print(counter)
  counter = counter + 1 # Update the loop control variable

print("Outside Loop")
```

- Predict the output of the above program
- Trace the changes in the variable **`counter`** for each iteration (repetition).

Class Work

- Python Program

```
# While Loop
#

counter = 10                    # Initialize loop control variable

while counter < 10:     # Check the loop control variable
  print(counter)
  counter = counter + 1 # Update the loop control variable

print("Outside Loop")
```

- Predict the output of the above program
- Trace the changes in the variable **`counter`** for each iteration (repetition).

Class Work

- Write a Python program to add the first 10 numbers (1 to 10)

- Python Program

```
# Add first 10 numbers

counter = 0
iResult = 0

while counter < 10:
   counter = counter + 1
   iResult = iResult + counter

print (”Sum is: “, iResult)
```

Class Work

- Write a Python program with while loop to add ANY 6 numbers given by the user.

- **Algorithm**
  1. Start
  2. counter = 0
  3. iResult = 0
  4. Write “Enter a number: ”
  5. Read num
  6. iResult = iResult + num
  7. counter = counter + 1
  8. if (counter < 6) goto Step 4
  9. Write iResult
  10. Stop

Initialization

Loop

Display Result

Class Work

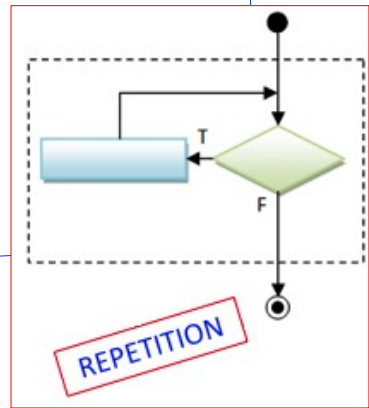
- **Algorithm**
  1. Start
  2. counter = 0
  3. iResult = 0
  4. Write "Enter a number: "
  5. Read num
  6. iResult = iResult + num
  7. counter = counter + 1
  8. if (counter < 6) goto Step 4
  9. Write iResult
  10. Stop

- **Python**

```
# Add ANY 6 numbers
#
# Initialize Variables
iCounter = 0
iResult = 0

# Loop
while iCounter <6:
    iNum = int(input("Give me a number: "))
    iResult = iResult + iNum
    iCounter = iCounter + 1

# Display Result
print("The sum is: ", iResult)
```



Class Work

- Write a Python program using a while loop to find the average of ANY 6 numbers given by the user.

- INPUT: ?
- OUTPUT: ?

Class Work

- Write a Python program using a while loop to find the average of *N* numbers.

- INPUT: ?
- OUTPUT: ?

Class Work

- Write a Python program using a while loop to print the sum of all even numbers less than 50 (inclusive).

- INPUT: ?
- OUTPUT: ?

Create a Turtle

- The use of turtle graphics is an interesting application of using loops.
- The turtle has three attributes: a location, an orientation (or direction), and a pen. The pen, too, has attributes: color, width, and on/off state (also called *down* and *up*).
- turtle allows programmers to control one or more turtles in a two-dimensional space
- You can draw complex shapes like squares, triangles, circles and other composite figures

- Creating a turtle graph
- Test this Python Program

```
import turtle                    # Allows us to use turtles

window = turtle.Screen()         # Creates a playground for the turtle
mat = turtle.Turtle()   # Create a turtle
mat.shape("turtle")     # Create a shape
mat.color("blue")       # Select the turtle color


window.mainloop()   # Pause the Window
```

Turtle Moves

```
import turtle                    # Allows us to use turtles

# Create Window
window = turtle.Screen() # Creates a playground  for turtle

# Create Turtle
mat = turtle.Turtle()
mat.shape("turtle")
mat.color("blue")

# Move Turtle
mat.forward(50)
mat.left(90)
mat.forward(50)
mat.circle(50)

window.mainloop()
```

Turtle Loops

```
import turtle

# Create Window
window = turtle.Screen()

# Create Turtle
sam = turtle.Turtle()
sam.shape("turtle")

# Loop Turtle
loopCount = 1
while loopCount <= 12:
    sam.left(30)
    sam.forward(50)
    sam.left(120)
    sam.forward(50)
    loopCount=loopCount+1
```

```
# Final moves
sam.backward(5)
sam.left(90)
sam.forward(20)

window.mainloop()
```

Home Work

- Write a Python program using a while loop to print the multiplication table of a number N given by the user.
- Expected Output:

  Which number for printing multiplication table? 2

  2 * 1 = 2

  2 * 2 = 4

  2 * 3 = 6

  2 * 4 = 8

  2 * 5 = 10

  2 * 6 = 12

  2 * 7 = 14

  2 * 8 = 16

  2 * 9 = 18

  2 * 10 = 20

Control Structure - Repetition

For Loop

For Loop - Syntax

- Syntax:

```
for [iterating variable] in [sequence]:
    [statements to repeat]
```

- The "`statements to repeat`" are repeated until the `sequence` is finished

- Try:

```
for i in range(0,15,1):
    print(i)
```

For Loop with range()

- In loops, range() is used to control how many times the loop will be repeated.

```
for i in range(6):
    print(i)
```

- This for loop sets up i as its iterating variable,
- The default starting value is 0
- The range is till 6 (excluding 6)

For Loop with range()

- What is the output?

```
for i in range(20,25):
    print(i)
```

- What is the output?

```
for i in range(100,0,-10):
    print(i)
```

For Loop and the Turtle

```
import turtle

# Create Window
window = turtle.Screen()

# Create Turtle
sam = turtle.Turtle()

# Loop Turtle
for i in range(0,11):
    sam.left(30)
    sam.forward(50)
    sam.left(120)
    sam.forward(50)

window.mainloop()
```

Loop through String

- The for loop is also used to iterate through characters of a string

```
# Loop through a string
#
stream = 'Software'
for letter in stream:
    print(letter)
```

For Loop and Sequential Data Types

- Rather than iterating through a range(), you can define a list and iterate through a List.

```
# Loop through a list
#
weekDays = ['sunday', 'monday', 'tuesday', 'wednesday', 'thursday','friday', 'saturday']

for day in weekDays:
    print(day)
```

The break Statement

- The `break` statement is used to exit a loop when a condition is triggered.
- The `break` statement is written inside the loop block of code usually with an if statement.

```
# break statement
#


for number in range(10):

    if number == 5:
        break

    print('Num is '+ str(number))

print('Out of loop')
```

The continue Statement

- The `continue` statement is used to skip a loop iteration when a condition is triggered.

- The `continue` statement is written inside the loop block of code usually with an if statement.

```
# break statement
#

for number in range(10):

    if number == 5:
        continue

    print('Num is '+ str(number))

print('Out of loop')
```

Nested Loops

- A nested loop is a loop that occurs within another loop
- The program first encounters the outer loop, executing its first iteration.
  - This first iteration triggers the inner, nested loop, which runs to completion.
- Then the program returns back to the outer loop, executing the second iteration.
  - This second iteration triggers the inner, nested loop which rims to completion.
- This goes on till the outermost loop completes all its iterations.

```
# Nested Loop
numList = [0, 1, 2]
alphaList = ['a', 'b', 'c']

for number in numList:
    for letter in alphaList:
        print(number, letter)

print ("Outside Loop!")
```

The for – else Loop

- The **`for`** loops can also have an **`else`** clause
- The **`else`** clause executes when the loop completes normally.
- This means that the loop did not encounter any break.

Syntax:

```
for <variable> in <sequence>:
    <statements>
else:
    <statements>
```

The for – else Loop

```
# Python program to check if a given number is prime or not

# Input from the user
num = int(input("Enter a number: "))

# Prime numbers are greater than 1
if num > 1:

    # check for factors
    for i in range(2, num):
        if (num % i) == 0:
            print(num, "is not a prime number")
            print(i, "times", num // i, "is", num)
            break
    else:
        print(num, "is a prime number")

# if input number is less than or equal to 1, it is not prime
else:
    print(num, "is not a prime number")
```

Home Work

- What is the output? (Work out on paper)

```
# Test yourself
x = 4
y = 0

while x>=0:
    x = x-1
    y = y+1
    if x==y:
        continue
    else:
        print (x,y)
```

Home Work

- Write a Python program to use the while loop to find the factorial of a given number.

- Examples:
  - Factorial of 3 is 6: 3! = 1*2 *3 = 6
  - Factorial of 5 is 120: 5!= 1*2*3*4*5=120

Home Work

- Write a program to ask the user to keep entering as many numbers as she/he wants.

- When the user finishes, your program should report:

  - How many positive numbers?
  - How many negative numbers?
  - How many even numbers?
  - How many numbers divisible by 7?

Home Work

- Write a program to find all the prime numbers between 1 and 500.
- *Note: make use of break, continue, nested loops as required.*

In Summary

- Control Structure – Repetition
- Understanding the Syntax of
  - While loop
  - For loop
- Understanding the Start, Stop and Step conditions for loop execution
- Break and Continue Statements
- Nested Loops
- For-else Loop