

Repetition: Loop Types

CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I
Fall 2020

Jonathan Hudson, Ph.D
Instructor
Department of Computer Science
University of Calgary

Tuesday, September 8, 2020

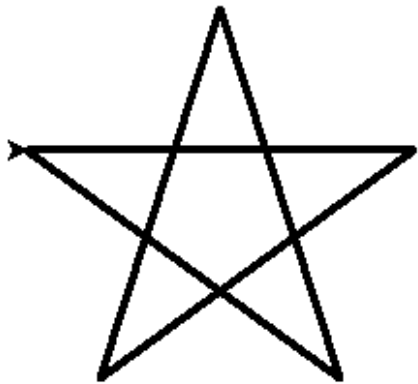


Review

- So far, we have learned...
 - How to use variables
 - Read values from the user
 - Make decisions
 - Compute a result
 - Output a result

- What if we want to perform a task several times?

```
alex.forward(200)
alex.right(144)
alex.forward(200)
alex.right(144)
alex.forward(200)
alex.right(144)
alex.forward(200)
alex.right(144)
alex.forward(200)
alex.right(144)
```



Loops

- Computer programs are very good at performing tasks over and over
- If part of the code is repeated it probably should be a loop

Loop Terminology

- Body of the Loop:
 - simple or compound statement that is repeated
- Loop Condition:
 - a Boolean expression
 - tested to determine if the loop will continue executing

Loop types

Post-test loops

- Checking the looping condition *after* executing the body of the loop.
- The loop body executes at least *one* time.

Pre-test loops

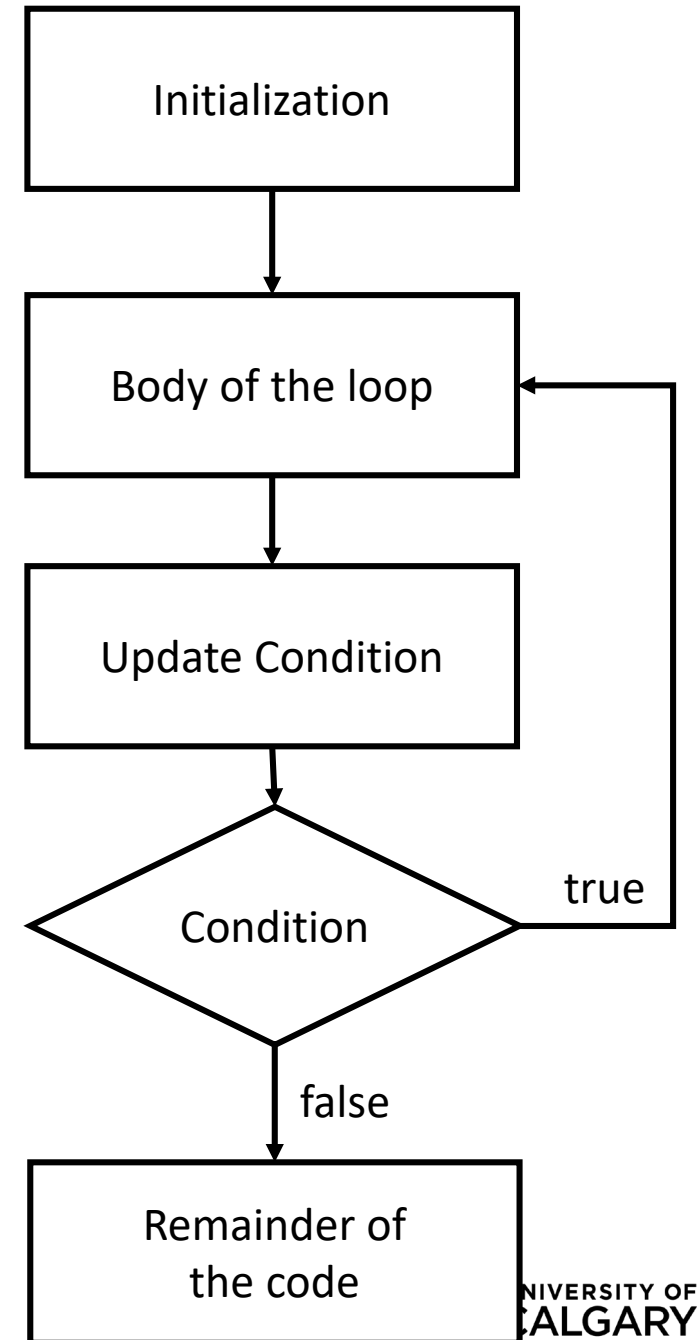
- Checking the looping condition *before* executing the body of the loop.
- The loop body executes *zero or more* times.

Post-Test Loop

Not in Python

Post-test loops

1. Initialize the loop control
2. Execute the body of the loop
(the part to be repeated)
3. Update the loop control
4. Check the condition
 - **False** → stop the loop and go to the rest of program
 - **True** → repeat from step 2



Post-test loops

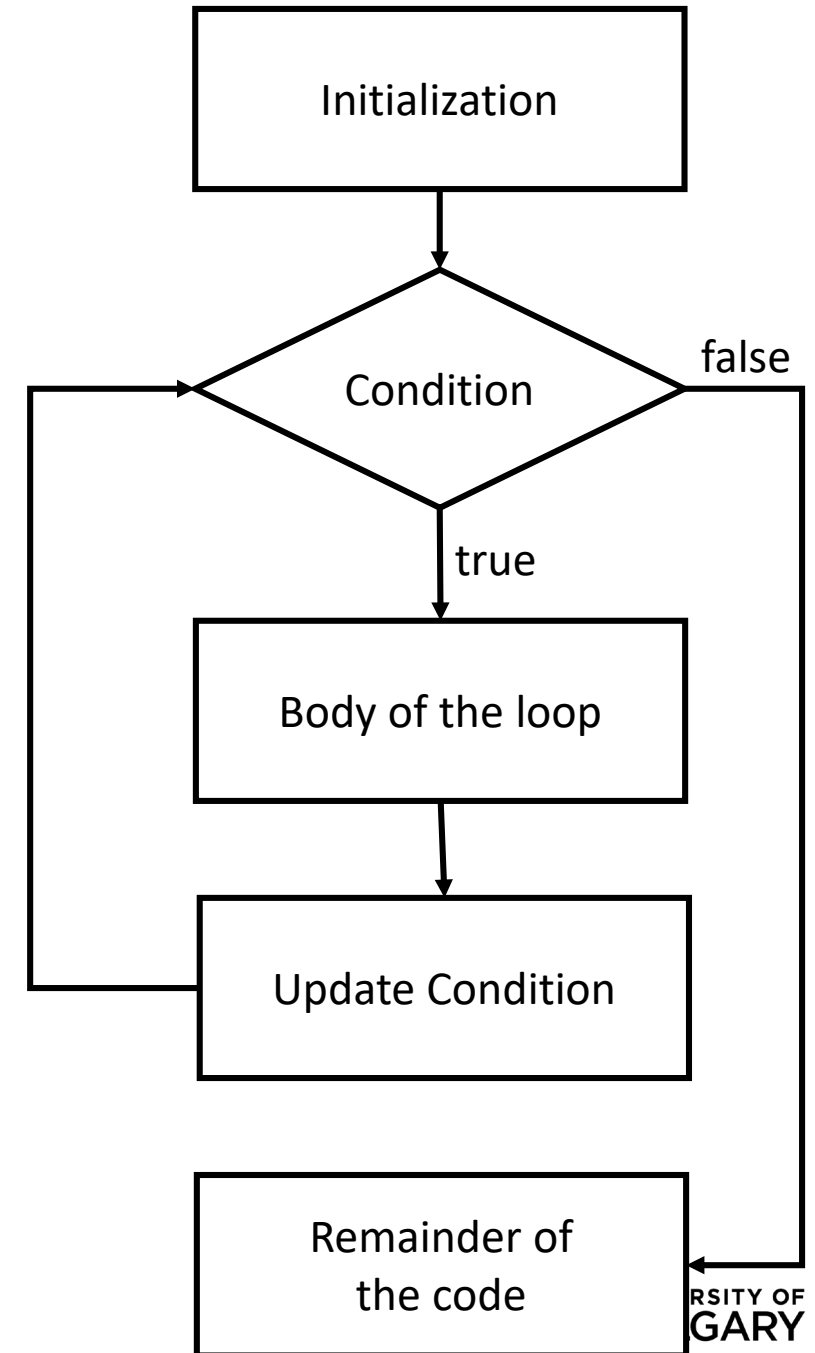
- The looping condition is checked **after** the body is executed
- The loop body will be executed **at least once**.
- **Python does not provide post-test loop**, common in Java and C/C++

Pre-Test Loop

In Python!

Pre-test loops

1. Initialize loop control
2. Check the condition
 - False** → stop the loop, skip the body, go to the rest of program
 - True** → Execute the body of the loop
4. Update the loop control
5. Repeat from step 2



Pre-test loops in Python

- Condition is checked **before** the body is executed
- The loop body will be executed for **zero or more times**
- Available loops in Python
 - **for** loops
 - **while** loops

While loops

While loop (indeterminate)

- Executes as long as some **condition** is true
- A pre-tested loop
 - loop **condition** is tested before the loop executes the first time
- General form:

```
while condition:  
    body
```

While loop

- When it is not known how many times to repeat → consider a while loop
- Here 0 is a sentinel value (a value that indicates loop completion)

- Example:

```
userinput = int(input('Please enter number to add to sum'))
sum = 0
while (userinput != 0):
    sum = sum + int(userinput)
    userinput = input('Plz enter a number, '0' to end')
```

While loop

- When it is not known how many times to repeat → consider a while loop
- Here 0 is a sentinel value (a value that indicates loop completion)
- Example:

```
Initialization ← userInput = int(input('Please enter number to add to sum'))  
← sum = 0  
while (userinput != 0):  
    indentation ← sum = sum + int(userinput)  
    userInput = input('Plz enter a number, '0' to end')  
Body of the loop
```


Example while

```
#Start in middle
pointer.up()
pointer.goto(WIDTH/2,HEIGHT/2)

#Asking use for desired data
sXLocation = input("Enter new x coordinate in (800,600) window: ")
sYLocation = input("Enter new y coordinate in (800,600) window: ")
while sXLocation != "" and sYLocation != "":
    sColor = input("Enter color [1:red 2:green 3:blue otherwise:black]: ")

    x = int(sXLocation)
    y = int(sYLocation)

    if sColor == "1":
        pointer.color("red")
    elif sColor == "2":
        pointer.color("green")
    elif sColor == "3":
        pointer.color("blue")
    else:
        pointer.color("black")

    pointer.down()
    pointer.goto(x,y)
    pointer.up()
    sXLocation = input("Enter new x coordinate in (800,600) window: ")
    sYLocation = input("Enter new y coordinate in (800,600) window: ")
print("Done")
#Close the graphic window on user's click
screen.exitonclick()
```

While example

```
total = 0
count = 0
moreItems = True
while moreItems:
    price = float(input('Enter price of item (0 when done): '))
    if price != 0:
        count = count + 1
        total = total + price
        print('Subtotal: $', total)
    else:
        moreItems = False
average = total / count
print('Total items:', count)
print('Total $', total)
print('Average price per item: $', average)
```

For loops

For loop (determinate)

- A counting loop
 - Typically used when we know how many times we need to perform a task in advance
 - A pre-tested loop
- General form:

```
for variable_name in list:  
    body
```

For loop

- When it is known **how many times to repeat** → consider a **for** loop

- Example:

```
last = int(input('Please enter number of iterations'))
total = 0
for i in range (1, last + 1, 1):
    total = total + i
```

For loop

- When it is known **how many times to repeat** → consider a **for** loop

- Example:

```
Initialization ← last = int(input('Please enter number of iterations'))
                ← total = 0
                for i in range (1, last + 1, 1): → i in (1, 2, 3)
                    total = total + i
                    ← indentation
                    ← Body of the loop
```

What was range()?

Python specific integer loop function

Loops in Python - *for* Loop – range()

- *range(start, end, step)* is a function that can take up to 3 parameters and returns a range of integer values.
 1. **start**: is an *optional* parameter (default is 0).
 - It represents the start of the range.
 2. **end**: is a **required** parameter, which represents the end of the range.
 - **The end value itself is excluded from the range.**
 3. **step**: is an *optional* parameter (default is 1).
 - It represents the increment value.

Step Values

- Range is flexible
 1. With one parameter
 - Counts from 0 up to (but not including) the number provided
 2. With two parameters
 - Counts from the first number to the second number (exclusive), increasing by one each time
 - Generates the empty list if the second number is less than or equal to the first
 3. With three parameters
 - Counts from the first number to the second (exclusive), increasing by the third

Loops in Python - *for* Loop – range()

- What does the range(5) return?
- What does range(1, 6) return?
- What does range(0, 5, 2) return?

Loops in Python - *for* Loop – range()

- What does the range(5) return?
 - start = 0, end = 5, step = 1
- What does range(1, 6) return?
 - start = 1, end = 6, step = 1
- What does range(0, 5, 2) return?
 - start = 0, end = 5, step = 2

Loops in Python - *for* Loop – range()

- What does the range(5) return?
 - A range object representing 0, 1, 2, 3, 4
- What does range(1, 6) return?
 - A range object representing 1, 2, 3, 4, 5
- What does range(0, 5, 2) return?
 - A range object representing 0, 2, 4

Example `range()` for loops

Example for

```
#Start in middle
pointer.up()
pointer.goto(WIDTH/2,HEIGHT/2)

sLines = input("Enter number of lines to draw: ")
iLines = int(sLines)

for i in range(0,iLines,1):
    #Asking use for desired data
    sXLocation = input("Enter new x coordinate in (800,600) window: ")
    sYLocation = input("Enter new y coordinate in (800,600) window: ")
    sColor = input("Enter color [1:red 2:green 3:blue otherwise:black]: ")

    x = int(sXLocation)
    y = int(sYLocation)

    if sColor == "1":
        pointer.color("red")
    elif sColor == "2":
        pointer.color("green")
    elif sColor == "3":
        pointer.color("blue")
    else:
        pointer.color("black")

    pointer.down()
    pointer.goto(x,y)
    pointer.up()

print("Done")
#Close the graphic window on user's click
screen.exitonclick()
```

Range

```
strMaxNumber = input("Please enter a number:")
maxNumber = int(strMaxNumber)
sum = 0
for i in range(0, maxNumber+1, 2):
    print (i)
print("__")
for i in range(1, maxNumber+1, 2):
    print (i)
print("__")
for i in range(maxNumber, 0, -2):
    print (i)
```

```
Please enter a number:10
0
2
4
6
8
10
—
1
3
5
7
9
—
10
8
6
4
2
```


Example for list loops

For each character in string

- A string is a list of characters we can loop through

```
for c in "Hello World":  
    print(c)
```

For in list of strings



```
for f in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:  
    print("Hi", f, "Please come to my party on Saturday")
```

```
Hi Joe Please come to my party on Saturday  
Hi Amy Please come to my party on Saturday  
Hi Brad Please come to my party on Saturday  
Hi Angelina Please come to my party on Saturday  
Hi Zuki Please come to my party on Saturday  
Hi Thandi Please come to my party on Saturday  
Hi Paris Please come to my party on Saturday
```

Onward to ... loop usage.

Jonathan Hudson
jwhudson@ucalgary.ca
<https://pages.cpsc.ucalgary.ca/~hudsonj/>



UNIVERSITY OF
CALGARY