

# Repetition: Loop Usage

---

**CPSC 217: Introduction to Computer Science for Multidisciplinary Studies I**  
**Jul 2021 - CBE**

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

Wednesday, June 2, 2021

*Copyright © 2021*



# Compare Loop Types

---

For vs While

# Loops in Python – Developing for/while

---

The following are equivalent loops:

```
sum = 0

for i in range(0,10,1):
    sum = sum + i

print(sum)
```

```
sum = 0
i = 0
while i < 10:
    sum = sum + i
    i = i + 1
print(sum)
```

# Loops in Python – Developing for/while

The following are equivalent loops:

```
sum = 0
for i in range(0, 10, 1):
    sum = sum + i
print(sum)
```

```
sum = 0
i = 0
while i < 10:
    sum = sum + i
    i = i + 1
print(sum)
```

# Break/Continue

---

# Break and Continue

---

- Allow a loop iteration to end prematurely
- `break`
  - Entire loop ends immediately
  - Execution continues at the first statement after the loop body
- `continue`
  - Current iteration ends immediately
  - Execution returns to the top of the loop
    - In a for loop, the next item in the list is used

# Break and Continue

---

```
i = 0
while i <= 10:
    if i == 5:
        break
    print(i)
```

0, 1, 2, 3, 4

ends loop at 5 before print

```
i = 0
while i <= 10:
    if i == 5:
        continue
    print(i)
```

0, 1, 2, 3, 4, 6, 7, 8, 9, 10

skips rest of body before print at 5

# Nesting

---



# Nested loops

---

- Loops are powerful components in programming
- A loop can be the body of another loop, and so on
- Different types of loops can be combined together

**while** (logical expression):

first part of while loop body

**for** <variable> in <something that can be iterated>:

body of the for loop

remainder of the while loop body

remainder of the program

**while** (*logical expression*):

first part of while loop body

**while** (*logical expression*):

*body of the inner while loop*

*remainder of the outer while loop*

*remainder of the program*

#outer loop

#inner loop

# Nested loops

```
while (logical expression):                                # outer while loop
    for <variable> in <something that can be iterated>:
        while (logical expression):                        # inner while loop
            body of the inner while loop
        remainder of the for loop
    remainder of the outer while loop
remainder of the program
```

Indentation is critical

```
while (logical expression):                                # 1st while loop
    while (logical expression):                            # 2nd while loop
        while (logical expression):                        # 3rd while loop
            body of the 3rd while loop
        remainder of the 2nd while loop
    remainder of the 1st while loop
remainder of the program
```

**you do need to make sure your program is still readable.**

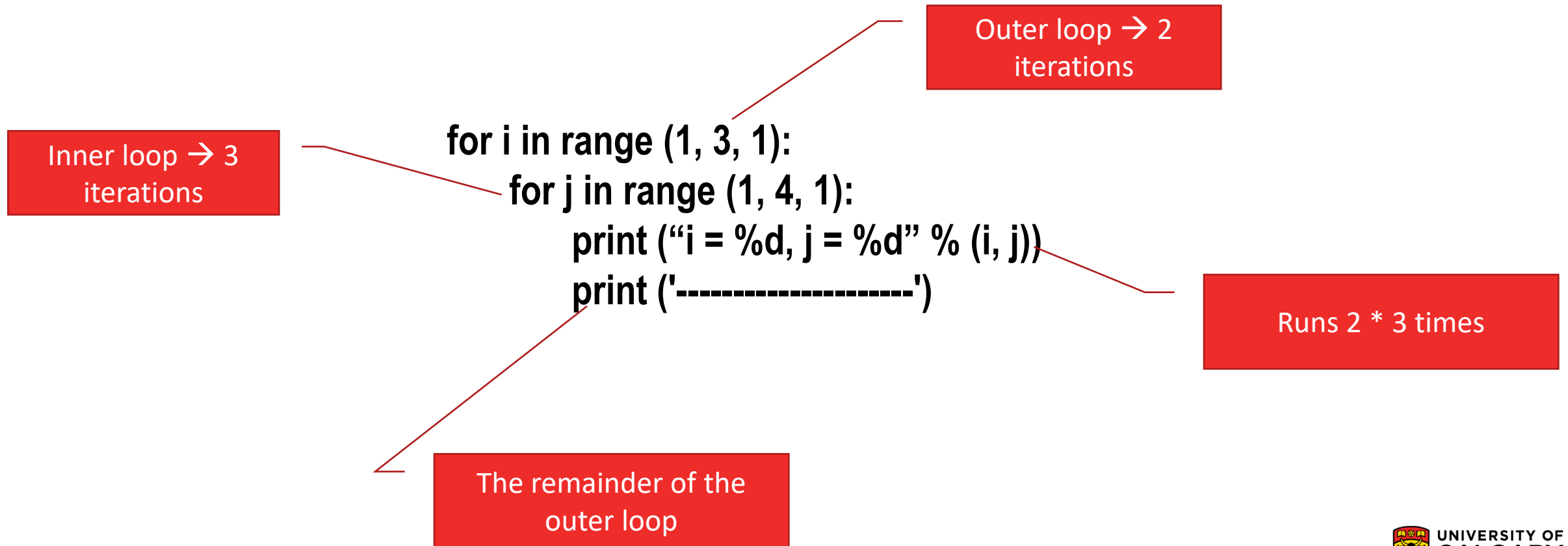
# Example

---

```
for i in range (1, 3, 1):  
    for j in range (1, 4, 1):  
        print ("i = %d, j = %d" % (i, j))  
        print ('-----')
```

# Example

---



# Loop Errors


---

# Infinite loop

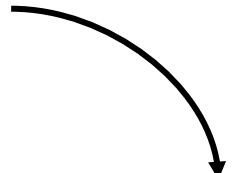
---

- When the looping condition is always satisfied → Loop never ends
- Caused by logical error:
  - The loop control does not get updated (**update error**)
  - The update will always satisfy the loop condition (**update error**)

```
i = 1
while (i <= 10):
    print ('i = %d' % (i))
    i = i + 1
```



```
i = 1
while (i <= 10):
    print ('i = %d' % (i))
    i = i - 1
```



- To stop an infinite loop use “Ctrl + C”
  - (CMD C on Apple)

# Erroneous loops

---

- The looping condition is not met before entering the loop.
- **A type of initialization error**
- Example:

```
i = 10
while (i < 10):
    print ('i = %d' % (i))
    i = i + 1
```

```
for i in range (5, 0, 2):
    print ('i = %d' % (i))
```

```
range (1, 4, 1) → (1, 2, 3)
range (4, 1, -1) → (4, 3, 2)
range (1, 5, 2) → (1, 3)
range (5, 0, 2) → ???
```

# Other Errors

---

- Off-by-one errors (the loop runs the wrong amount of times)
- Initialization Errors: Incorrect first value for loop control (either doesn't enter [erroneous] or enters with wrong value)



# Testing Loops

---

# Testing loops

---

- Make sure the loop executes the proper number of times.
- Test conditions:
  - Loop does not run
  - Loop runs exactly once
  - Loop runs exactly N times

# Tracing

---

- Tracing code:
  - Examine each statement in sequence
  - Perform whatever tasks the statement requires, recording values of interest
    - Usually requires that the value of each variable is recorded
  - Result of tracing could be the value of one or more variables, or the output generated
- **Very** important skill for debugging!
- Can be done by hand, or by using print statements to display intermediate values during the execution of the loop

# Loop Practice

---

# Practice - Multiplication Table

---

Produce a multiplication table from **1** to some **value** inputted by user:

```
max_multiplier = int(input("Enter the maximum multiplier: "))

for i in range(1, max_multiplier+1):
    row = ""
    for j in range(1, max_multiplier+1):
        row += str(i*j) + "\t"
    print(row)
```

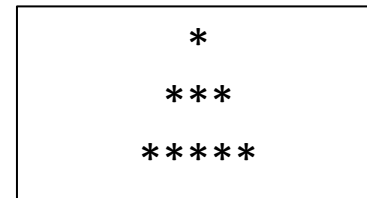
1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

# Christmas tree

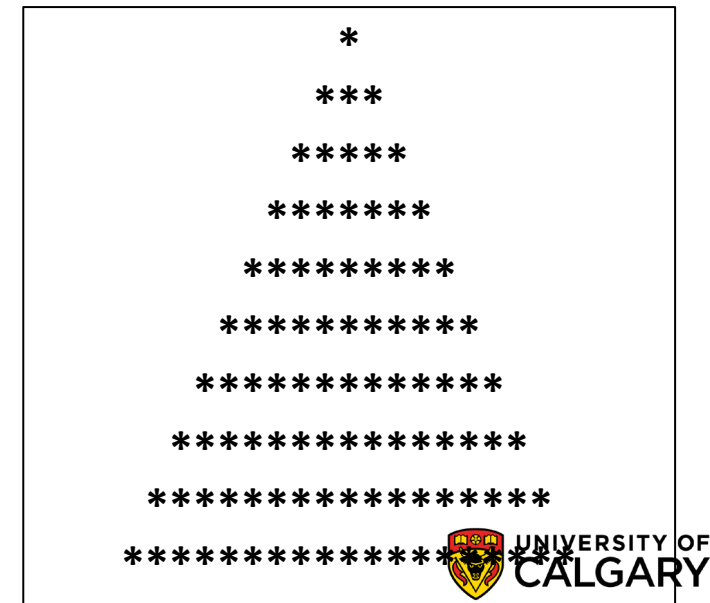
---

- Write a program that will print a triangle of a height provided by the user. For example:

- If the height is 3, the triangle will look like:



- If the height is 10, the triangle will look like:



# Practice - Christmas Tree Solution 1

---

```
iHeight = int(input("Please enter the height of the triangle: "))

for i in range(iHeight):
    row = ""
    for j in range(iHeight-i-1):
        row += " "
    for j in range(i*2 +1):
        row += "*"
    print(row)
```

# Practice - Christmas Tree Solution 2

---

```
iHeight = int(input("Please enter the height of the triangle: "))

max_num_spaces = (2 * (iHeight - 1)) + 1
num_spaces = max_num_spaces // 2
for i in range(iHeight):
    num_astericks = max_num_spaces - 2 * num_spaces
    print(" " * num_spaces + "*" * num_astericks)
    num_spaces -= 1
```



# Onward to ... functions.

---

Jonathan Hudson  
[jwhudson@ucalgary.ca](mailto:jwhudson@ucalgary.ca)  
<https://pages.cpsc.ucalgary.ca/~hudsonj/>



UNIVERSITY OF  
CALGARY