

Loops In Python: Part 1

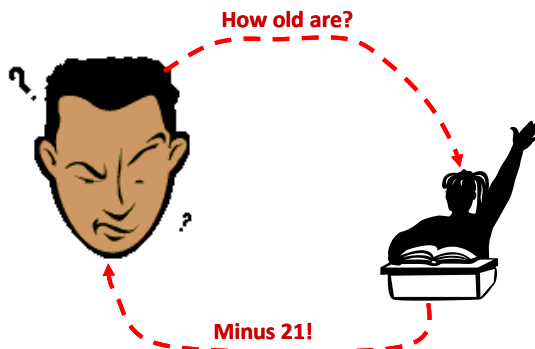
In this section of notes you will learn how to rerun parts of your program without duplicating instructions.

James Tam

Repetition: Computer View

- Continuing a process as long as a certain condition has been met.

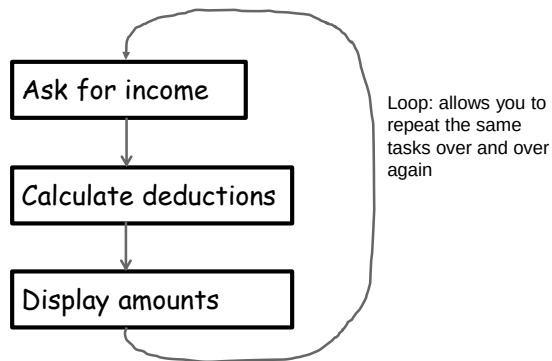
Ask for age as long as the answer is negative (outside allowable range)



James Tam

Looping/Repetition

- How to get the program or portions of the program to automatically re-run
 - Without duplicating the instructions
 - Example: you need to calculate tax for multiple people



James Tam

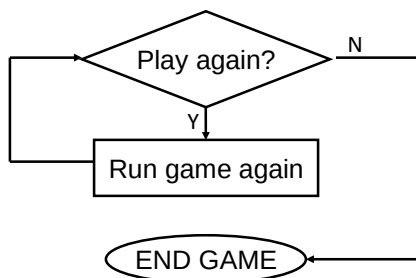
How To Determine If Loops Can Be Applied

- Something needs to occur multiple times (generally it will repeat itself as long as some condition has been met).
- Example 1:



Re-running the entire program

Flowchart



Pseudo code

While the player wants to play
Run the game again

James Tam

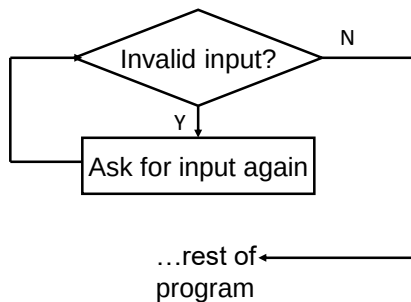
How To Determine If Loops Can Be Applied (2)

- Example 2:

```
Enter your age (must be non-negative): -1
Enter your age (must be non-negative): 37
Enter your height (must be non-negative):
```

Re-running specific parts of the program

Flowchart



Pseudo code

```
While input is invalid
    Prompt user for input
```

James Tam

Basic Structure Of Loops

Whether or not a part of a program repeats is determined by a loop control (typically the control is just a variable).

- Initialize the control to the starting value
- Executing the body of the loop (the part to be repeated)
- Update the value of the control
- Somewhere: Testing the control against a stopping condition (Boolean expression)
 - Without this test the loop will never end (endless loop)

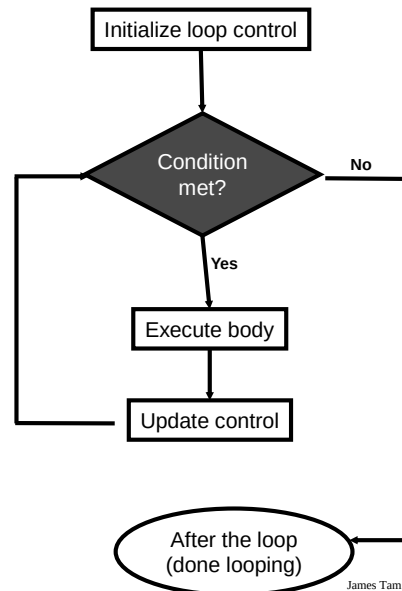
James Tam

Pre-Test Loops

Pre-test loops

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

1. Initialize loop control
2. Check if the repeating condition has been met
 - a. If it's been met then go to Step 3
 - b. If it hasn't been met then the loop ends
3. Execute the body of the loop (the part to be repeated)
4. Update the loop control
5. Go to step 2

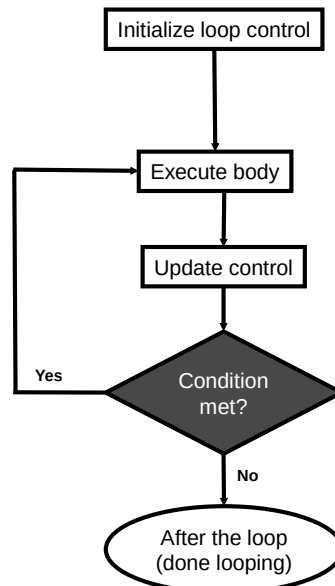


Post-Test Loops (Not Implemented In Python)

Post-test loops

- Checking the stopping condition *after* executing the body of the loop.
- The loop executes *one or more* times.

1. Initialize loop control (sometimes not needed because initialization occurs when the control is updated)
2. Execute the body of the loop (the part to be repeated)
3. Update the loop control
4. Check if the repetition condition has been met
 - a. If the condition has been met then go through the loop again (go to Step 2)
 - b. If the condition hasn't been met then the loop ends.



Loops In Python

- Pre-test: for, while
- Post-test: none

James Tam

The While Loop

- This type of loop can be used if it's *not known* in advance how many times that the loop will repeat (most powerful type of loop, any other type of loop can be simulated with a while loop).
 - It can repeat so long as some arbitrary condition holds true.

- Format:**

(Simple condition)

```
while (Boolean expression):  
    body
```

(Compound condition)

```
while ((Boolean expression) Boolean operator (Boolean expression)):  
    body
```

James Tam

The while Loop (2)

• **Program name:** 1while1.py

• **Learning objective:** a simple counting loop stepping through a sequence (1 - 3)

```
i = 1
while (i <= 3):
    print("i =", i)
    i = i + 1
print("Done!")
```

1) Initialize control

2) Check condition

3) Execute body

4) Update control

James Tam

The while Loop (2)

• **Program name:** 1while1.py

```
i = 1
while (i <= 3):
    print("i =", i)
    i = i + 1
print("Done!")
```

James Tam

Countdown Loop

- **Program name:** 2while2.py
- **Learning objective:** a simple counting loop stepping down through a sequence (3 - 1)

```
i = 3
while (i >= 1):
    print("i =", i)
    i = i - 1
print("Done!")
```

James Tam

Common Mistakes: While Loops

- Forgetting to include the basic parts of a loop.
 - **Updating the control**

```
i = 1
while(i <= 4):
    print("i =", i)
    # i = i + 1
```



```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
```

James Tam

Practice Exercise #1

- The following program that prompts for and displays the user's age.
- Modifications:
 - As long as the user enters a negative age the program will continue prompting for age.
 - After a valid age has been entered then stop the prompts and display the age.

```
age = int(input("Age: "))  
print(age)
```

```
Age: -1  
Age cannot be negative  
Age: -123  
Age cannot be negative  
Age: 27  
You are 27 years young :P
```

James Tam

The For Loop

- In Python a for-loop is used to step through a sequence e.g., count through a series of numbers or step through the lines in a file.

- Syntax:

```
for <name of loop control> in <something that can be iterated>:  
    body
```

- Program name: 3for1.py

- Learning objective: a simple for counting loop stepping through a sequence (1 - 3)

```
for i in range (1, 4, 1):
```

```
    print("i=", i)
```

```
print("Done!")
```

1) Initialize control

2) Check condition

4) Update control

3) Execute body

James Tam

Counting Down With A For Loop

- **Program name:** 4for2.py
- **Learning objective:** a simple counting loop stepping down through a sequence (3 - 1)

```
for i in range (3, 0, -1):  
    print("i = ", i)  
print("Done!")
```

James Tam

For Loop: Stepping Through A Sequence Of Characters

- **Recall:** A for-loop in Python can step through any iterable sequence (number sequence, characters in a string, lines in a file).

- **Program name:** 5for3.py

- Learning objective: a for loop stepping through a sequence in a string

We are taking the dog for a '

```
activity = input("What are you doing with dog now: ")  
print("We are taking the dog for a '", end="")
```

b-a-t-h-'

```
for ch in activity:  
    print(ch + "-", end="")  
print("")
```

James Tam

Erroneous For Loops (If There Is Time)

- The logic of the loop is such that the end condition has already been reached with the start condition.
- **Program name:** 6for_error.py
 - Learning objective: a loop that never executes

```
for i in range (5, 0, 1):  
    print("i = ",i)  
print("Done!")
```

```
[csc loops 18 ]> python for_error.py  
Done!
```

James Tam

Loop Increments Need **Not Be Limited To One**

- **While:** 7while_increment5.py

```
i = 0  
while (i <= 100):  
    print("i =", i)  
    i = i + 5  
print("Done!")
```

- **For:** 8for_increment5.py

```
for i in range (0, 105, 5):  
    print("i =", i)  
print("Done!")
```

```
i = 0  
i = 5  
i = 10  
i = 15  
i = 20  
i = 25  
i = 30  
i = 35  
i = 40  
i = 45  
i = 50  
i = 55  
i = 60  
i = 65  
i = 70  
i = 75  
i = 80  
i = 85  
i = 90  
i = 95  
i = 100  
Done!
```

James Tam

Sentinel Controlled Loops

- The stopping condition for the loop occurs when the 'sentinel' value is reached e.g. sentinel: number less than zero (negative)
- **Program name:** 9sum.py
 - Learning objective: loops that execute until the sentinel value has been encountered.

```
total = 0
temp = 0
while(temp >= 0):
    temp = input("Enter a non-negative integer (negative to end
    series): ")
    temp = int(temp)
    if (temp >= 0):
        total = total + temp
print("Sum total of the series:", total)
```

```
Enter a positive integer (negative to end series):1
Enter a positive integer (negative to end series):2
Enter a positive integer (negative to end series):3
Enter a positive integer (negative to end series):-1
Sum total of the series: 6
```

Q: What if the user
just entered a single
negative number?

James Tam

Sentinel Controlled Loops (2)

- Sentinel controlled loops are frequently used in conjunction with the error checking of input.
- **Example** (sentinel value is one of the valid menu selections, repeat while selection is not one of these selections):

10sentinel_controlled_loop.py

```
selection = " "
while selection not in ("a", "A", "r", "R", "m", "M", "q", "Q"):
    print("Menu options")
    print("(a)dd a new player to the game")
    print("(r)emove a player from the game")
    print("(m)odify player")
    print("(q)uit game")
    selection = input("Enter your selection: ")
    if selection not in ("a", "A", "r", "R", "m", "M", "q", "Q"):
        print("Please enter one of 'a', 'r', 'm' or 'q' ")
```

```
Menu options
(a)dd a new player to the game
(r)emove a player from the game
(m)odify player
(q)uit game
Enter your selection: x
Please enter one of 'a', 'r', 'm' or 'q' 
```

```
Menu options
(a)dd a new player to the game
(r)emove a player from the game
(m)odify player
(q)uit game
Enter your selection: A
```

Valid option
entered, loop ends

James Tam

Recap: What Looping Constructs Are Available In Python/When To Use Them

Construct	When To Use
Pre-test loops	You want the stopping condition to be checked before the loop body is executed (typically used when you want a loop to execute zero or more times).
• While	• The most powerful looping construct: you can write a 'while' loop to mimic the behavior of any other type of loop. In general it should be used when you want a pre-test loop which can be used for most any arbitrary stopping condition e.g., execute the loop as long as the user doesn't enter a negative number.
• For	• In Python it can be used to step through some sequence
Post-test: None in Python	You want to execute the body of the loop before checking the stopping condition (typically used to ensure that the body of the loop will execute at least once). The logic can be simulated with a while loop.

James Tam

After This Section You Should Now Know

- When and why are loops used in computer programs
- What is the difference between pre-test loops and post-test loops
- How to trace the execution of pre-test loops
- How to properly write the code for a loop in a program
- What is a sentinel controlled loop and when should they be employed

James Tam

Copyright Notification

- “Unless otherwise indicated, all images in this presentation are used with permission from Microsoft.”