Loops In Python: Part 1

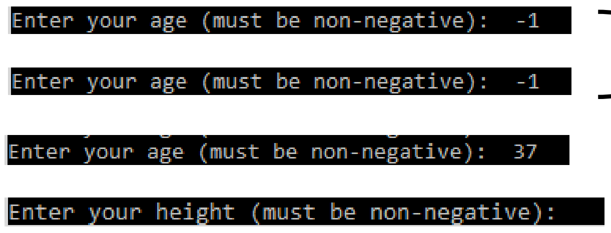
In this section of notes you will learn how to rerun parts of your program without duplicating instructions.

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)



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

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

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

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

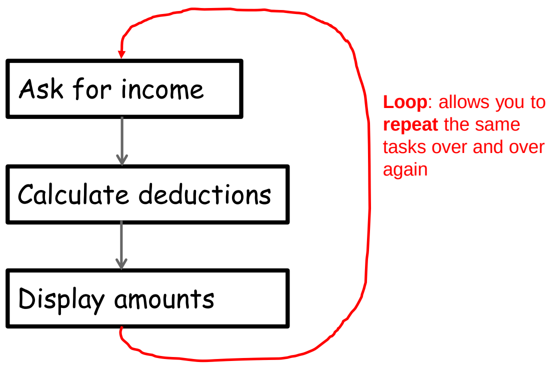
Condition met: age is negative (repeat prompt)

Condition no longer met: stop repetition

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

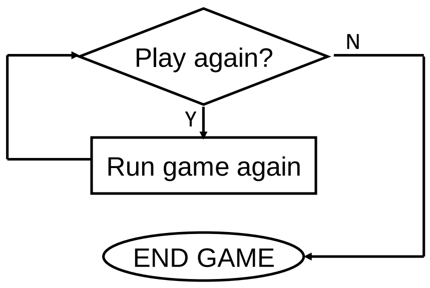

Ask for income
Calculate deductions
Display amounts
Loop: allows you to **repeat** the same tasks over and over again

James Tam

How To Determine If Loops Can Be Applied

- Something needs to occur multiple times (generally it will repeat itself as long as it’s true some condition has been met).
- **Example 1 (re-run an entire program)**:

Flowchart



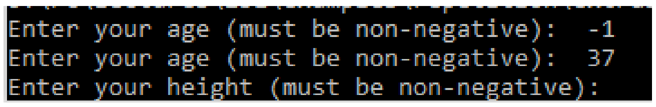
Pseudo code (code like format)

```
While the player wants to play
    Run the game again
End while
```

James Tam

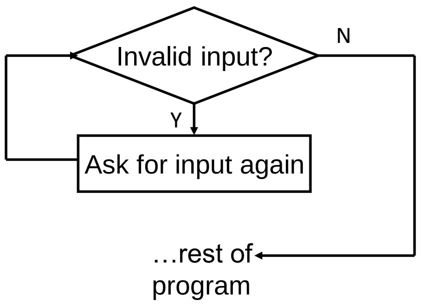
How To Determine If Loops Can Be Applied (2)

- **Example 2** (re-running specific parts of the program)



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

Flowchart



Pseudo code

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

James Tam

Basic Structure Of Loops

New term, loop control (typically the control is just a variable(s)): Determines if a part of a program repeats

- Example program counts through (and displays) the numbers from 1 – 10.
- Initialize the control to the starting value.
  `e.g. i = 1`
- Executing the body of the loop (the part to be repeated).
  - It's similar to the body of a branch except that it **s**.

  `e.g. print(i)`
- Update the value of the control
  `e.g. i = 1 + 1`
- Somewhere ('top' of the loop in the form of Boolean expression):
  - Testing the control against a stopping condition (Boolean expression)

  `e.g. while(i <= 10):`
  - Without this test the loop will never end (endless loop)

James Tam

Loops In Python

- `for`
  - **Python**: can be used when the program can step through ('iterate') through a sequence.
    - E.g. 1: count through a numerical sequence (1, 2, 3…)
    - E.g. 2: the sequence of characters in a string
    - E.g. 3: the sequence of lines in a file.
  - **Strength of python**:
    - With most other languages `for-loops` can only count through a numerical sequence (5, 25, 125…). Consequently referred to as "counting loops".
    - With python `for-loops` they can not only count through (iterate) a sequence but also iterate through other things as well e.g. read in lines in a text file
  - **Drawback with python**: Python `for-loops` can only count through a sequence using addition or subtraction (other operations not possible).
- `While`
  - The most flexible (powerful) type of loop.
  - It can be used almost any time repetition is needed.
    - Situations when it can't be used are very specific (when back tracking (going back after repetitions is done i.e. with 'recursion').

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 Boolean condition is true.
- **Format:**

(Simple condition)

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

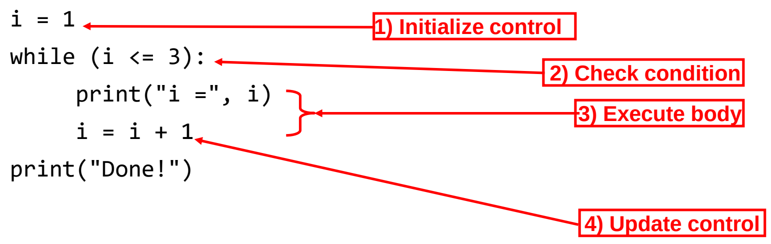
(Compound condition)

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

James Tam

The While Loop (2)

- **Program name:** `1while1_counting_up.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!")
```

James Tam

The While Loop (2)

- **Program name:** `1while1_counting_up.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!")
```

James Tam

Countdown Loop

- **Program name:** `2while2_counting_down.py`
- Learning objective: a simple counting loop stepping down through a sequence (3 - 1)

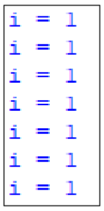
```
i = 3
while (i >= 1):
    print("i =", i)
    i = i - 1
print("Done!")
```

James Tam

Common Mistakes: While Loops

- **Name of the online example:** `3error_not_updating.py`
- Forgetting to include the basic parts of a loop.
  - **Not updating the control**

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



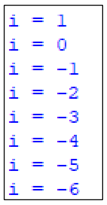
```
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
i = 1
```

James Tam

Common Mistakes: While Loops

- **Name of the online example:** `4error_errorenous_updating.py`
- Improperly implementing a basic parts of a loop.
  - **The updating of the control doesn't bring the value any closer to the stopping condition.**

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



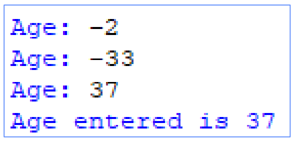
```
i = 1
i = 0
i = -1
i = -2
i = -3
i = -4
i = -5
i = -6
```

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: -2
Age: -33
Age: 37
Age entered is 37
```

James Tam

General Use: 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.
- **General syntax:**

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

- **Syntax, counting loop (steps through number sequence):**

```
for <name of loop control> in range():
    body
```

- **Example, counting loop (steps through number sequence):**

```
for i in range(1,4,1):
    print("i=", i)
```

James Tam

Example Use Of The For Loop

- **Program name:** `5for1_counting_up.py`
- Learning objective: a simple for counting loop stepping through a sequence (1 - 3)

1) Initialize control (include)

2) Check condition (exclude)

4) Update control

```
for i in range(1, 4, 1):
    print("i=", i)
print("Done!")
```

3) Execute body

James Tam

Counting Down With A For Loop

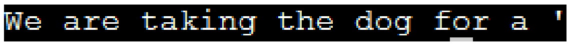
- **Program name:** `6for2_counting_down.py`
- Learning objective: a simple counting loop stepping down through a sequence (3 - 1)

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

James Tam

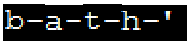
For Loop: Stepping Through A Sequence Of Characters

- Recall: A for-loop in Python can step through any iteratable sequence (number sequence, characters in a string, lines in a file).
- **Program name:** `7for3_iterating_string.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="")
```



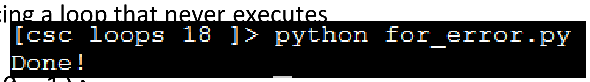
```
b-a-t-h-'
```

```
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.
  - Typically occurs when the programmer has combined a loop that combines counting up with a loop that counts down.
- **Program name:** `8Afor_error.py`
  - Learning objective: tracing a loop that never executes



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

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

```
#8Bwhile_equivalent_for_error.py
i = 5
while(i < 0):
    print("i = ",i)
    i = i + 1
print("Done!")
```

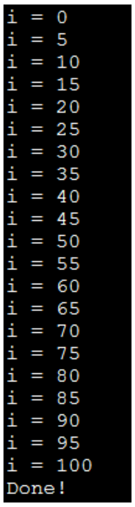
Loop Increments Need Not Be Limited To One

- **While:** `9while_increment5.py`

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

- **For:** `10for_increment5.py`

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

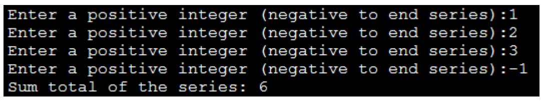


```
i = 0
i = 5
i = 10
i = 15
i = 20
i = 25
i = 30
i = 35
i = 40
i = 45
i = 50
i = 55
i = 60
i = 65
i = 70
i = 75
i = 80
i = 85
i = 90
i = 95
i = 100
Done!
```

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**: `11sentinel_sum.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
      sequence): ")
    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
```

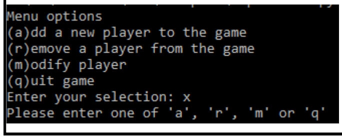
Q: What if the user just entered a single negative number?

James Tam

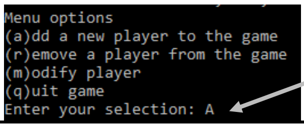
Sentinel Controlled Loops (2)

- 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): `12sentinel_controlled_menu.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

Post Test Loops

- Python doesn't happen to implement them but they are common in other languages and they can be useful for certain situations.
  - Recall: this is not a "python programming" course but instead it's a course where you learn basic programming principles (e.g. input, output, variables, constants, branching, loops etc.) without being limited by a particular language.
- When to use post loops: when you need a loop to always execute at least once (even if the Boolean expression evaluates to false the first time that the loop is encountered).
- The guaranteed execution of 1+ times occurs because the Boolean expression is checked after the body executes.

James Tam

Comparison Of Loop Types

- Pre-test loops (for, while):
  - Evaluates the Boolean expression **before** executing the body.
  - Executes zero or more times.
    - E.g.

```
age = 12
while (age < 0):
```

  - **Structure**:

```
while (BE):
    Body
```

- Post-test loops (nothing in python C, C++, Java has the do-while loop)
  - Evaluates the Boolean expression **after** executing the body.
  - Guaranteed to execute at least once.
  - Execute one or more times.
  - **Structure**:

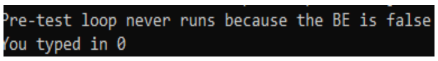
```
do:
    Body
while (BE):
```

James Tam

Examples Pre Vs. Post Test Loops (Java - For Illustration Only, From CPSC 233)

- **Pre-test**

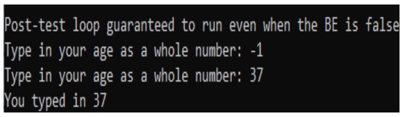
```
age = 0;
System.out.print("Pre-test");
while(age<0) {
    System.out.print("Age: ");
    age = userInput.nextInt();
}
System.out.println("You typed in
" + age);
```



```
Pre-test loop never runs because the BE is false
You typed in 0
```

- Post-test

```
age = 0;
System.out.print("Post-test");
do {
    System.out.print("Age: ");
    age = userInput.nextInt();
} while(age<0);
System.out.println("You typed in
" + age);
```



```
Post-test loop guaranteed to run even when the BE is false
Type in your age as a whole number: -1
Type in your age as a whole number: 37
You typed in 37
```

James Tam

'Simulating' A Post-Test Loop Using A `While`-Loop

- **'Prime' the loop control.**
  - Set the variable(s) to a starting value(s) to guarantee execution at the start.
- **Program name**: `13guaranteed_pre_test_execution.py`

```
age = -1
while (age < 0):
    print("Type in your age as a whole number: ", end = "")
    age = int(input())
print("You typed in %d"  %age);
```

James Tam

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

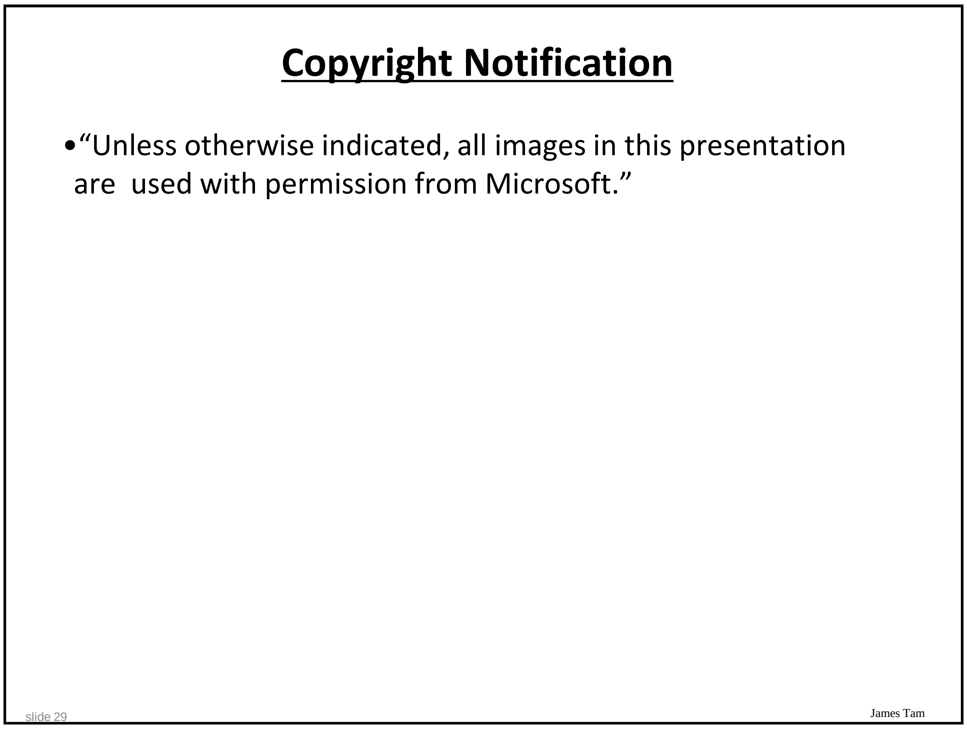
Construct	When To Use
Pre-test loops	You want the stopping condition to be checked before the loop body is executed (typically used when you want a loop to execute zero or more times).
• `While`	• The most powerful looping construct: you can write a 'while' loop to mimic the behavior of any other type 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

slide 29

James Tam