Loops In Python: Part 1

- In this section of notes you will learn how to rerun parts of your program without duplicating instructions
- Basic introduction into the use of the while-loop

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):
```

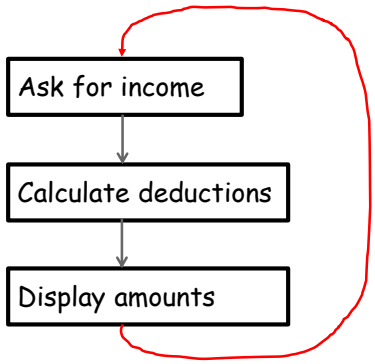
Condition met: age is negative (repeat prompt)

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



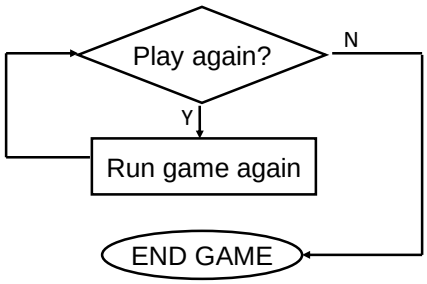
Loop: allows you to **repeat** the same tasks over and over again

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 (a Boolean expression) 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
```

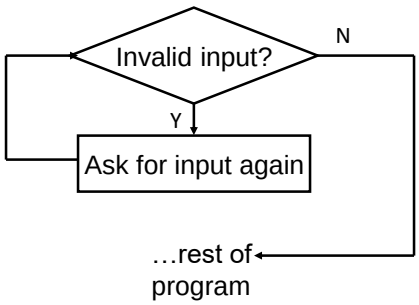
James Tam

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 **repeats code multiple times.**

  `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

- `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 during 'recursion' is needed).
- `for`
  - It will be covered in the next section.

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:**
- **Examples**

(Simple condition)

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

(Compound condition)

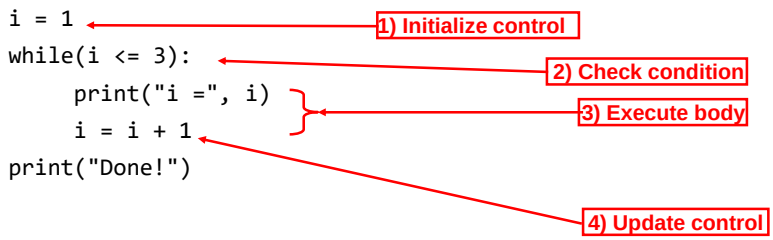
```
while((Boolean expression) Boolean operator (Boolean expression)):
      body
```

The Boolean expressions used for branches are similar in form to the ones used for loops if you need to imagine an example at this point.

James Tam

The `While` Loop (2)

- **Program name:** `1while1_counting_up.py`
- Learning objective: a simple counting loop stepping through a sequence (1 - 3)



James Tam

The `While` Loop (2)

- **Program name:** `1while_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:** `2while_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

Common Mistakes: While Loops

- Improperly implementing a basic parts of a loop.
  - **The initial value of the loop control is such that the Boolean expression evaluates to false (loop never executes).**

```
i = 5
while(i <= 4):
    print("i =", i)
    i = i - 1

#No example program: above code produces no visible output
```

James Tam

Common Mistakes: `While` Loops

- Loop doesn't run the expected number of times.
  - **"Off by one error": actual vs. expected number of times mismatched by one.**
  - [JT: because the **Boolean expression is explicit** when defining a WHILE-loop this problem is more common with FOR-loops].

```
#Loop runs 4 times, off by one trace rare for while-loops
i = 1
while(i <= 4):
    print("i =", i)
    i = i + 1

#Loop runs 5 times, off by one if person doesn't count zero.
i = 0
while(i <= 4):
    print("i =", i)
    i = 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: -2
Age: -33
Age: 37
Age entered is 37
```

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**: `5sentinel_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
```

```
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):
  `6sentinel_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

Copyright Notification

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.