GOVERNMENT OF
WESTERN AUSTRALIA

TAFE WA South Metropolitan

Loops in Python

(Session 6)

Review (String Methods)

- Python uses **+** symbol to *concatenates* strings.
- The *title method* returns a string where each word starts with an uppercase character.

Let’s look at an example:

```
>>> first_name = "monty"
>>> last_name = "python"
>>> print("Hello", first_name.title(), last_name.title())
Hello Monty Python
```

Overview

- Loops in Python
- Introducing While Loops
- Flowchart of While Loop
- User Input and While Loops
- Using break to Exit a Loop
- Using continue in a Loop
- while else

Loops in Python

- **while** Loop
  - used when asking for user input, when you do not know how many times the loop should run
  - while loop repeats a section of code until a specific condition is met
- **for** Loop
  - used for iterating over a sequence such as a string
  - does not require an indexing variable to be set beforehand
  - when you know how many times the loop should run

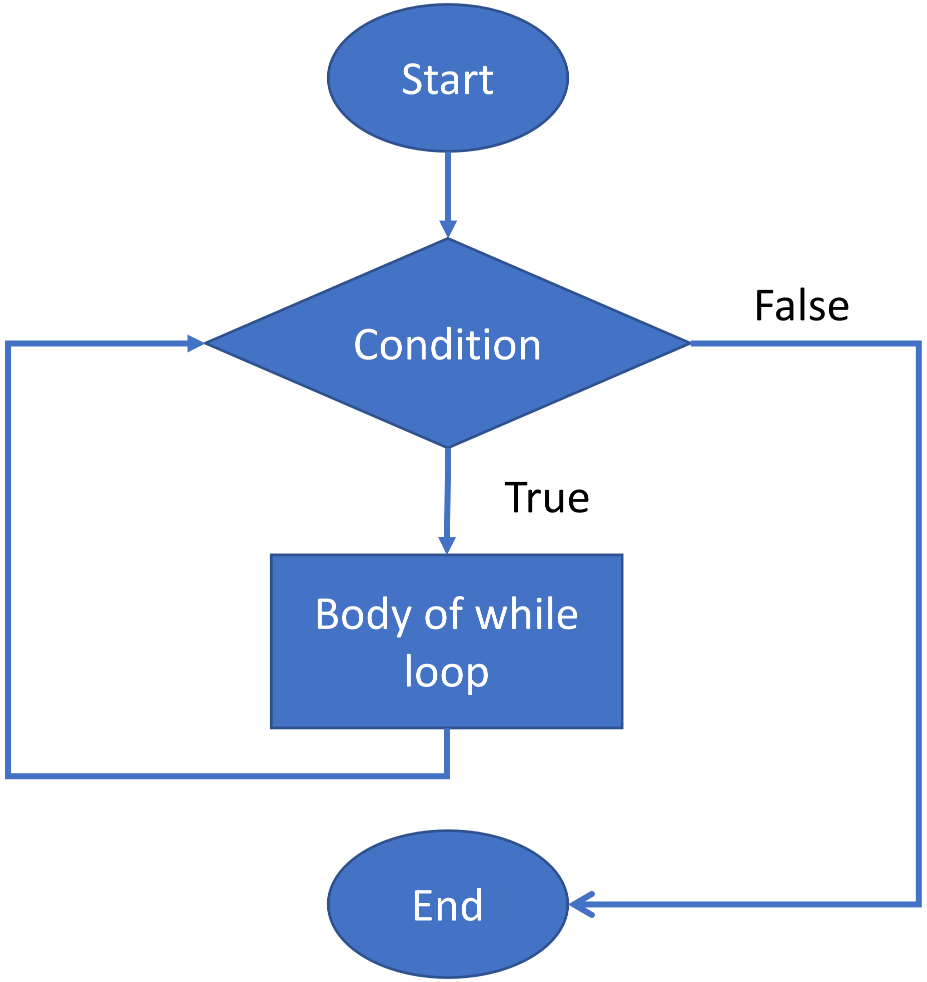
Introducing While Loops

- while Loops are used to execute a block of statements
- Statements will execute repeatedly until a given condition (conditional_expression) is **True**
- *Syntax of* while Loop in Python

```
while conditional_expression:
        body of while loop (block of statements)
```

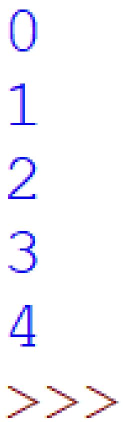
If the condition is False the body is not executed

While Loop Flowchart



While Loop Example

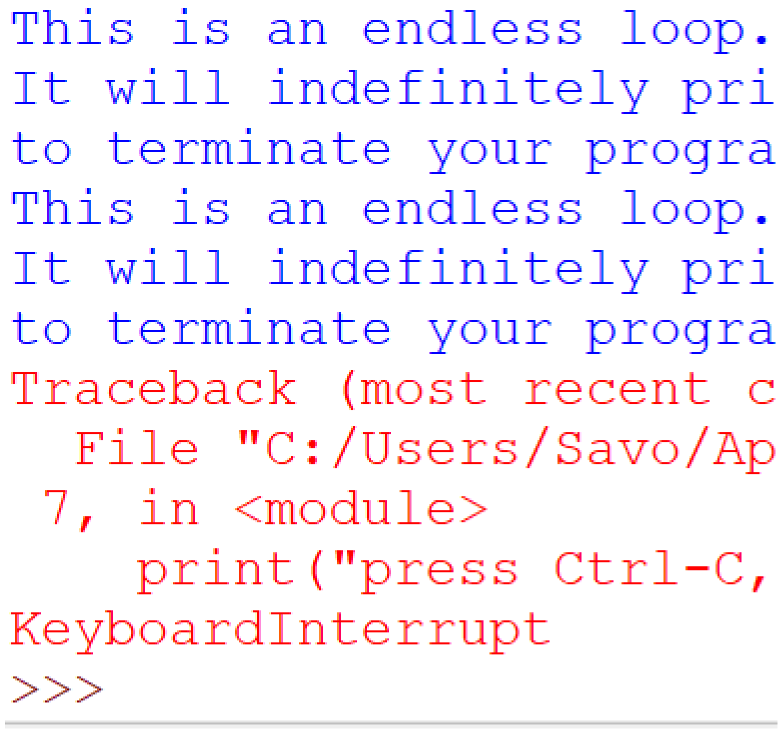
```
# while loop repeats the execution as long as i is less than 5
# failing to increase the value of i will result in an infinite loop
i = 0
while i < 5:
    print(i)
    i = i + 1    # we need to increase the value of i
```



```
0
1
2
3
4
>>>
```

An Infinite Loop

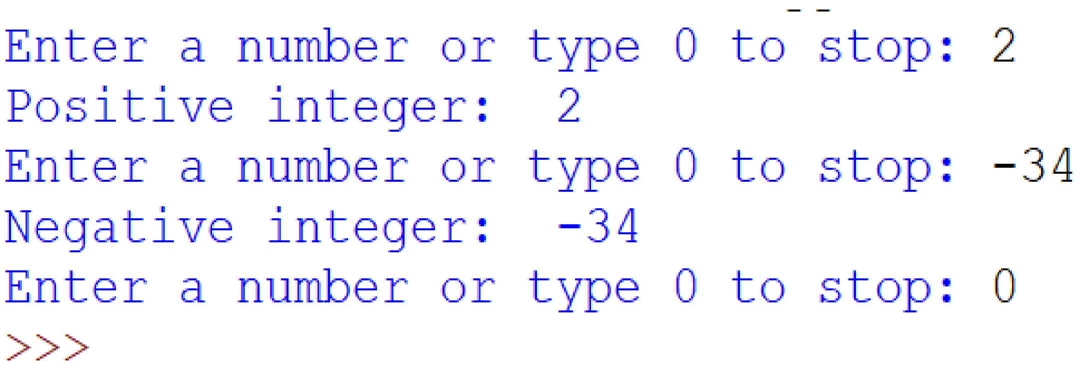
```
test = True

while test:
    print("This is an endless loop.")
    print("It will indefinitely print...")
    print("to terminate your program: - ", end="")
    print("press Ctrl-C, so called KeyboardInterrupt.")
```



```
This is an endless loop.
It will indefinitely pri
to terminate your progra
This is an endless loop.
It will indefinitely pri
to terminate your progra
Traceback (most recent c
  File "C:/Users/Savo/Ap
 7, in <module>
    print("press Ctrl-C,
KeyboardInterrupt
>>>
```

Activity 1: User Input and while Loop

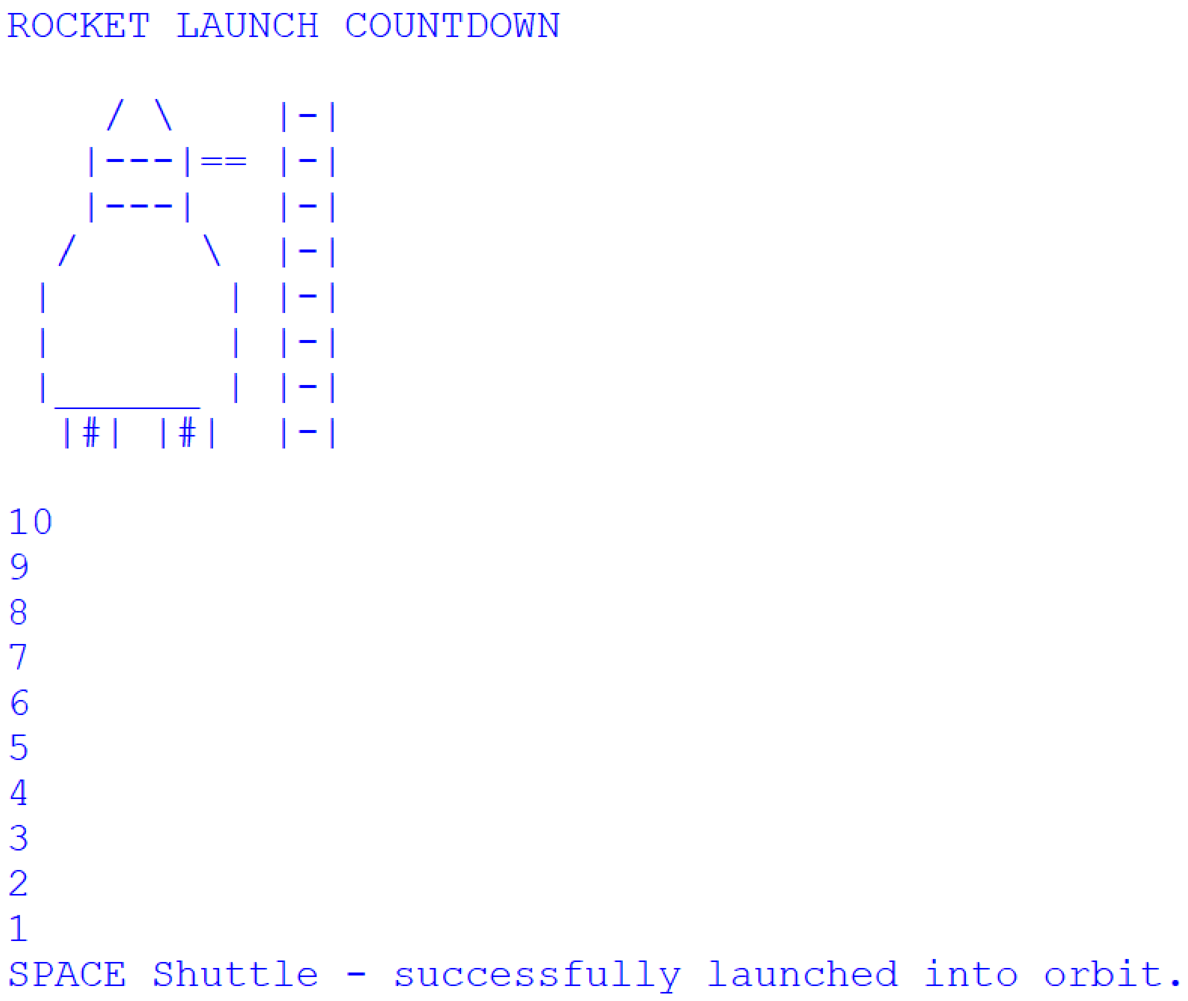
```
# Input the first value.
number = int(input("Enter a number or type 0 to stop: "))

while number != 0:
    # Is number larger than 0?
    if number > 0:
        print("Positive integer: ", number)
    else:
        print("Negative integer: ", number)
    # Input the next number.
    number = int(input("Enter a number or type 0 to stop: "))
```



```
Enter a number or type 0 to stop: 2
Positive integer:  2
Enter a number or type 0 to stop: -34
Negative integer:  -34
Enter a number or type 0 to stop: 0
>>>
```

Activity 2: Using a Counter to Exit Loop

```
import time    # needs to make a time delay
print("ROCKET LAUNCH COUNTDOWN")
print("""
   / \   |-|
  |---|== |-|
  |---|   |-|
 /     \ |-|
 |       | |-|
 |       | |-|v
 |______ | |-|
  |#| |#|  |-|
""")


counter = 10
while counter != 0:
    print(counter)
    counter = counter - 1
    time.sleep(1) # delay 1 second
print("SPACE Shuttle - successfully launched into orbit.")
```

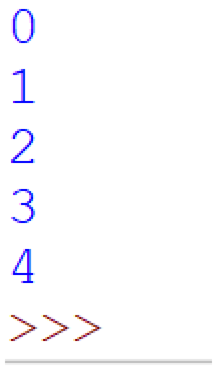


```
ROCKET LAUNCH COUNTDOWN

    / \    |-|
   |---|== |-|
   |---|   |-|
  /     \  |-|
 |       | |-|
 |       | |-|
 |_______| |-|
  |#| |#|  |-|

10
9
8
7
6
5
4
3
2
1
SPACE Shuttle - successfully launched into orbit.
```

The break statement

- The *break* statement terminate the current loop even if the while condition is true.
- For example:

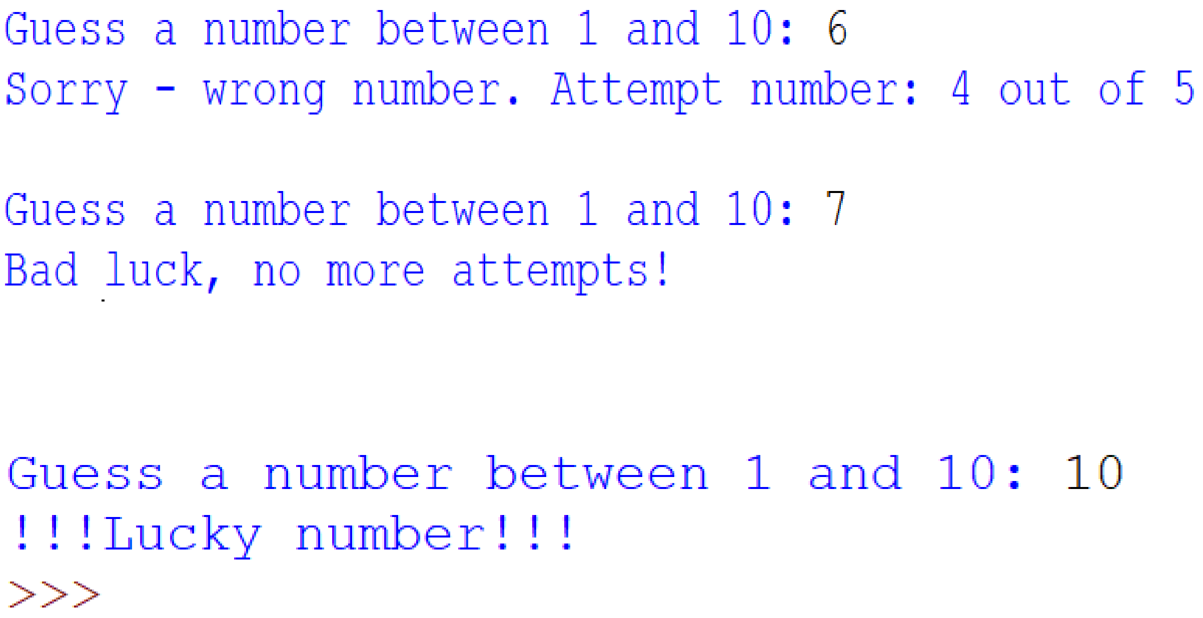
```
i = 0
while i < 20:
    print(i)
    i = i + 1
    if i == 5:
        break
```



```
0
1
2
3
4
>>>
```

Activity 3: Number Guessing Program (fix errors)

```
import random
print("Welcome to number guessing program.")
max_attempts = 5
lucky_number = random.randint(1, 10)  # random number
number = int(input("\nGuess a number between 1 and 10: "))
i = 0
while number != lucky_number:
    i = i + 1
    if i >= max_attempts
        print("Bad luck, no more attempts!")
        break
    else:
        print("Sorry - wrong number. Attempt number:", i, "out of 5")
        number = int(input("\nGuess a number between 1 and 10: "))
if i < max_attempts:
    print("!!!Lucky number!!!")
```

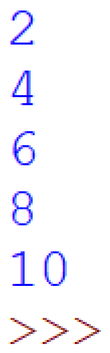


```
Guess a number between 1 and 10: 6
Sorry - wrong number. Attempt number: 4 out of 5

Guess a number between 1 and 10: 7
Bad luck, no more attempts!


Guess a number between 1 and 10: 10
!!!Lucky number!!!
>>>
```

The continue statement

- Used to stop the current iteration and continue with next
- The condition expression (while i < 10) is tested immediately

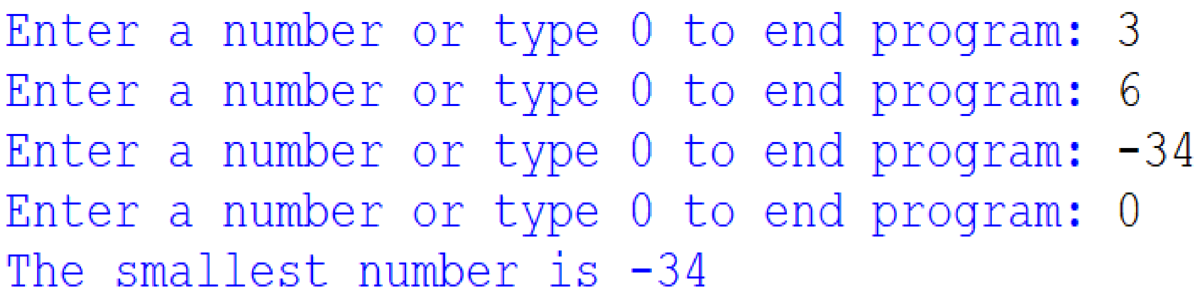
```
i = 0
while i < 10:
    i = i + 1
    if i % 2 == 1:
        continue
    print(i)
```



```
2
4
6
8
10
>>>
```

Activity 4: Find the smallest among the entered numbers

```
import sys
smallest_number = sys.maxsize
counter = 0
number = int(input("Enter a number or type 0 to end program: "))

while number != 0:
    if number == 0:  # terminate loop by checking condition
        continue
    counter += 1
    if number < smallest_number:
        smallest_number = number
    number = int(input("Enter a number or type 0 to end program: "))

if counter:
    print("The smallest number is", smallest_number)
else:
    print("Program ended.")
```

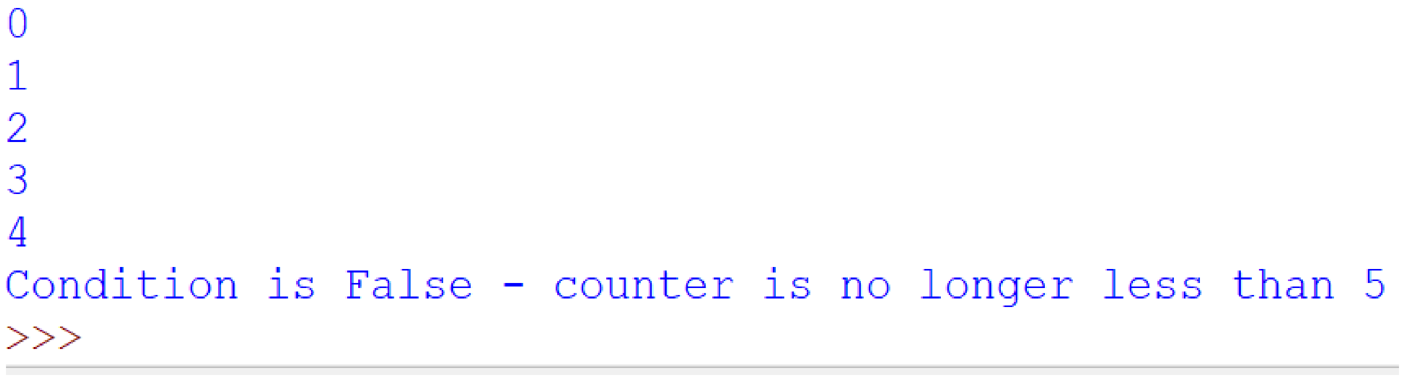


```
Enter a number or type 0 to end program: 3
Enter a number or type 0 to end program: 6
Enter a number or type 0 to end program: -34
Enter a number or type 0 to end program: 0
The smallest number is -34
```

The while Loop and the else branch

- Loops may have the else branch
- The else branch can run a block of code when the condition is False

```
counter = 0
while counter < 5:
  print(counter)
  counter = counter + 1
else:
  print("Condition is False - counter is no longer less than 5")
```



```
0
1
2
3
4
Condition is False - counter is no longer less than 5
>>>
```

Questions?