Flow Control (for Loop)

while Loop vs. for Loop

- while Loop continues while its **condition is True**
- for Loop executes the block of code a **certain number of times**. You know when to start and when to stop.

for Loop:

- Use to **iterate** through a sequence (like a “string”, numbers, [lists], etc.)
- **Iteration** – is a repeat of an action a predefined number of times.

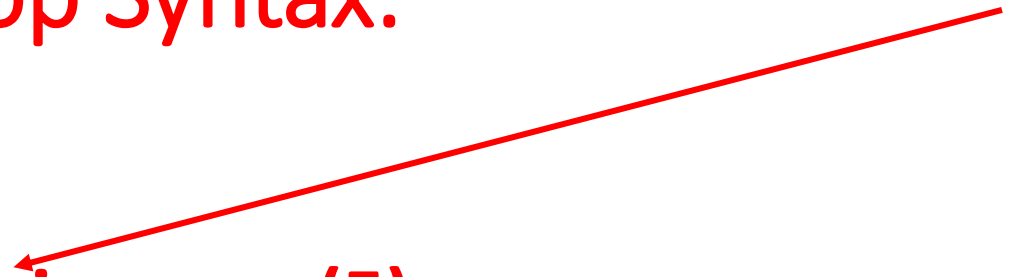
for Loop

The FOR statement will execute the block a certain number of times.

The FOR Statement consists of the following:

- The **for** keyword
- A **iteration variable** name
- The **in** keyword
- A call to the **range()** method, with up to three integers passed to it.
- A **colon**
- Starting on the next line, the for clause (**indented** block of code) will be executed.

for Loop Syntax:



#num is the iteration variable

```
for num in range(5):
        body of for loop
        to be repeated
```

#range() method called

#range(stop)

range( ) method – takes up to three arguments

range(start, stop[, step])

- **start** - integer starting from which the sequence of integers is to be returned
- **stop** - integer before which the sequence of integers is to be returned. The range of integers end at **stop - 1**.
- **step (Optional)** - integer value which determines the increment between each integer in the sequence, can be a positive or negative number

range( ) method – takes up to three arguments

#range(stop)

```
for num in range(5):
        body of for loop
        to be repeated
```

range( ) method – takes up to three arguments

#range(start, stop)

```
for num in range(2,5):
        body of for loop
        to be repeated
```

range( ) method – takes up to three arguments

#range(start, stop [,step])

```
for num in range(2,10,2):
        body of for loop
        to be repeated
```

Enter Code in: today's_date.py

Iteration variable num starts at 0 up to **NOT including 5.**

A call to the **range**() method.

```
for num in range(5):
    print(num)
```

num

OUTPUT:

0

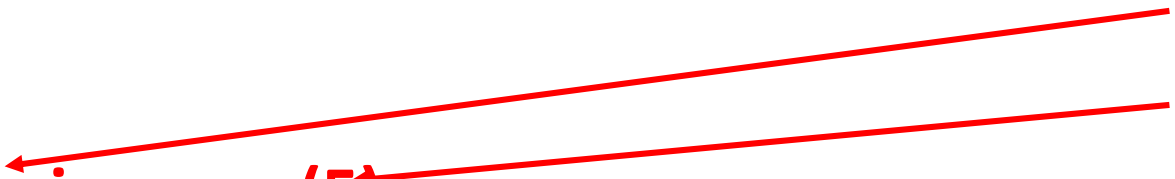
1

2

3

4

for Loops and range method Syntax



num starts at 0 up to not including 5.

A call to the **range**() method.

```
for num in range(5):
    print(num, end = " ")
```

#Keeps output on same line

OUTPUT:

0 1 2 3 4

for Loops Syntax

```
for num in 5:
        print(num, end = " ")
```

#Error 'int' object is not iterable.
#Iteration is a group of something
#(string or number) to go one after
#another.

for Loops Syntax (start and stop values)

Iteration variable num starts at 12 up to **NOT including 16.**

A call to the **range**() method, can have up to **three integers** passed to it.

```
for num in range(12, 16):
        print(num, end = " ")

#Num starts at 12, stops at 16 (not including stop value).
```

OUTPUT:

12 13 14 15

for Loops Syntax (start, stop and step values)

Iteration variable num starts at 0 up to **NOT including 10.**

A call to the **range**() method, can have up to three integers passed to it.

```
for num in range(0, 10, 2):
        print(num, end = " ")

#Num starts at 0, stops at 10
#(not including stop value), steps up by 2.
```

OUTPUT:

0 2 4 6 8

num

for Loops Syntax (range negative values)

```
for num in range(5, -1, -1):
        print(num, end = " ")


#Num starts at 5, stops at -1
#(not including stop value), steps down by -1.
```

OUTPUT:

5 4 3 2 1 0

num

for Loops and range values are exactly what you want.

- FOR Statement
  - When range is exactly what you want,
  - range keyword is not needed
  - **(Values can be numbers or strings)**
  - **[Insert lists in brackets]**

for Loops and range values are exactly what you want.

#range keyword is not needed

```
for steps in (11, 13, 22, 54):
        print(steps)
```

OUTPUT:

11

13

22

54

#steps is the iteration variable used to go through each number one by one

for Loops and range values are exactly what you want.

#range keyword is not needed

```
for steps in (11, 13, 22, 54):
        print(steps, end = “”)
```

OUTPUT:

11 13 22 54

#steps is the iteration variable used to go through each number one by one

#range keyword is not needed

```
for character in “banana”
        print(character)
```

OUTPUT:

b

a

n

a

n

a

#character is the iteration variable to go through the string character by character

#range keyword is not needed

```
for character in “banana”
        print(character, end=“ “)
```

OUTPUT:

b a n a n a

#character is the iteration variable to go through the string character by character

break and continue

- Can also be used in for Loops.