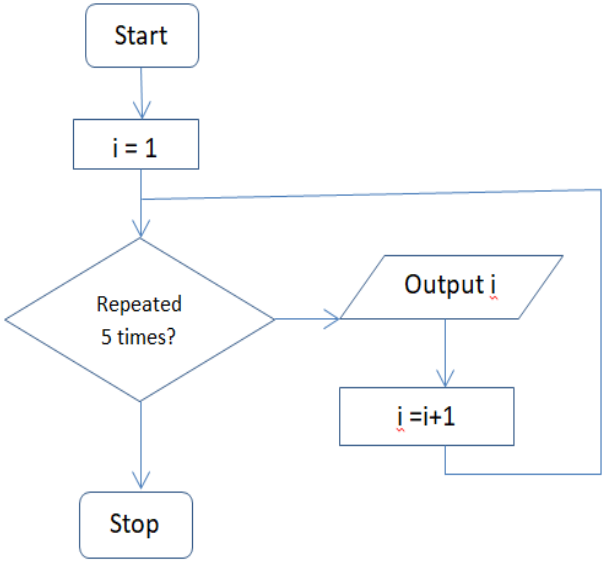
Repetition

For Loops

- A `for loop` allows Python to keep repeating the code a set number of times.
- A `for loop` is also known as `counting loop`.
- A for loop executes once *`for`* each item in the `collection`.
- The `collection` can be a range of integers, the letters in a string or values stored in a data structure such as list.
- Format of for loop:

  for `<variable>` in `<collection>`

  `<body>`




```
for i in range(1,5):
    print(i)
```

This loop uses a variable called "`i`" to keep track of times the loop has been repeated. It will start `i` at `1` and repeating the loop, adding `1` to `i` each time and displaying the value of `i` until it reaches `5`, where it will stop. It will not repeat the loop the fifth time and only has the following output: `1, 2, 3, 4`

For Loops: Examples

```
#read the limit from the user
limit = int(input("Enter an integer: "))

#Display the positive multiples of 3 up to the limit
print("The multiples of 3 up to and including", limit, "are: ")
for i in range (3, limit + 1, 3):
    print(i)
```

```
================== RESTART: C:/Users/mireilla/s
Enter an integer: 17
The multiples of 3 up to and including 17 are:
3
6
9
12
15
```

```
for i in range(1, 10, 2):
    print(i)
```

This range function includes a third value which shows how much is added to i in each loop (in this case, 2). The output will be: 1, 3, 5, 7, 9

```
for i in word:
    print(i)
```

This will display each character in a string called "word" as a separate output(e.g. on a separate line)

```
for i in range (20, 1, -4):
    print(i)
```

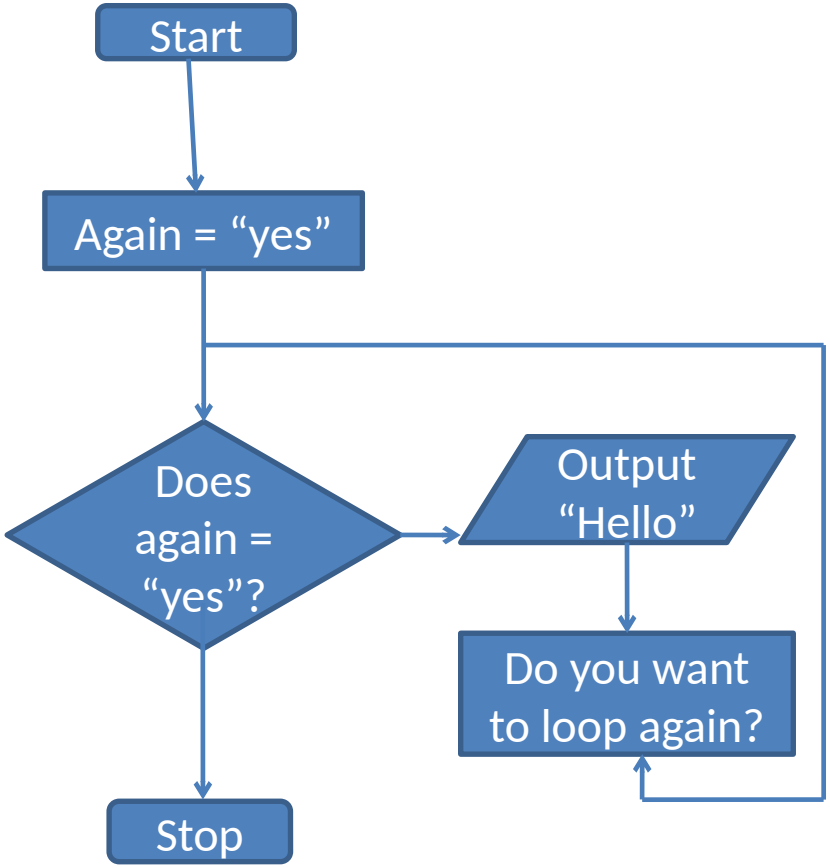
This range will subtract 4 from i each time. The output will be: 20, 16, 12, 8, 4

While Loops

- A `while loop` causes one or more statements to execute as long as, or while, a condition is met (evaluates to `True`).

- In a `while loop`, the condition is checked before the code is run which means it could skip the loop altogether if the condition is not being met at the beginning.

- It is important, to make sure the correct conditions are in place to run the loop before it starts.




```
============ RESTART: C:/Users/
Hello
Do you want to loop again?yes
Hello
Do you want to loop again?yes
Hello
Do you want to loop again? no
>>>
```

```
again = "yes"
while again == "yes":
    print ("Hello")
    again=input("Do you want to loop again?")
```

It will keep repeating this code until the user enters anything other than yes

While Loops - example

Example 1:

This programme will create a variable called total and store the value as 0. It will ask the user to enter a number and will add it to the total. It will keep repeating this as long as the total is still below 100. When the total equals 100 or more, it will stop running the loop and display the total.

```
total = 0
while total < 100:
    number = int(input("Enter a number: "))
    total = total + number
print ("Hello, the total is: ", total)
```

```
Enter a number: 23
Enter a number: 45
Enter a number: 12
Enter a number: 56
Hello, the total is:  136
>>>
```

```
File  Edit  Shell  Debug  Options  Window  He
Python 3.7.2 (tags/v3.7.2:9a3ffc0492
(Intel)] on win32
Type "help", "copyright", "credits"
>>>
========== RESTART: C:/Users/mire
Enter an integer (0 to quit): 2
That's a positive number.
Enter an integer (0 to quit): 0
>>>
========== RESTART: C:/Users/mire
Enter an integer (0 to quit): -2
That's a negative number.
Enter an integer (0 to quit):
```

File Edit Format Run Options Window Help

```
#Read the first value from the user
x = int(input("Enter an integer (0 to quit): "))

#Keep looping while the user enters a non-zero number
while x != 0:
    #Report the nature of the number
    if x > 0:
        print("That's a positive number.")
    else:

        print("That's a negative number.")
    break   # If you don't add this break, it will run indefinitely

#Read the next value from the user
x = int(input("Enter an integer (0 to quit): "))
```

Example 2: Self – explanatory comments

Nested Loops

- The statements inside the body of a loop can include another loop.
- This is known as `nested loop`

In this example of `nested loop`, we have a `for loop` inside a `while loop`.

```
#Read the first message from the user
message = input("Enter a message (blank to quit): ")

#Loop until the message is a blank line
while message != "":
    #read the number of times the message should be displayed
    n = int(input("How many times should it be reapted? "))

    #Display the message the number of time requested
    for i in range(n):
        print(message)

    #Read the next message from the user
    message = input("Enter a message (blank to quit): ")
```

```
Enter a message (blank to quit): Hello everyone, how are you today? This is an exam
ple
How many times should it be reapted? 5
Hello everyone, how are you today? This is an example
Hello everyone, how are you today? This is an example
Hello everyone, how are you today? This is an example
Hello everyone, how are you today? This is an example
Hello everyone, how are you today? This is an example
Enter a message (blank to quit):
>>>
```