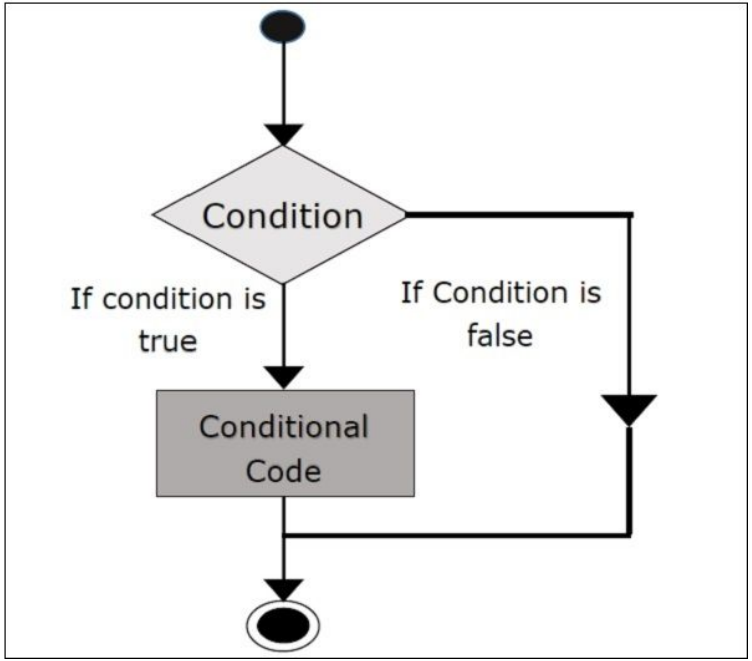
Flow Control & Loops

Creating Intelligent Programs

The “*if*” statement

This is a conditional flow statement which uses a binary boolean input.



If statement syntax

if(condition):

    Conditional code

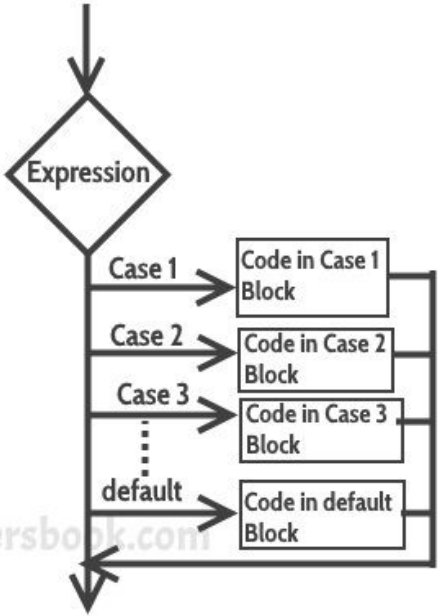
How to use conditions:

A condition is the statement which the if statement checks. These conditions can be very simple or very complex. The conditions are a **Boolean** statement.

Don’t forget to TAB

The “*switch*” statement

This is a multiple I/O if-type statement. Best used for menus.



Switch Statement Syntax

```
switch(variable_to_check) {
        Case some_value:
                Conditional Code
                break;
        Case some_value:
                Conditional Code
                Break;
        Case default:
                Conditional Code
}
```

Exercise 1: Don’t divide with Zero

Level: Easy

Create a program that takes two numbers. Check that the denominator is **not** equal to zero. If it is equal to zero don’t run the division.

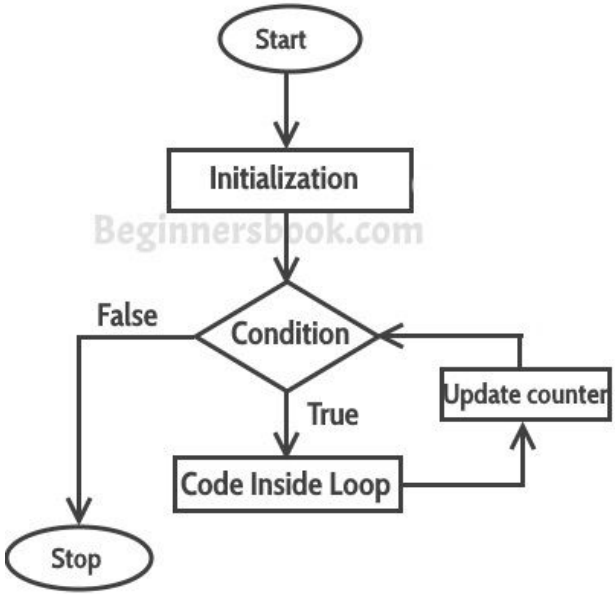
Extra points if you use the *right* conditional.

Allowed Time: 5 Minutes

Loops

The “*for*” Loop

This is the most common loop and basically iterates through a structure until it reaches the end.



How *‘for’* loops work in other languages

For (i=0,i<10,i++){

code;

}

This is in Java. But what does it say?
The iterative variable ‘i’ starts at 0 and continues until i<10 is false, at each run i = i +1 (this is the i++).

In Py it is confusing.

For Loop Syntax

*For i in **SOME STRUCTURE:***

    CONDITIONAL CODE

Example:

```
For i in range(0,10):
    print(i)
```

What will this print?

Iterative Structures

Looping a number of times using range:

range(number1,numberN)

Using range as the structure we iterate from number1 to numberN

We can loop through different things. We can automatically iterate arrays, files, dicts etc. This is not the case in other languages!

What is Range?

Range is a list of numbers. This means that instead of range(m,n) we can use a **custom** array or list!.

This is significantly harder to do in other languages.

More of this next week.

Exercise 2: Addition

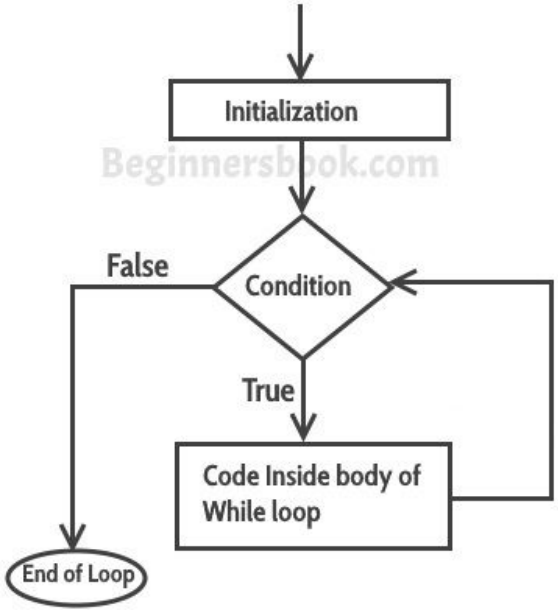
Level: Easy

Find the sum of the arithmetic sequence S = [0,1,2,...,99,100] using a *for loop* and output the sum to the user.

Allowed Time: 5 Minutes

“*While*” Loop

The while loop is a conditional statement. It performs a task “while” the condition is true.



While Loop Syntax

While ***condition:***

    CONDITIONAL CODE

Example:

While 1 > 0:

    print(“Hello World”)

What’s the Difference

- For loops are extended While Loops
- For loops in Python are designed to iterate inside structures (array’s, lists, strings etc.)
- While loops are simplistic

Designing Intelligent Programs

Problem 1: Fibonacci Sequence

The Fibonacci Sequence is an arithmetic sequence such that: “*each term is the sum of the two previous terms*”.

We want to create a program that outputs a given number of Fibonacci numbers after the user inputs how many terms of the sequence they want to see.

Let’s design it!

What cases do we have?

Input <= 0 then we can’t run

Input == 1 then we need to return 0

Input > 1 then we need to calculate

How do we calculate?

We need a *loop* & the user input.
We need to make sure that loop runs up to the user input.
Then we need to add the terms of the sequence. How do we do this?

1. Declare Term1 & Term2 → (0,1)
2. Add these on the first loop iteration
3. Change the values and print on each iteration

Changing the Values

N3 (which is n1+n2)*	n1	n2
null	0	1
1	1	1
2	1	1
3	1	2
5	2	3

n1 and n2 are declared as 0 & 1 by the coder. At each iteration we add and change.
N3 is the one we print since it is the sum of n1+n2.

Step by step

N3 is n1+n2
n1 is n2
n2 is n3

In Code:

n3 = n1 + n2
n1 = n2
n2 = n3

The calc. loop in Python

```
While count < nterms

    n3 = n1 + n2

    print(n3)
    n1 = n2
    n2 = n3
    count++
```

```
For i in range(0,nterms)
    print(n1)
    n3 = n1 + n2
    n1 = n2
    n2 = n3
```

The full code is in the Drive

The ‘else’ statement

What if the condition is not True? Then we just use else.
We can use else in If, while and, for.

In the for loop:

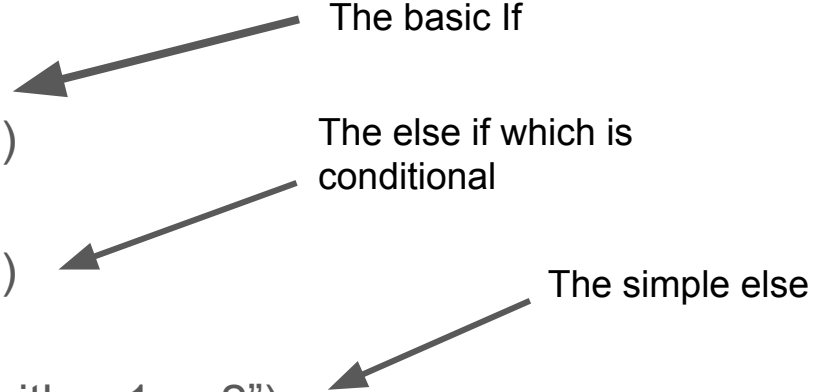
```
For i in range(0,10):
        print(i)
else:
        print(“the loop is over :(“)
```

In the If statement

In the if statement we use: if, else or, else if to work with cases (a basic switch)

Example:

```
If(a = 1):
      Print(“a = 1”)
elif(a = 2):
      Print(“a = 2”)
else:
      Print(“a is neither 1 or 2”)
```

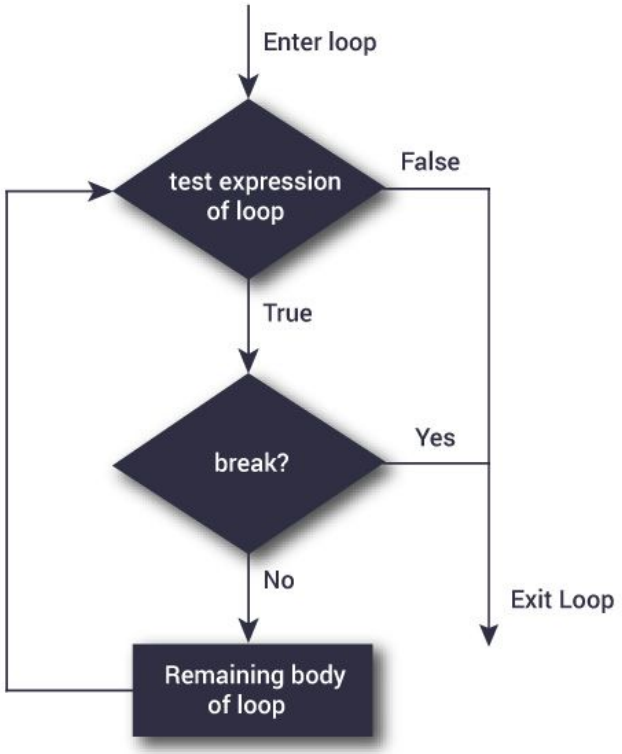


The ‘break’ statement

Do we need to exit the loop early? **Break!**

```
For i in range(0,10):
        print(i)
        if(i == 3):
                break
```

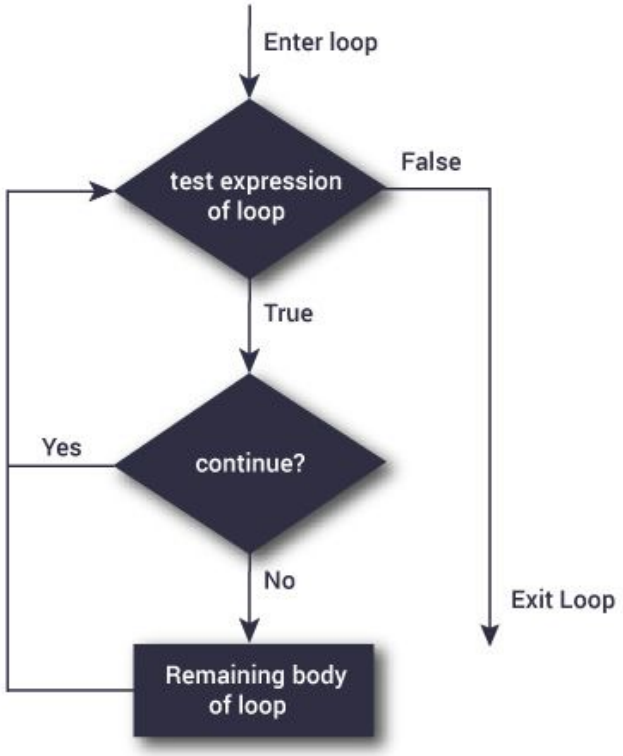
This can be used for Optimization.



The ‘continue’ statement

Do we want to skip the conditional code? Then continue!

```
For i in range(0,10):
        if(i == 2):
                continue
        print(i)
```



Exercise 3: Search for a number

Given the array defined below (just copy it in the script as is) find the a number which the user inputs. Have an output for whether it exists or not and break early if you find it.

The array: **array1 = [1,3,11,0,2,92,420,8,34,69]**
To access an element in the array type: **array1[*index*]**
Index is a number which corresponds to the position in the array (i.e. array1[0] = 1)

The array index starts at 0

Allowed Time: 10 Minutes.

Exercise 3: Optimization Hints

Use boolean statements and a combination of if & else statements to make it.

You should use a *flag* technique which means that if you find the element in the array you should change a flag variable to True from False.