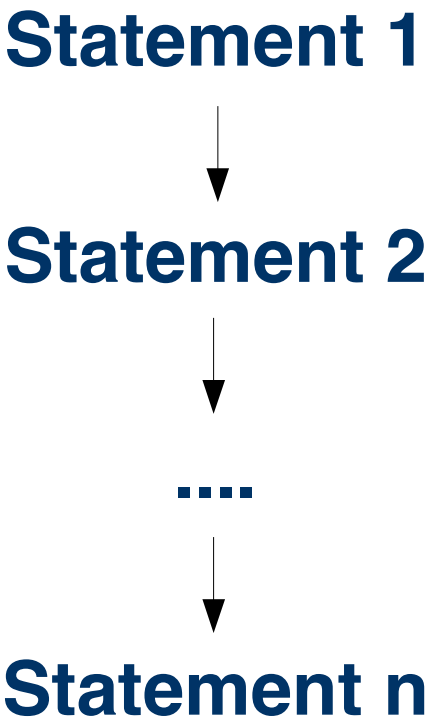
Flow control
Conditional statements

Sequential flow

Python read and execute each statement
in the script file **sequentially = line by line**
Until the end of the file



if statements

- Often to solve a problem one may or must control the flow
- Check if certain conditions are met

if statements

- Often to solve a problem one may or must control the flow
- Check if certain conditions are met

Example:

```
x=2
y=0
z=0
z=x/y
print ('z=',z)
```

if statements

- Often to solve a problem one may or must control the flow
- Check if certain conditions are met

Example:

```
x=2
y=0
z=0
z=x/y
print ('z=',z)
```

ERROR:
ZeroDivisionError

if statements

```
x=2
y=0
z=0
z=x/y
print ('z=',z)
```

To avoid this error we have to check:
if **y** is not equal to 0
before to execute the statement: **z=x/y**

if statement constructs - general syntax

```
if condition is True:
        Statement 1
        …
        Statement N
```

if statement constructs - general syntax

```
if condition is True:
        Statement 1
        …
        Statement N
```

The **if** block

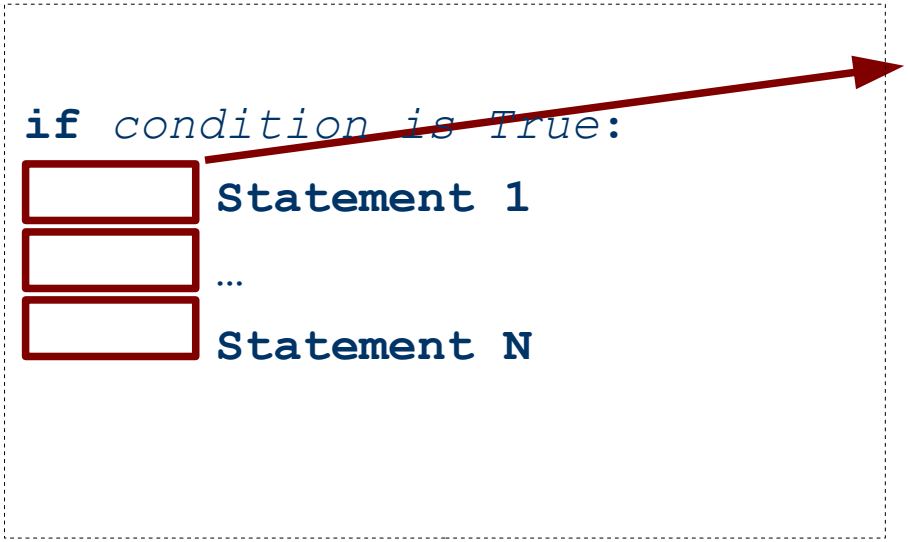
if statement, condition is evaluated first.

If condition is true (if its value is nonzero) then the statement(s) block are executed. Otherwise, the next statement following the statement(s) block is executed.

if statement constructs - general syntax

```
if condition is True:
        Statement 1
        …
        Statement N
```

if statement constructs - general syntax



Indentation
Could be space/tab

At least 1 space or 1 tab

Every **Statement** in the block
Must start at the same column

All the statements indented by the same number of character spaces/tab after a programming construct are considered to be part of a single block of code.

Python uses indentation as its method of grouping statements.

How to evaluate if it is *True* or *False*

Operators that return Boolean values:

== equal

!= not-equal

< less than

<= less than or equal

> greater than

>= greater than or equal

Example:

a == b

a < b

Operators - Logical Operations

The classic Boolean operators are **not**, **and**, and **or**.

Can be combined with the comparison operators

Logical Operators - Example

```
x = 4
y = 3
```

The expression: **x < 8** gives **True**

The expression: **y != 3** gives **False**

The expression **x<8 and y==3** gives **True**

(True **and** True)

Truth Table for Basic Logical Operators

Condition A	Condition B	A or B	A and B	not A
True	True	True	True	False
True	False	True	False	False
False	True	True	False	True
False	False	False	False	True

if statements

```
x=2
y=0
z=0
z=x/y
print ('z=',z)
```

To avoid this error we have to check:

if **y** is not equal to 0
before to execute the statement: **z=x/y**

```
x=2
y=0
z=0
If y != 0:
        z=x/y
print('z=',z)
```

Gives:

```
z=0
```

if - else statements

else statement can be used with ***if*** optionally.

If condition specified in ***if*** statement is False statement(s) after ***else*** will be executed

```
if condition is True:
        Statement 1
        Statement 2
else:
        Statement 3
        Statement 4
```

if - else statements

```
x=2
y=0
z=0
if y != 0:
      z=x/y
else:
      print ('div by 0 not possible')
Print('z=',z)
```

Gives:

div by 0 not possible

if - elif - else statements

The ***elif*** statement stand for “else if”

elif Allows you to check multiple expressions for truth value and execute a block of code as soon as one of the conditions evaluates to true.

Like the ***else***, the ***elif*** statement is optional.

However, unlike ***else***, for which there can be at most one statement, there can be an arbitrary number of ***elif*** statements following an ***if***.

if - elif - else statements

```
if condition1 is True:
        Statement 1
        Statement 2
elif condition2 is True:
        Statement 3
        Statement 4
elif condition3 is True:
        Statement 5
        Statement 6
else:
        Statement 7
```

Some notes about the code

Only one of the condition ***if*** or ***elif*** will be met and hence **only one** of the circumstances will be executes

If **none** of the conditions are met then the ***else*** circumstance is executed.

if - elif - else statements

```
x=10
if x>30:
        print ('x is big')
elif x==10:
        print ('x equal to 10')
elif x<=0:
        print ('x is 0 or negative')
else:
        print ('other')
print ('bye')
```

Gives:

```
x equal to 10
bye
```

if - elif - else statements

```
if x>30:
        print ('x is big')
elif x==10:
        print ('x equal to 10')
elif x<=0:
        print ('x is 0 or negative')
else:
        print ('other')
print ('bye')
```

x = 35

x = 7

x = -15

What is the result in each case?

if within an *if* – Nested *ifs*

In some cases if may be necessary to have an if statement nested within another:

```
if condition1 is True:
        Statement 1
        if condition2 is True:
                Statement 2
```

if within an *if* – Nested *ifs*

```
x = 100
if x < 200:
   print ('x value is less than 200')
   if x == 100:
      print ('Which is 100')
   elif x == 50:
      print ('Which is 50')
elif x < 50:
   print ('x value is less than 50')
else:
   print ('Could not find true expression')
print ('bye')
```

if within an *if* – Nested *ifs*

```
x = 100
if x < 200:
   print ('x value is less than 200')
   if x == 100:
      print ('Which is 100')
   elif x == 50:
      print ('Which is 50')
elif x < 50:
   print ('x value is less than 50')
else:
   print ('Could not find true expression')
print ('bye')
```

if within an *if* – Nested *ifs*

```
x = 100
if x < 200:
    print ('x value is less than 200')
    if x == 100:
        print ('Which is 100')
    elif x == 50:
        print ('Which is 50')
elif x < 50:
    print ('x value is less than 50')
else:
    print ('Could not find true expression')
print ('bye')
```

The nested *if* block

Gives:

```
x value is less than 200
Which is 100
bye!
```

if statements characters based (string)

All logical operators can be used BUT we will use only:

'=='	2 strings are equal (identical character by character)
'!='	2 strings are not equal

Two additional operators :

In	string is substring of another one
not in	string is not a substring of another one

if statements characters based (string)

```
str1='Monty Python spam script'
str2='spam'

If str1 == str2:
       Print 'str1 equal str2'
if str2 in str1:
       print (str2+' is in the string:'+str1)
else:
       print ('is not..')
print ('bye!')
```

Gives:

```
spam is in the string:Monty Python spam script
bye!
```