Conditional Statements, For loops, Range, Exceptions

creative thinking with programming day 2

Boolean Expressions

Python's Boolean type is true or false

There are only 2 values, True and False

```
x = 5
```

Sets x equal to 5

```
if x == True
```

A boolean expression evaluates what's on the right as either True or False and assigns it to what's on the left

Boolean Expressions ask a question and produce a yes or no result we use to control how a program flows

Boolean Expressions use comparison operators to evaluate if something is true or false, yes or no.

Comparison operators look at variables but do not change variables

Conditional Statement

Evaluates the Boolean expression

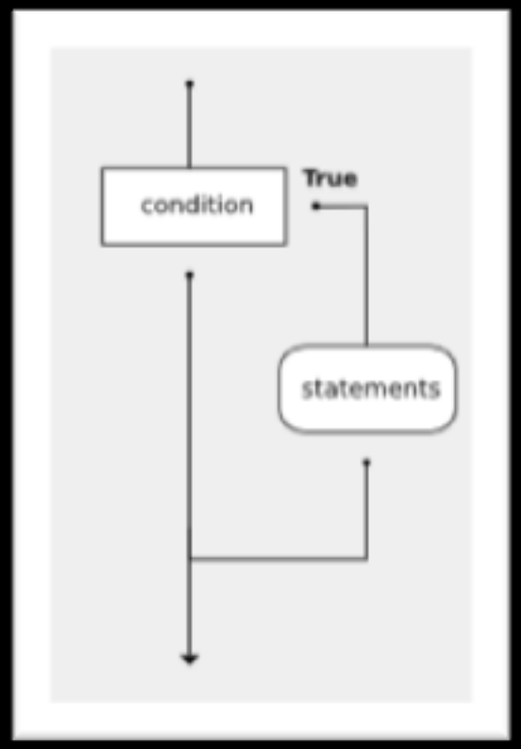
Gives the ability to check conditions and change the behavior of the program accordingly

The simplest form of selection is the **if statement**.
You can think of it like if this then do that or do something else.

If statements part 1

If statements do something if a condition is true or false. This is a Unary Selection.

```
if x==10:
  x++
```



Indentation

Indentation matters in Python. If you have a statement that ends in a : you can add lines of code underneath it tabbed in with one tab and those lines of code are grouped within the **scope** of that condition

When you un-indent you leave this **scope** and are back *outside* the conditional block

Indentation Example

```
x = 5
y = 7
if x==5 or y==6:
    x = 10
    y = 10
    print “I ran!”
```

More Boolean Expressions

```
x != y
# x is not equal to y

x > y
# x is greater than y

x < y
# x is less than y

x >= y
# x is greater than or equal to y

x <= y
# x is less than or equal to
```

Logical Operators

and

```
x = True
y = 6
if x==True and y == 6:
    print "both conditions are true"
```

Logical Operators

or

```
x = 5
y = 7
if x==5 or y==6:
    print "one condition is true"
```

Logical Operators

not

```
x = 12
y = 7
if !x==5 or y==6:
    print “x is not 5 or y is 6”
```

Logical Operators

other ways to do not

```
x = 12
y = 7
if not x==5 or y==6:
    print “x is not 5 or y is 6”
```

Order of logical operators

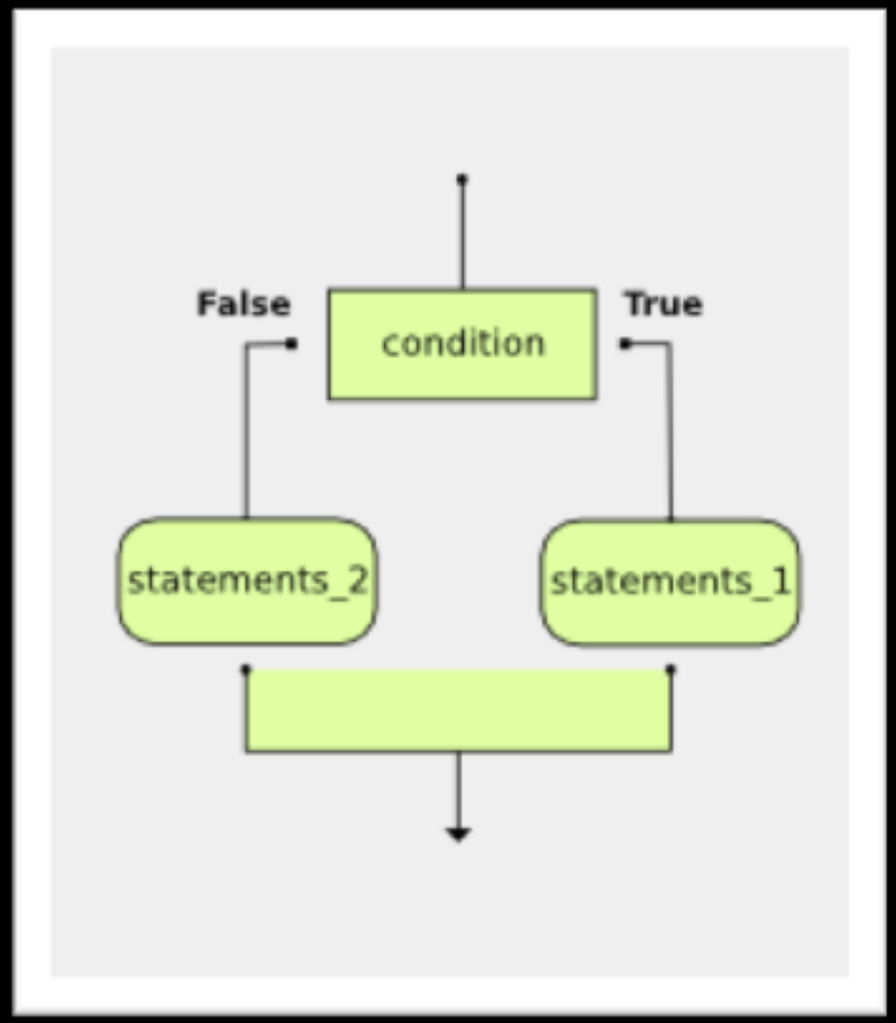
Python will always evaluate the arithmetic operators first (** is highest, then multiplication/division, then addition/subtraction). Next comes the relational operators. Finally, the logical operators are done last.

Level	Category	Operators
7(high)	exponent	**
6	multiplication	*,/,//,%
5	addition	+,-
4	relational	==,!=,<=,>=,>,<
3	logical not	
2	logical and	
1(low)	logical or	

if statements can have second parts

```
x = 11
if x=10:
    STATEMENTS_1
else:
     STATEMENTS_2
```

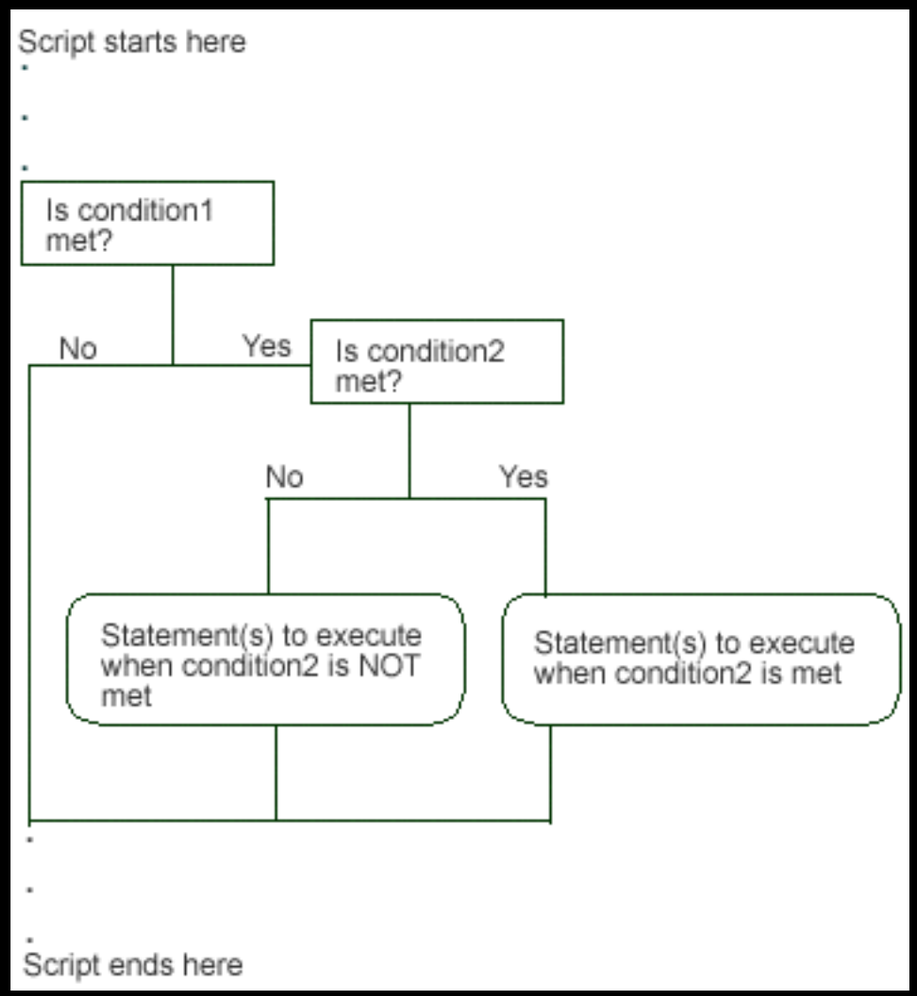
if statement one is true run the first set of statements, else run the statements in the second block



You can nest them inside each other

You can nest these things inside each other

```
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater
than y")
    else:
        print("x and y must
be equal")
```


Script starts here
Is condition1 met?
No
Yes
Is condition2 met?
No
Yes
Statement(s) to execute when condition2 is NOT met
Statement(s) to execute when condition2 is met
Script ends here

Chained Conditionals

Python provides an alternative way to write nested selection such as the one shown in the previous section. This is sometimes referred to as a **chained conditional**

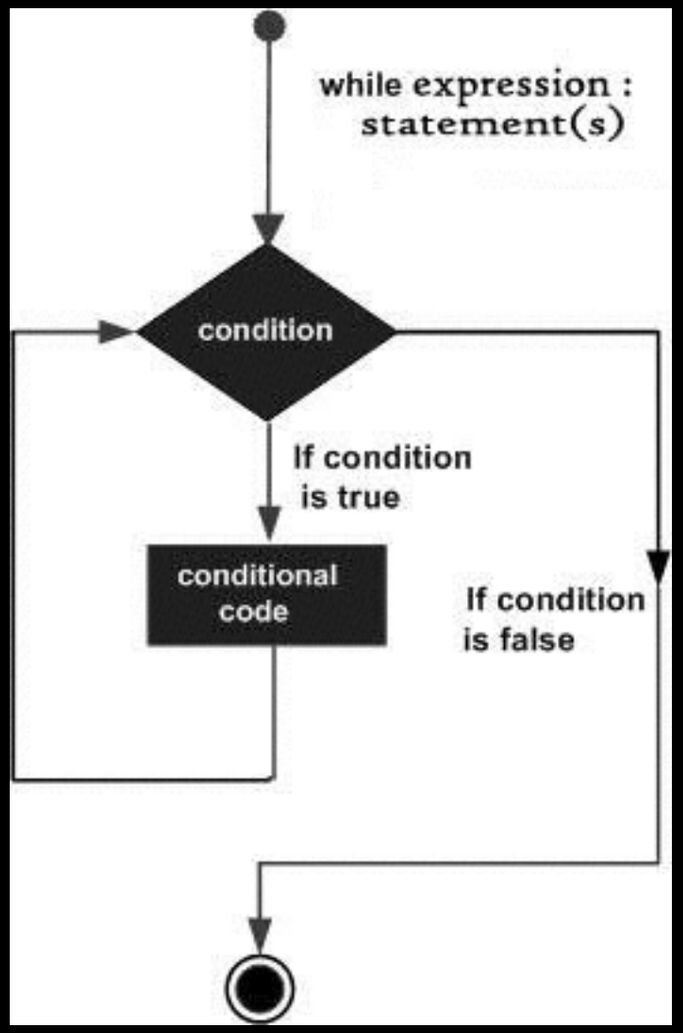
```
if x < y:
    print("x is less than y")
elif x > y:
        print("x greater than than y")
else:
    print("x and y are equal")
```

While loop

While some condition is true do something

```
while(x < 5):
  print “ x is “ + str(x)
  x++

print “this is it!”
```



range

Range does something as long as a condition is in the range. It does not include the last number in it.

this code:

gives i the value 3

runs
gives i the value 3
increments i

runs
gives i the value 4
increments i

runs
gives i the value 5
increments i

exits

```
for i in range (3,6):
    x+=20
```

Exit conditions

Note this and the for loop both have a condition that ends the loop. You want this. Infinite loops are usually undesirable

```
while(x < 5)
  print “ x is “ + str(x)
  x++

print “this is it!”
```

```
for i in range (3,6):
    x+=20
```

Quiz:

1. What is the exit condition in the while loop?

2. What is it in the for loop

try and except

Try and except is a way to make sure some input or file or something you need to be one kind of thing is that kind of thing.

For example, if you want something to be an int and it's not you can now do something about it!

```
while True:
      try:
          x = int(raw_input("Please enter a number: "))
          break
      except ValueError:
          print "Oops!  That was no valid number.  Try again..."
```

% the modulus operator

divides two numbers and give you the remainder. Really good at ranges because it will alway give you 0 when you hit a multiple in a range of numbers.

```
print 10 % 3
#give you 1

print 1 % 2
# gives you 1

print 2 % 2
#gives you 0

print 3 % 2
#gives you 1
```