Python If ... Else

Python conditions and if statement:

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

If statement:

Example:

```
a = 33

b = 200

if b > a:

  print("b is greater than a")
```

In this example we use two variables, `a` and `b`, which are used as part of the if statement to test whether `b` is greater than `a`. As `a` is `33`, and `b` is `200`, we know that 200 is greater than 33, and so we print to screen that "b is greater than a".

Indentation:

Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

If statement, without indentation (will raise an error):

Example:

```
a = 33

b = 200
```

```
if b > a:

print("b is greater than a")
```

elseif (elif)

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

Example:

```
a = 33

b = 33

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")
```

In this example `a` is equal to `b`, so the first condition is not true, but the `elif` condition is true, so we print to screen that "a and b are equal".

else

The `else` keyword catches anything which isn't caught by the preceding conditions.

Example:

```
a = 200

b = 33

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")

else:

  print("a is greater than b")
```

In this example `a` is greater than `b`, so the first condition is not true, also the `elif` condition is not true, so we go to the `else` condition and print to screen that "a is greater than b".

You can also have an `else` without the `elif`:

```
a = 200

b = 33

if b > a:

  print("b is greater than a")

else:

  print("b is not greater than a")
```

Short Hand if

If you have only one statement to execute, you can put it on the same line as the if statement.

One line if statement:

Example:

```
a = 200

b = 33

if a > b: print("a is greater than b")
```

Short Hand if else

One line if else statement:

Example:

```
a = 2

b = 330

print("A") if a > b else print("B")
```

This technique is known as **Ternary Operators**, or **Conditional Expressions**.

You can also have multiple else statements on the same line:

Example:

```
a = 330

b = 330

print("A") if a > b else print("=") if a == b else print("B")
```

and

The `and` keyword is a logical operator, and is used to combine conditional statements:

```
a = 200

b = 33

c = 500

if a > b and c > a:

  print("Both conditions are True")
```

Or

The `or` keyword is a logical operator, and is used to combine conditional statements:

```
a = 200

b = 33

c = 500

if a > b or a > c:

  print("At least one of the conditions is True")
```

Test if `a` is greater than `b`, OR if `a` is greater than `c`:

Nested if

You can have `if` statements inside `if` statements, this is called *nested* `if` statements.

```
x = 41

if x > 10:

  print("Above ten,")

  if x > 20:

    print("and also above 20!")

  else:

    print("but not above 20.")
```

The pass statements

`if` statements cannot be empty, but if you for some reason have an `if` statement with no content, put in the `pass` statement to avoid getting an error.

```
a = 33

b = 200


if b > a:

  pass

# having an empty if statement like this, would raise an error without the pass statement
```

Python Challenges:

1) Print "Hello World" if a is greater than b.

```
a = 50
b = 10
[ ] a [ ] b[ ]
  print("Hello World")
```

2) Print "Hello World" if a is not equal to b.

```
a = 50
b = 10
[ ] a [ ] b[ ]
  print("Hello World")
```

3) Print "Yes" if a is equal to b, otherwise print "No".

```
a = 50
b = 10
[ ] a [ ] b[ ]
  print("Yes")
[ ]
  print("No")
```

4) Print "1" if a is equal to b, print "2" if a is greater than b, otherwise print "3".

```
a = 50
b = 10
[ ] a [ ] b[ ]
  print("1")
[ ] a [ ] b[ ]
  print("2")
[ ]
  print("3")
```

5) Print "Hello" if `a` is equal to `b`, and `c` is equal to `d`.

```
if a == b  c == d:
  print("Hello")
```

6) Print "Hello" if `a` is equal to `b`, or if `c` is equal to `d`.

```
if a == b  c == d:
  print("Hello")
```

7) This example misses indentations to be correct.

Insert the missing indentation to make the code correct:

```
if 5 > 2:
print("Five is greater than two!")
```

8) Use the correct short hand syntax to put the following statement on one line:

```
if 5 > 2:
print("Five is greater than two!")
```

9) Use the correct short hand syntax to write the following conditional expression in one line:

```
if 5 > 2:
  print("Yes")
else
  print("No")
```