Python Basics PT2

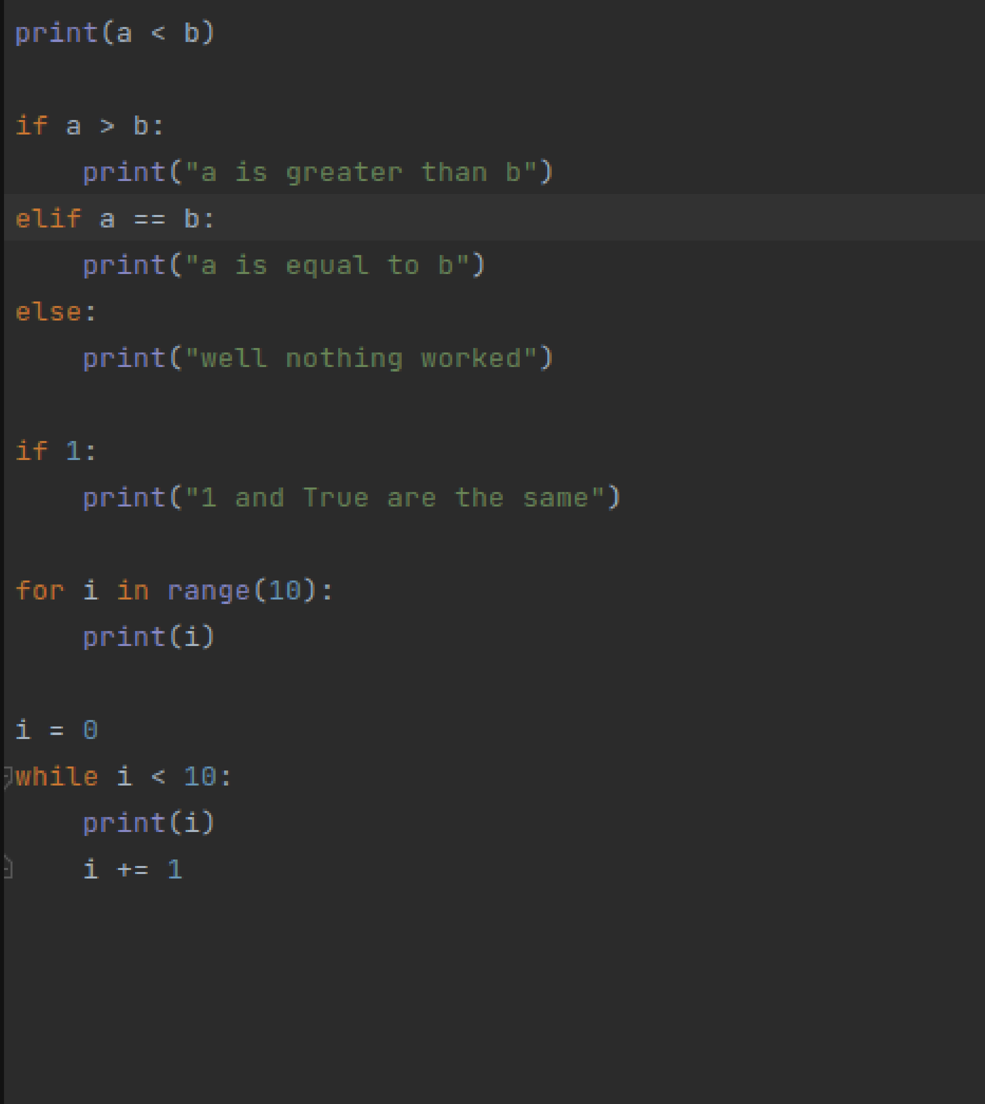
Research Data Services

By Avery Fernandez

Plans For Today

Learning the basics to python

- **Boolean Logic** how to compare variables
- **If/Else Statements** using conditionals
- **For Loops** looping through data
- **While Loops** another way of looping



```python
print(a < b)

if a > b:
    print("a is greater than b")
elif a == b:
    print("a is equal to b")
else:
    print("well nothing worked")

if 1:
    print("1 and True are the same")

for i in range(10):
    print(i)

i = 0
while i < 10:
    print(i)
    i += 1
```

Boolean Logic: What is it?

Boolean logic is how we compare variables and create logical statements

There are 5 main operators

```
<
<=
==
>=
>
```

They stand for, respectively:

- Less than
- Less than or equal to
- Equal to
- Greater than or equal to
- Greater than

Boolean Examples

Outputs

```
print( 10 > 4 )
```

```
True
```

```
print( 7 == 8 )
```

```
False
```

```
print( 7 != 8 )
```

```
True
```

```
print( 13 < 5 )
```

```
False
```

```
print( "apple" == "orange" )
```

```
False
```

```
print( "apple" == "apple" )
```

```
True
```

If Else Statements

You can use conditional statements to execute code using If and else statements

Syntax

```
if {statement}:
    run this if true
else:
    run this is previous statement came out false
```

Example

```python
user = input("What is your name? ")
if user == "Jose":
    print("Welcome home!")
else:
    print("This is not your home")
```

Terminal

```
What is your name? Jose
Welcome home!

What is your name? John
This is not your home
```

Logical Operators

We can combine logic into a single statement using logical operators

Logical operators are:

- or -> if either or both are true, then say true
- and -> if both are true, then say true

Examples

```
if True and True:
    print("Both True")
if True or False:
    print("At least one was true")
```

Output:

```
Both True
At least one was true
```

More If Else statements

We can make more complex logic statements using elif with if else statements

It will go down the list of statements and stopping when it finds one that is true

```python
user = int(input("Enter a number: "))
if user == 1:
    print("The first")
elif user == 2:
    print("The second")
elif user == 3:
    print("The third")
else:
    print("Not an option")
```

Output

```
Enter a number: 3
The third
Enter a number: 1
The first
Enter a number: 10
Not an option
```

Exercise One

Combine user inputs, variable casting, and if statements

Take two user inputs

Output the sum of the numbers if both numbers are the same

Output the product of the numbers if both numbers are greater than 10

Output the individual numbers if the previous statements were false

Exercise One Code

```python
user1 = int(input("Enter the first number: "))
user2 = int(input("Enter the second number: "))

if user1 == user2:
    print(user1 + user2)
elif user1 > 10 and user2 > 10:
    print(user1 * user2)
else:
    print(user1, user2)
```

For Loops

We can do repeated tasks by using loops

We can use the **range()** function

```python
for i in range(5): # runs 5 times
    print(i)
```

We can specify what number to start with

```python
for i in range(3, 5):
    print(i)
```

We can also specify how much we increment

```python
for i in range(3, 10, 2):
    print(i)
```

Output

```
0
1
2
3
4
```

```
3
4
```

```
3
5
7
9
```

How does it work

What is happening behind the scenes

In order to understand for loops in python, we must understand what **range()** is doing

```
a = list(range(5))
print(a)
```

Output

```
[0, 1, 2, 3, 4]
```

The elements match the print outputs

```
print(list(range(3, 5)))
print(list(range(3, 10, 2)))
```

Output

```
[3, 4]
[3, 5, 7, 9]
```

While Loops

We can take the range() function and split it up

```python
i = 2 # Starting point
while i < 10: # end point
    print(i)
    i += 3 # increment
```

```
2
5
8
```

As long as the statement inside of the while loop is True, it will continue to run

Loop Functions

Making loops more useful

Even though For and While loops are very useful, they are very smart with how their work.

We can add some more functionality using:

```
break
continue
pass
```

Break

Kill it all, quickly please!!!

The **break** statement allows us to end a loop early if we get the desired output

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

print("---")
for i in range(5):
    print(i)
    if i == 2:
        break
```

```
0
1
2
3
4
---
0
1
2
```

Continue

I don’t like that one...

The **continue** statement will skip that instance of the loop, and move on to the next instance

```python
for i in range(3):
    print(i)

print("---")

for i in range(3):
    if i == 1:
        continue
    print(i)
```

```
0
1
2
---
0
2
```

Pass

We will come back to it later...

The **pass** statement will just allow us to leave loops empty for later usage.

```
for i in range(5):
    pass
```

Exercise Two

Choose your own path...

Take a number from the user

Print out every number between the user’s number and the square of their number

Exercise Two Code

```python
user = int(input("Enter a number: "))

for i in range(user, user*user+1):
    print(i)
```

```python
user = int(input("Enter a number: "))

i=user
while i < user*user+1:
    print(i)
    i+=1
```

```python
user = int(input("Enter a number: "))

i=user
while True:
    print(i)
    i+=1
    if i == user**2+1:
        break
```