Python

prepared by: Ms. Soofi Shafiya
NET-JRF, JMI

-Branching & Control

Python If ... Else

Python Conditions and If statements

- 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

If statement:

```
a = 33
b = 200
if b > a:
  print("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.

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

Elif

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Else

```
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")
```

```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

Nested IF

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

The pass Statement

'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
```

Python While Loops

Python has two primitive loop commands:

- While
- For

The while loop

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Note: remember to increment i, or else the loop will continue forever.

The break Statement

Example

Exit the loop when I is 3

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

The continue Statement

Example

Continue to the next iteration if i is 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

The else Statement

Example

Print a message once the condition is false:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Python For Loops

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).
- This is less like the for keyword in other programming languages and works more like an iterator method as found in other object-orientated programming languages.

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Looping Through a String

- The for loop does not require an indexing variable to set beforehand.

Example

Loop through the letters in the word "banana":

```
for x in "banana":
  print(x)
```

The break Statement

- With the break statement we can stop the loop before it has looped through all the items:

Example

Exit the loop when x is “banana”:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

The continue Statement

Example

Do not print banana:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

The range() Function

To loop through a set of code a specified number of times, we can use the range() function:

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

```
for x in range(6):
  print(x)
```

The range() function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2,6) which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

Example

Increment the sequence with 3 (default is 1):

```
for x in range(2, 30, 3):
  print(x)
```

Else in For Loop

Example

Print all numbers from 0 to 5, and print a message when the loop has ended:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Exercise

Break the loop when x=3, and see what happens with the else block!

Nested Loops

- A nested loop is a loop inside a loop.
- The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

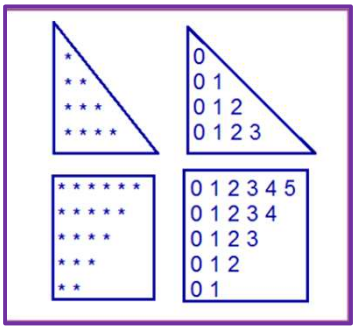
Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

Steps to Print Pattern in Python

1. Decide the number of rows and columns
2. Iterate rows
3. Iterate columns
4. Print star or number
5. Add a new line after each iteration of the outer loop



Example: **Program to print number pattern**

```
rows = 6 # if you want user to enter a number, uncomment the below line
# rows = int(input('Enter the number of rows'))
# outer loop
for i in range(rows):
# nested loop
      for j in range(i):
      # display number
            print(i, end=" " )
      # new line after each row
      print('')
```

Pyramid pattern of numbers

```
rows = 5
for i in range(1, rows + 1):
      for j in range(1, i + 1):
           print(j, end=' ‘)
      print('')
```

Inverted pyramid pattern of numbers

```
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
```

```
rows = 5
b = 0
# reverse for loop from 5 to 0
for i in range(rows, 0, -1):
      b += 1
      for j in range(1, i + 1):
            print(b, end=' ‘)
      print('\r')
```

Inverted Pyramid pattern with the same digit

```
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
```

```
rows = 5
num = rows
# reverse for loop
for i in range(rows, 0, -1):
     for j in range(0, i):
          print(num, end=' ‘)
     print("\r")
```