Part 2 - Conditions and Loops

Conditions

```
#Python uses boolean variables to evaluate conditions.
#The boolean values True and False are returned when an expression is
compared or evaluated.
```

```
x = 2
print(x == 2) # prints out True
print(x == 3) # prints out False
print(x < 3) # prints out True
```

```
a = 30
b = 20
if(a>b):
    print("a is greater than b")
```

```
#The "and" and "or" boolean operators allow building complex boolean
expressions, for example:
```

```
name = "John"
age = 23
if name == "John" and age == 23:
    print("Your name is John, and you are also 23 years old.")

if name == "John" or name == "Rick":
    print("Your name is either John or Rick.")
```

```
#The "in" operator could be used to check if a specified object exists
within an iterable object container, such as a list:
```

```
if name in ["John", "Rick"]:
    print("Your name is either John or Rick.")
```

```
#Python uses indentation to define code blocks, instead of brackets.
The standard Python indentation is 4 spaces, although tabs and any
other space size will work, as long as it is consistent. Notice that
code blocks do not need any termination.

#if-else:
```

```
marks = int(input("Enter student marks"))
if(marks>=40): # if marks are greater than equal to 40
    print("you have passed the exam")
else:
    print("Sorry you have failed the exam")
```

```

```

#Example of if-elif-else

```
if name == "John" and age == 23:
    print("Your name is John")
    print("You are also 23 years old.")
elif name == "John": # else if
    print("Your name is John")
    print("You are not 23 years old")
else:
    print("Your name is not John")
```

Using "not" before a boolean expression inverts it:

```
print(not False) # Prints out True
print((not False) == (False)) # Prints out False
```

#nested ifs

```
age = 22
run_a_mile_time = 3.56
height = 1.82
if(age>18):
    if(run_a_mile_time<4):
        if(height>1.75):
            print("Congratulations! military recruitment division has
shortlisted you as one of the candidates")
```

Loops

#There are two types of loops in Python, for and while.

#for loop
#iterate over a given sequence

```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
```

#using range() to return a list of numbers

```
#Prints out the numbers 0,1,2,3,4
for x in range(5):
    print(x)
```

#Prints out 3,4,5

```
for x in range(3, 6):
    print(x)
```

#Prints out 3,5,7

```
for x in range(3, 8, 2):
    print(x)
```

#while loop

```
count = 0
while count < 5:
    print(count)
    count += 1
```

#break is used to exit a for loop or a while loop, whereas continue is used to skip the current block, and return to the "for" or "while" statement.
Prints out 0,1,2,3,4

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
```

Prints out only odd numbers - 1,3,5,7,9

```
for x in range(10):
    # Check if x is even
    if x % 2 == 0:
        continue
    print(x)
```

#nested loops

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