LECTURE 5

DOHYUNG KIM

WHAT IS DISCUSSED IN THE LAST CLASS

- Conditionals in python

TODAY, WE WILL LEARN ABOUT

- Loops in python
  - For-loop
  - While-loop
  - Break / Continue

TODAY, WE WILL LEARN ABOUT

- Loops in python
  - For-loop
  - While-loop
  - Break / Continue

FOR LOOPS

- To repeat the same instruction 4 times

```
for i in range(4):
  print("Python is fantastic")
```

```
Python is fantastic
Python is fantastic
Python is fantastic
Python is fantastic
>
```

- Sum numbers from m to n

```
def sumToM(m):
  total = 0
  for i in range(m+1):
    total += i
  return total

def sumFromMToN(m, n):
  return sumToM(n) - sumToM(m-1)

print(sumFromMToN(1, 10))
```

FOR LOOPS

- Actually, for the sum from m to n, we don't need looping. We can use built-in function instead.

```
def sumFromMToN(m, n):
  return sum(range(m, n+1))

print(sumFromMToN(1, 10))
```

- Or we can use our mathematical knowledge

```
def sumToN(n):
  return n*(n+1)//2

def sumFromMToN_byFormula(m, n):
  return (sumToN(n) - sumToN(m-1))

print(sumFromMToN_byFormula(1, 10))
```

WAIT A SECOND!

- **range()** returns a range object, that is, a sequence of integers
- It is generally used to iterate over with the loop

- **range()** with a single parameter

```
for i in range(4):
  print("i=", i)
```

```
i= 0
i= 1
i= 2
i= 3
>
```

- **range()** with two parameters

```
for i in range(2, 4):
  print("i=", i)
```

```
i= 2
i= 3
>
```

- **range()** with three parameters

```
for i in range(1, 7, 2):
  print("i=", i)
```

```
i= 1
i= 3
i= 5
>
```

- **range()** with the negative third parameters

```
for i in range(7, 1, -2):
  print("i=", i)
```

```
i= 7
i= 5
i= 3
>
```

PRACTICE

- Write a function to get sum of evens from M to N

```
def sumOfEvensFromMToN(m,n):
  if (m%2 == 1):
    m += 1
  return sum(range(m, n, 2))

print(sumOfEvensFromMToN(1,5))
```

Change the code using **for-loop** instead of the built-in function sum

NESTED FOR LOOP

```
def nestedLoopTest(x, y):
  for i in range(x):
    for j in range(y):
      print ("x=", i, "y=", j)

nestedLoopTest(2,3)
```

```
x= 0 y= 0
x= 0 y= 1
x= 0 y= 2
x= 1 y= 0
x= 1 y= 1
x= 1 y= 2
```

- The value of **i** changes from 0 to 1
- The value of **j** changes from 0 to 2
- Here, look at the sequence of change on i, j
  - **i=0 —>** j=0, 1, 2 and then
  - **i=1 —>** j=0, 1, 2

NESTED FOR LOOP

- Let's print a rectangle of asterisks

```
def printRectangle(n):
  for i in range(n):
    for j in range(n):
      print("*", end="")
    print()

printRectangle(4)
```

- Practice : write a function to print a right-angled triangle shown below

```
*
**
***
****
>
```

TODAY, WE WILL LEARN ABOUT

- Loops in python
  - For-loop
  - While-loop
  - Break / Continue

WHILE LOOPS

- Print out the left-most digit of the input number

```
def leftmostDigit(n):
  n = abs(n)
  while (n >= 10):
    n = n//10
  return n

print(leftmostDigit(-4938573762))
```

```
4
>
```

- Generally, if the range is fixed, for-loop is recommended. Otherwise, you'd better use while-loop.

```
for i in range (n):
   res += i
```

PRACTICE

- Complete the function below to get multiples of 3 or 7

```
def isMultipleOf3or7(x):
  return ((x % 3) == 0) or ((x % 7) == 0)

def getMultiplesOf3or7(n):
  ...


getMultiplesOf3or7(10);
```

```
3 6 7 9 12 14 15 18 21 24 :
```

TODAY, WE WILL LEARN ABOUT

- Loops in python
  - For-loop
  - While-loop
  - Break / Continue

BREAK / CONTINUE STATEMENT

- Are used to alter the flow of a normal loop

```
for n in range(200):
  if (n % 3 == 0):
    continue
  elif (n == 8):
    break
  print(n, end=" ")
print()
```

- **Break**
  - Terminates the loop containing it.
  - The flow comes out of the loop body immediately.
- **Continue**
  - Skips the rest of the code inside the loop for the current iteration only
  - Loop does not terminate but continues on with next iteration

INFINITE WHILE WITH BREAK STATEMENT

- Loop continues unless you input the string "done"

```
def readUntilDone():
  linesEntered = 0
  while (True):
    response = input("Enter a string (or 'done' to quit): ")
    if (response == "done"):
      break
    print("  You entered: ", response)
    linesEntered += 1
  print("Bye!")
  return linesEntered

linesEntered = readUntilDone()
print("You entered", linesEntered, "lines (not counting 'done').")
```

PRACTICE CODE

- Get Prime Number

```
def isPrime(n):
  if (n < 2):
    return False
  for factor in range(2,n):
    if (n % factor == 0):
      return False
  return True

for n in range(20):
  if isPrime(n):
    print(n, end=" ")
print()
```

- Write a function to get n-th prime number

QUESTION?