Python Programing

Branch and

loop programming

Introduction to Repetition Structures

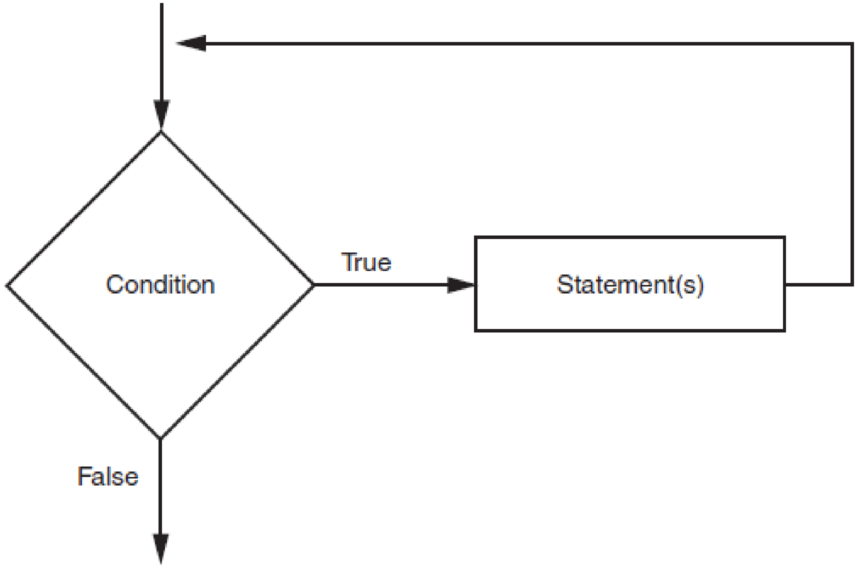
- Often have to write code that performs the same task multiple times
  - Disadvantages to duplicating code
    - Makes program large
    - Time consuming
    - May need to be corrected in many places
- Repetition structure: makes computer repeat included code as necessary
  - Includes condition-controlled loops and count-controlled loops

The `while` Loop

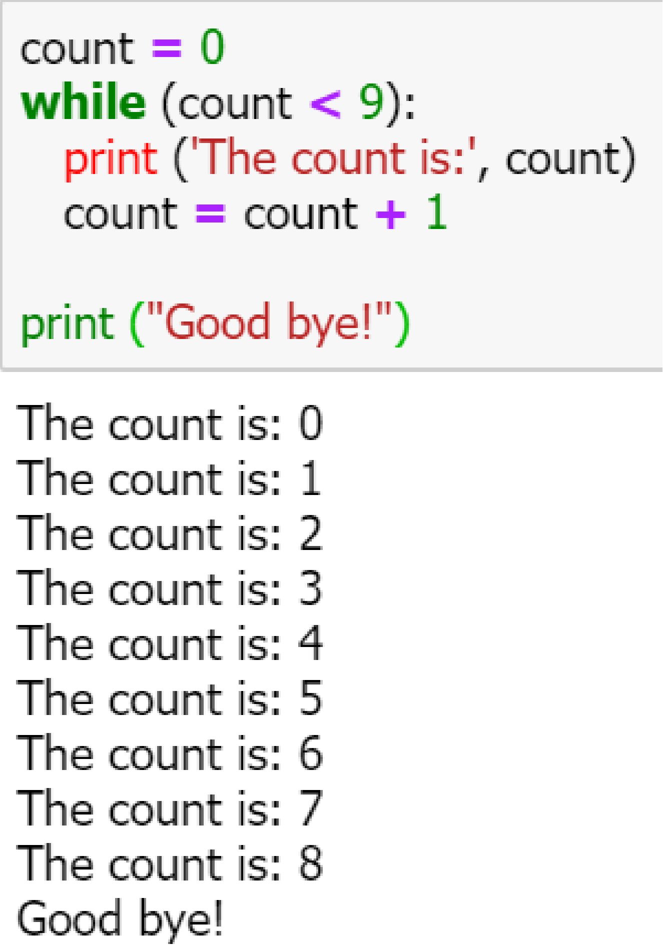
- `while` loop: while condition is true, do something

  - General format:

```
while condition:
        statements
```

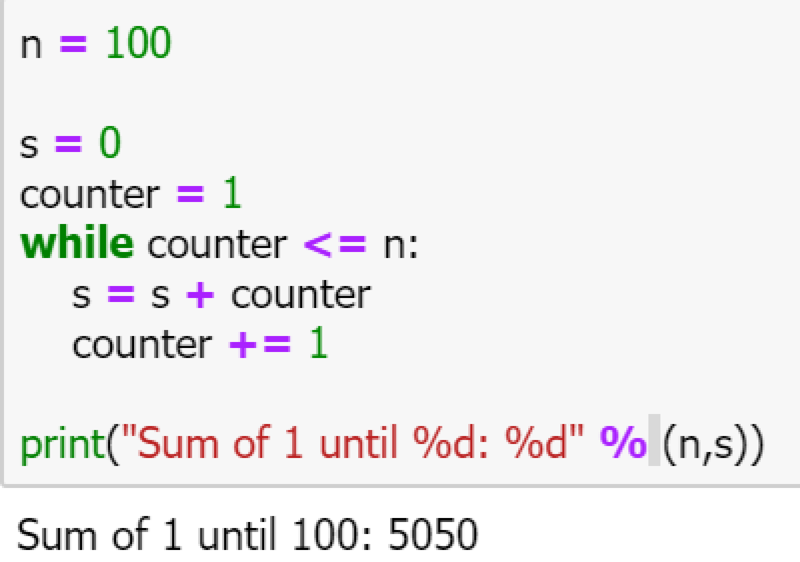


The `while` Loop



```
count = 0
while (count < 9):
   print ('The count is:', count)
   count = count + 1

print ("Good bye!")
```

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
```

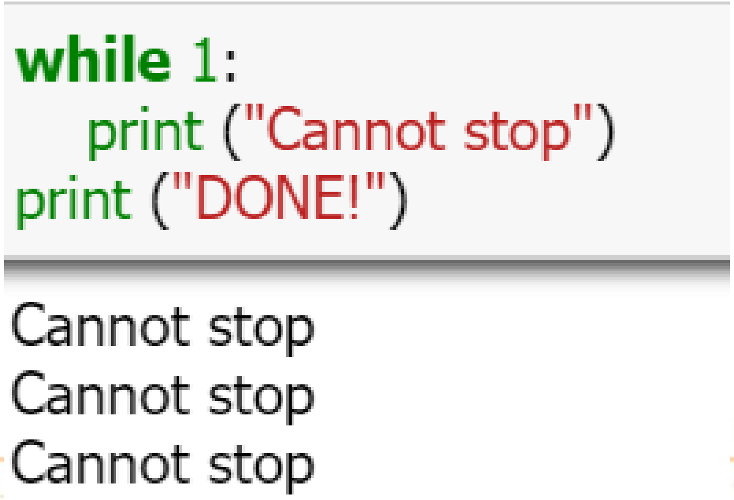


```
n = 100

s = 0
counter = 1
while counter <= n:
    s = s + counter
    counter += 1

print("Sum of 1 until %d: %d" % (n,s))
```

```
Sum of 1 until 100: 5050
```

Infinite Loops

- Infinite loop: loop that does not have a way of stopping
  - Repeats until program is interrupted
  - Occurs when programmer forgets to include stopping code in the loop

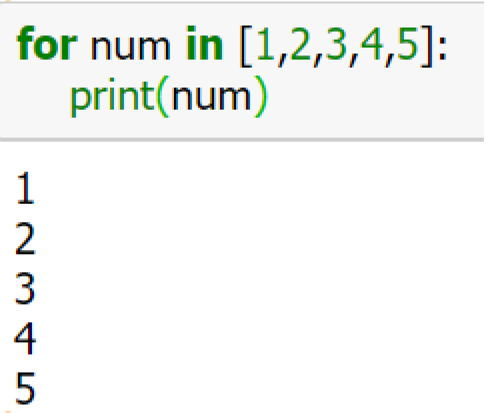


```
while 1:
    print ("Cannot stop")
print ("DONE!")
```

```
Cannot stop
Cannot stop
Cannot stop
```

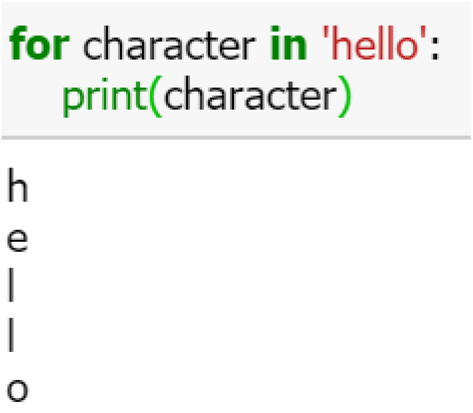
The `for` Loop

```
for variable in [val1, val2, etc]:
    statements
```



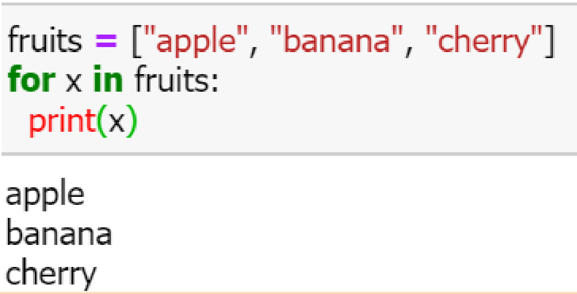
```
for num in [1,2,3,4,5]:
    print(num)
```

```
1
2
3
4
5
```



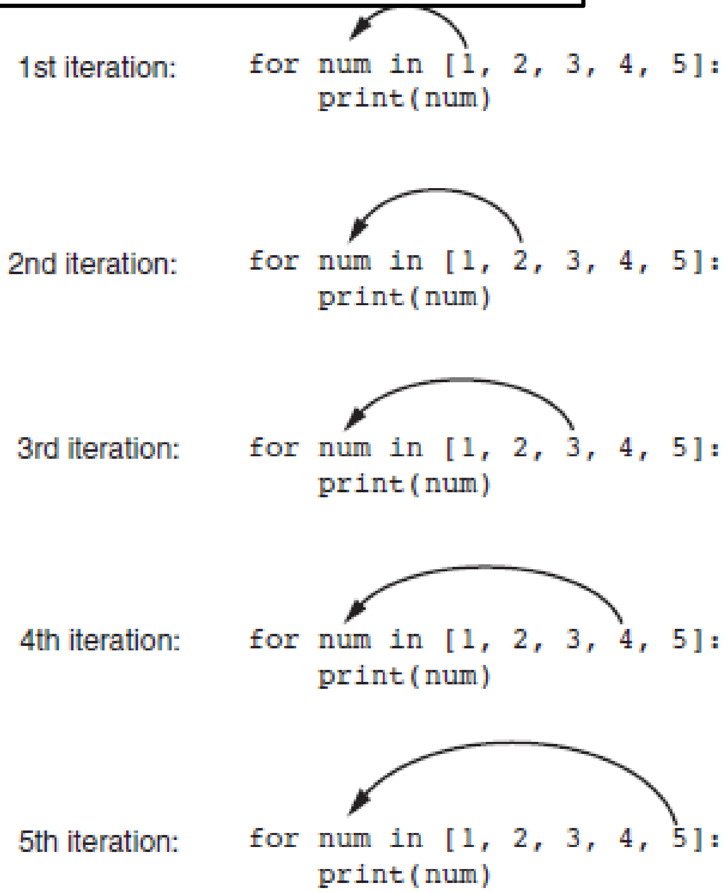
```
for character in 'hello':
    print(character)
```

```
h
e
l
l
o
```



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

```
apple
banana
cherry
```



1st iteration:
```
for num in [1, 2, 3, 4, 5]:
    print(num)
```

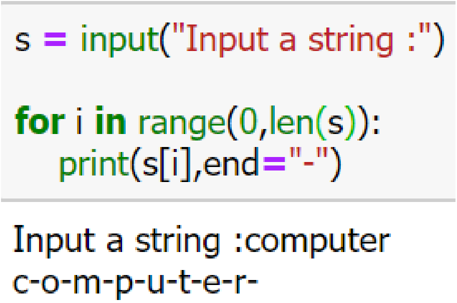
2nd iteration:
```
for num in [1, 2, 3, 4, 5]:
    print(num)
```

3rd iteration:
```
for num in [1, 2, 3, 4, 5]:
    print(num)
```

4th iteration:
```
for num in [1, 2, 3, 4, 5]:
    print(num)
```

5th iteration:
```
for num in [1, 2, 3, 4, 5]:
    print(num)
```

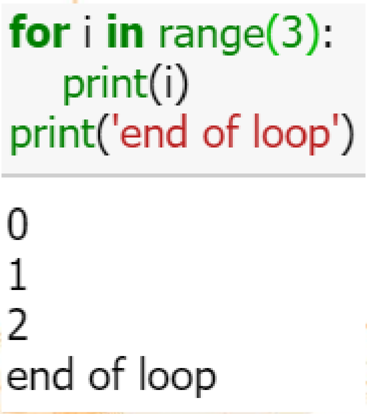
The for Loop - the range Function



```
s = input("Input a string :")

for i in range(0,len(s)):
    print(s[i],end="-")
```

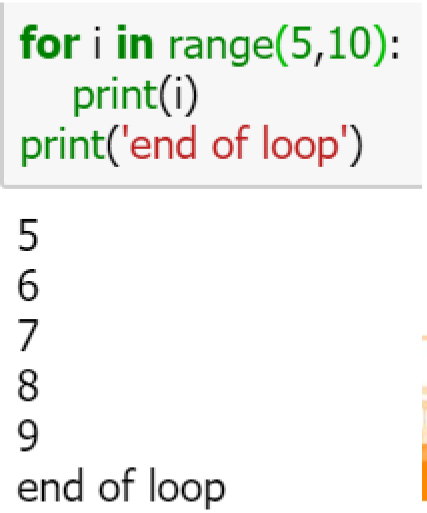
```
Input a string :computer
c-o-m-p-u-t-e-r-
```

- range characteristics:
  - One argument: used as ending limit
  - Two arguments: starting value and ending limit
  - Three arguments: third argument is step value



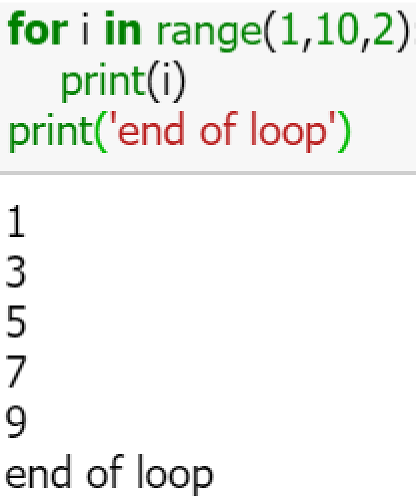
```
for i in range(3):
    print(i)
print('end of loop')
```

```
0
1
2
end of loop
```



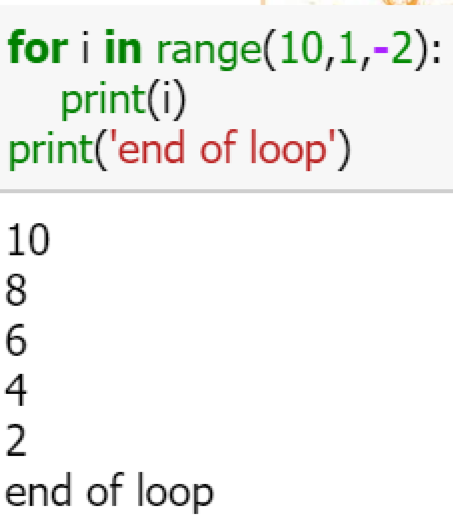
```
for i in range(5,10):
    print(i)
print('end of loop')
```

```
5
6
7
8
9
end of loop
```



```
for i in range(1,10,2):
    print(i)
print('end of loop')
```

```
1
3
5
7
9
end of loop
```

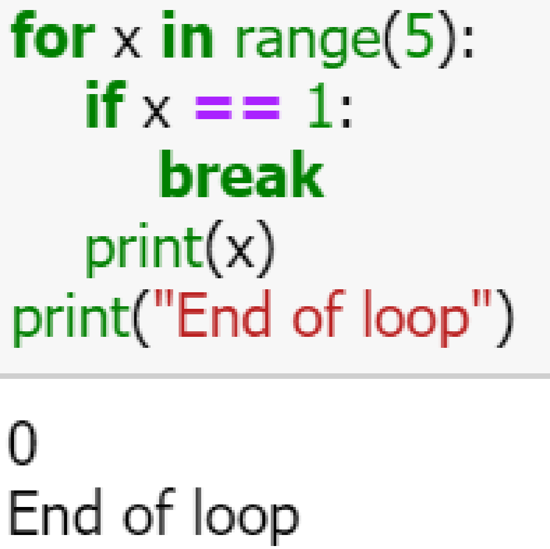


```
for i in range(10,1,-2):
    print(i)
print('end of loop')
```

```
10
8
6
4
2
end of loop
```

The `for,while` Loop – Break and Continue

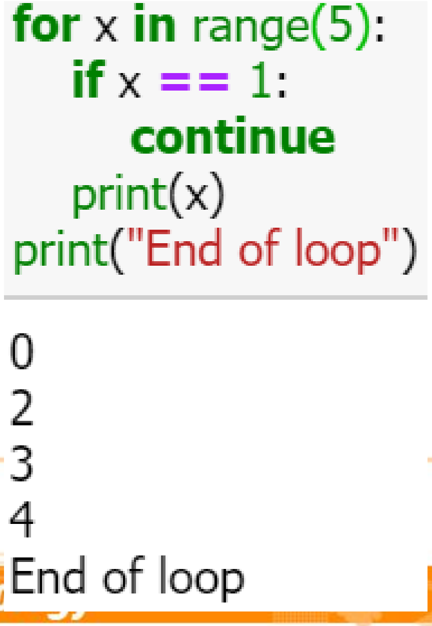
- “**break**” terminates the current loop and resumes execution at the next statement



```
for x in range(5):
    if x == 1:
        break
    print(x)
print("End of loop")
```

```
0
End of loop
```

- “**continue**” rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.



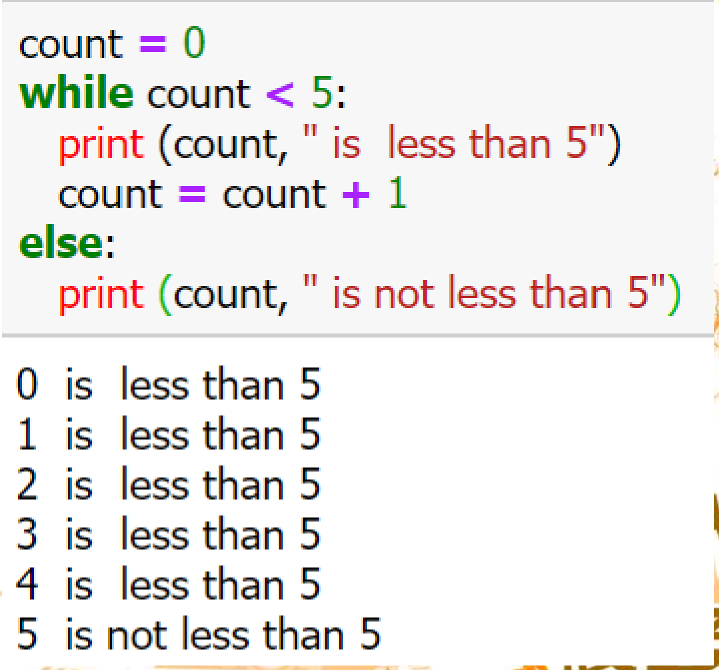
```
for x in range(5):
    if x == 1:
        continue
    print(x)
print("End of loop")
```

```
0
2
3
4
End of loop
```

The `while`, `for` Loop - The else Part

- If the **else** statement is used with a **while and for** loop, the **else** statement is executed when the condition becomes false.

```
while condition:
          statement_1
          ...
          statement_n
else:
          statement_1
          ...
          statement_n
```

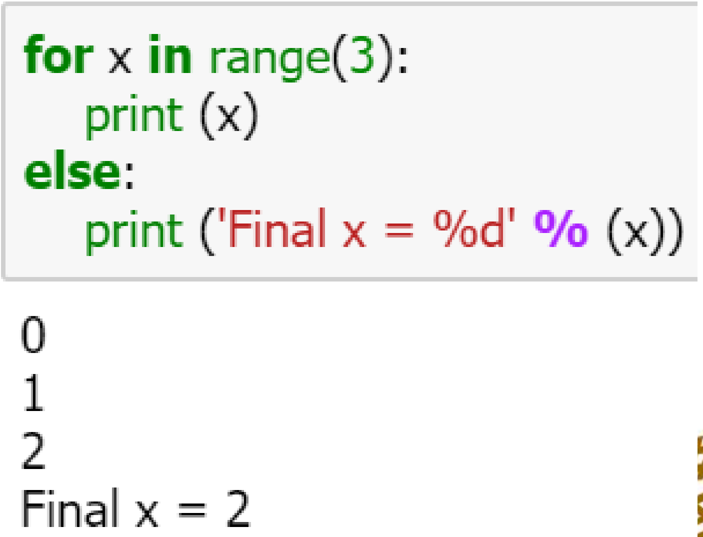


```
count = 0
while count < 5:
   print (count, " is  less than 5")
   count = count + 1
else:
   print (count, " is not less than 5")
```

```
0  is  less than 5
1  is  less than 5
2  is  less than 5
3  is  less than 5
4  is  less than 5
5  is not less than 5
```

The while, for Loop - The else Part

```
for variable in [val1, val2, etc]:
        statements
else:
        statement_1
        ...
        statement_n
```



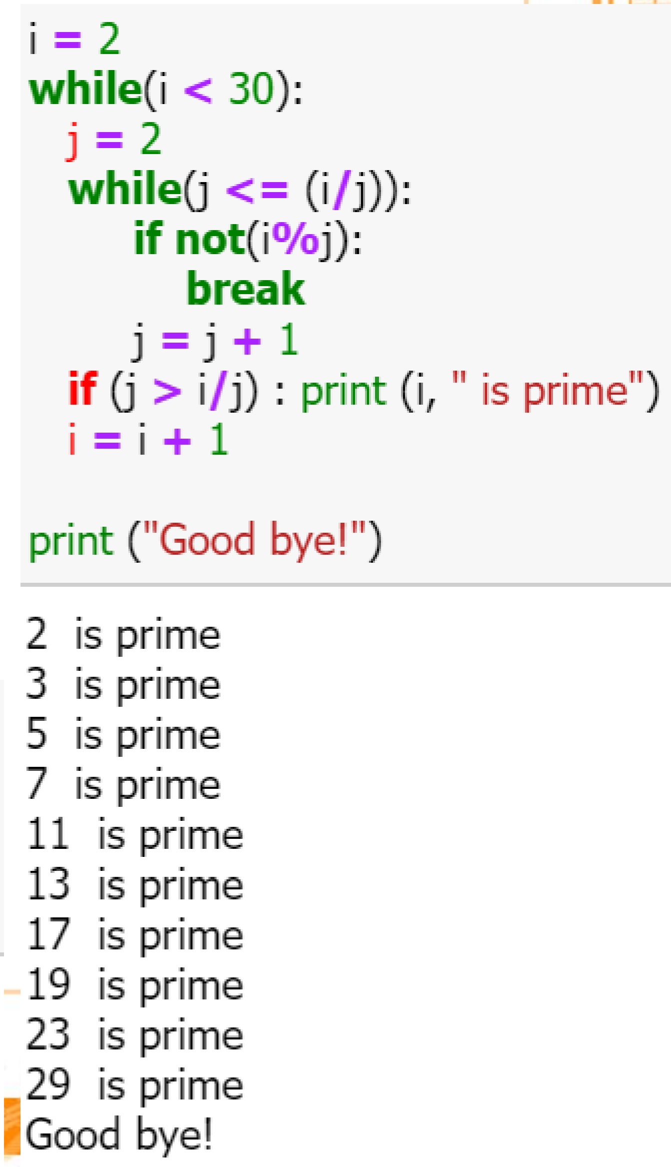
```
for x in range(3):
    print (x)
else:
    print ('Final x = %d' % (x))
```

```
0
1
2
Final x = 2
```

Nested loop – while loop

```
while expression:
   while expression:
      statement(s)
   statement(s)
```

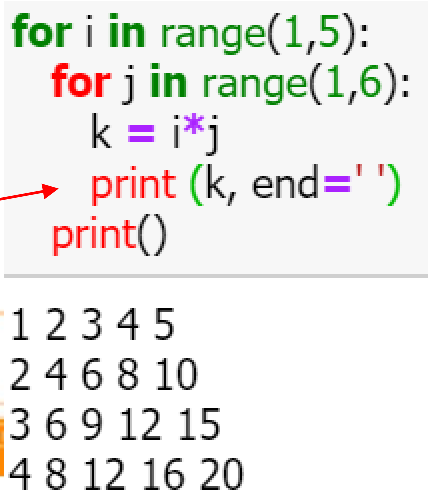
```
for iterating_var in sequence:
   for iterating_var in sequence:
      statements(s)
   statements(s)
```



```
i = 2
while(i < 30):
   j = 2
   while(j <= (i/j)):
      if not(i%j):
         break
      j = j + 1
   if (j > i/j) : print (i, " is prime")
   i = i + 1

print ("Good bye!")
```

```
2  is prime
3  is prime
5  is prime
7  is prime
11  is prime
13  is prime
17  is prime
19  is prime
23  is prime
29  is prime
Good bye!
```

The print() function inner loop has **end=' '** which appends a space instead of default newline. Hence, the numbers will appear in one row.



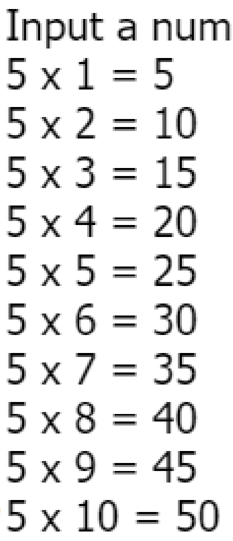
```
for i in range(1,5):
   for j in range(1,6):
      k = i*j
      print (k, end=' ')
   print()
```

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
```

Exercise1

- Write a Python program to create the multiplication table (from 1 to 10) of a number
- Fill …(A)….

Output

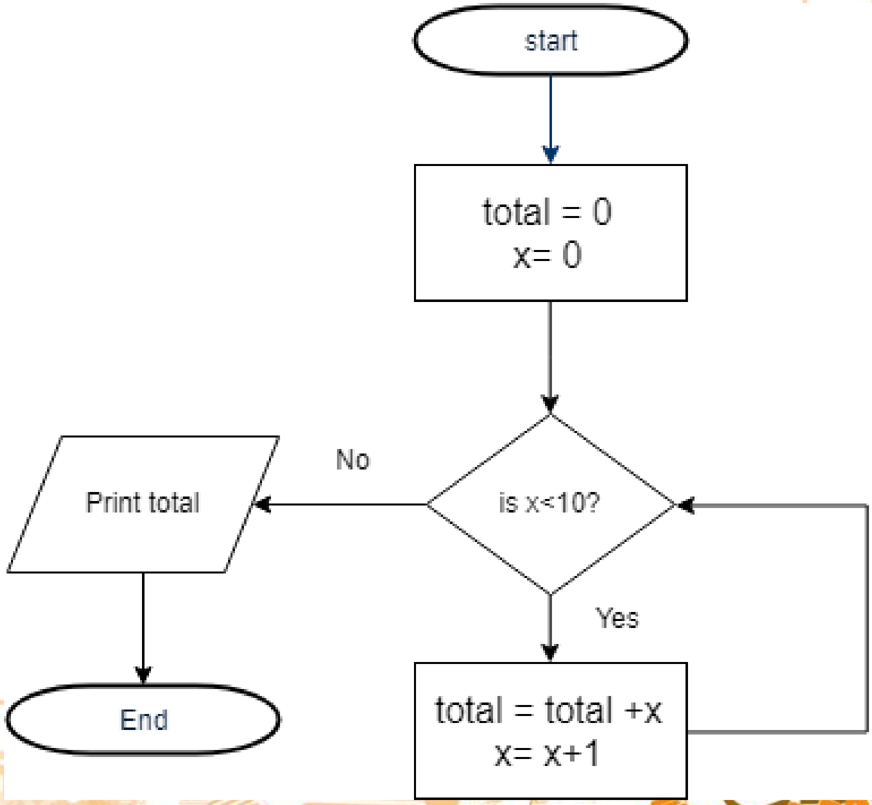


```
Input a number: 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

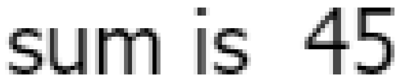
```
n = int(input("Input a number: "))

# use for loop to iterate 10 times
for i in range(1,11):
   …..(A)…..
```

Exercise2

- Find a summation of 0 to 9
- Try to use while loop and for loop, then see a difference



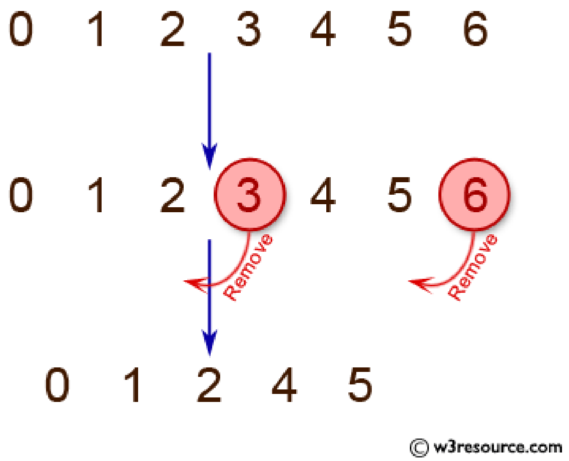
Output



```
sum is 45
```

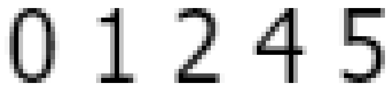
Exercise3

- Write a Python program that prints all the numbers from 0 to 6 except 3 and 6. Note : Use '**continue**' statement.
- Using for loop



© w3resource.com

Output

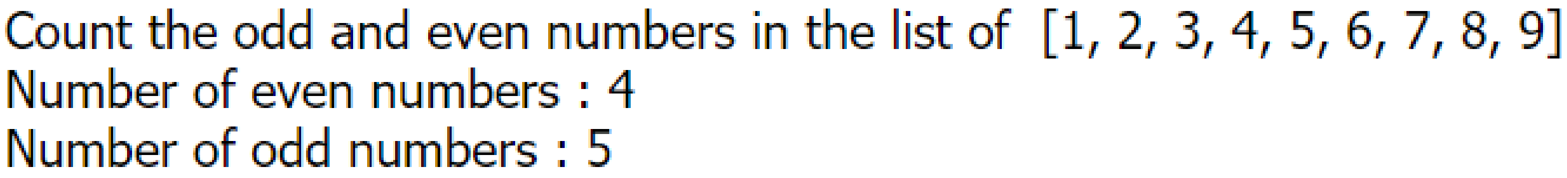


```
0 1 2 4 5
```

Exercise4

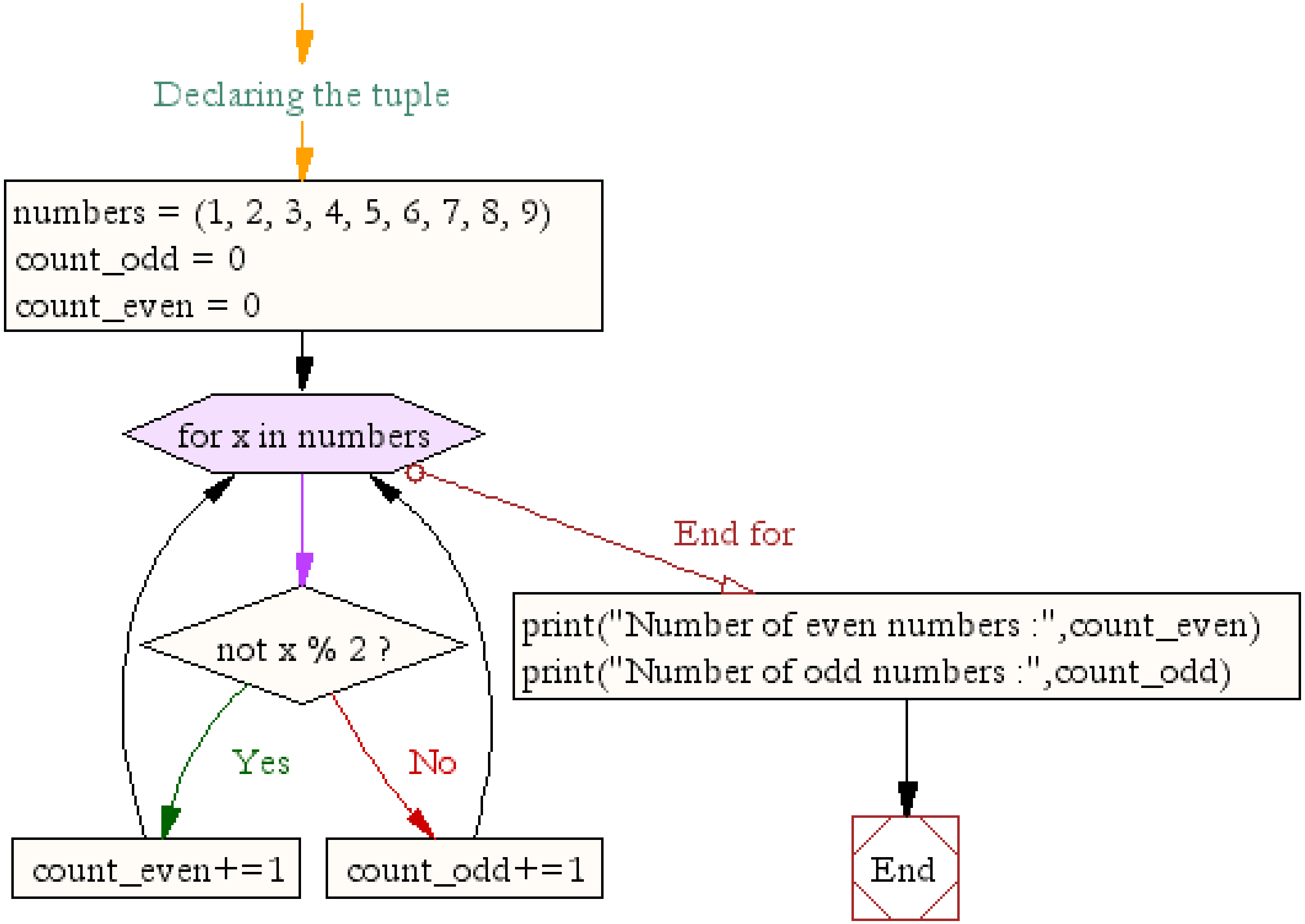
- Write a Python program to count the number of even and odd numbers from a series of numbers.
- Define a list of numbers =[1, 2, 3, 4, 5, 6, 7, 8, 9]
- Using for loop

Output



```
Count the odd and even numbers in the list of  [1, 2, 3, 4, 5, 6, 7, 8, 9]
Number of even numbers : 4
Number of odd numbers : 5
```

Exercise4


Declaring the tuple
numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9)
count_odd = 0
count_even = 0
for x in numbers
not x % 2 ?
Yes
No
count_even+=1
count_odd+=1
End for
print("Number of even numbers :",count_even)
print("Number of odd numbers :",count_odd)
End

Exercise5

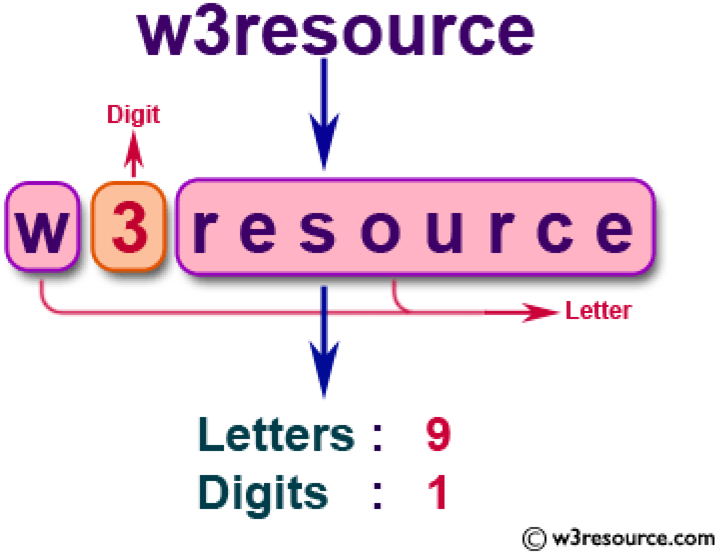
- A Python program construct the following pattern, using a nested loop number

```
for i in range(10):
        print(str(i) * i)
```

- See the result

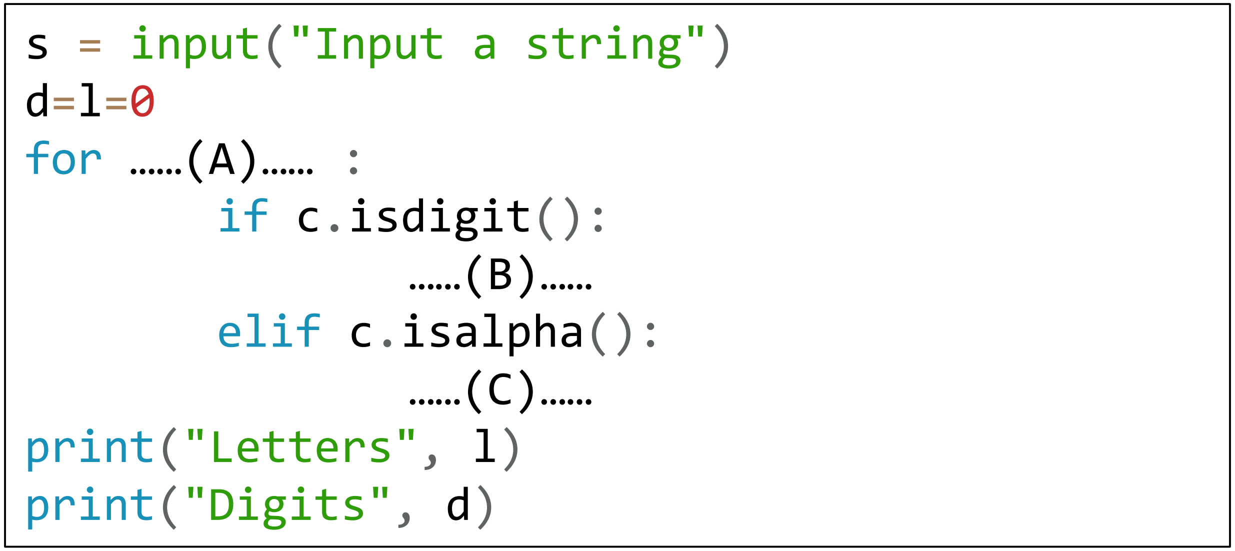
Exercise6

- Write a Python program that accepts a string and calculate the number of digits and letters.



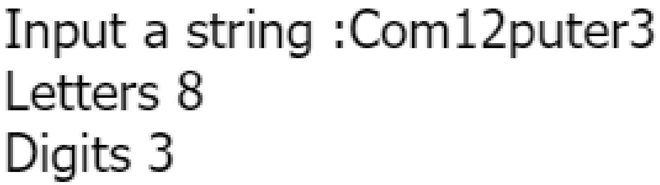
Exercise6

- Fill in the space A,B,C



```
s = input("Input a string")
d=l=0
for ……(A)…… :
       if c.isdigit():
              ……(B)……
       elif c.isalpha():
              ……(C)……
print("Letters", l)
print("Digits", d)
```

Output

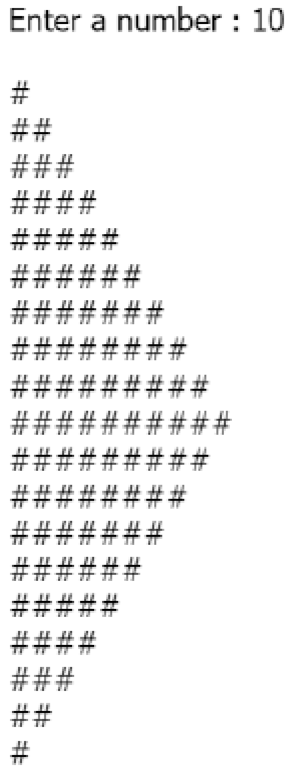


```
Input a string :Com12puter3
Letters 8
Digits 3
```

CW 4-1

- Write a Python program to construct the following pattern, using a nested for loop.

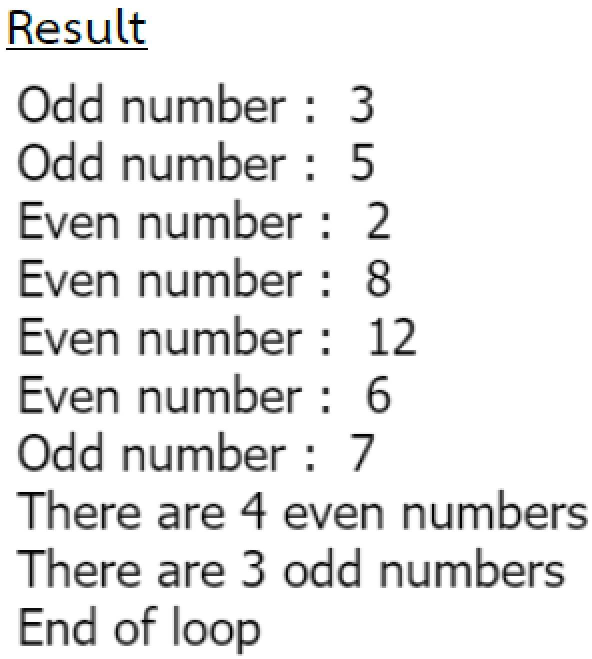
Result



```
Enter a number : 10

#
##
###
####
#####
######
#######
########
#########
##########
#########
########
#######
######
#####
####
###
##
#
```

CW 4-2

- Write a Python program to count and print the odd or even numbers in a list of [3,5,2,8,12,6,7]



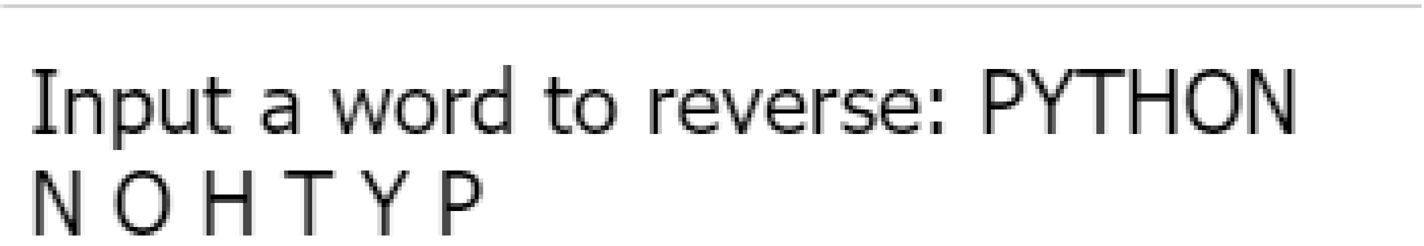
Result

```
Odd number :  3
Odd number :  5
Even number :  2
Even number :  8
Even number :  12
Even number :  6
Odd number :  7
There are 4 even numbers
There are 3 odd numbers
End of loop
```

CW 4-3

- Write a Python program that accepts a word from the user and reverse it.

- **Result**



```
Input a word to reverse: PYTHON
N O H T Y P
```