3.1 *If* Statement

The most well-known statement type is `if` statement.

```python
x = int(input("Please enter an integer: "))
if x < 0 :
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
    print('More')
```

Those `if`, `elif`, `else` sequence is a substitue of `switch` or `case`

3.1.1 Conditional if statement

`if` statement can be used as like below

```python
# Without conditional statement
score = int(input("Enter score"))
if score >= 60:
    message = "success"
else:
    message = "failure"
print(message)

# With conditional statement
score = int(input("Enter score : "))
message = "success" if score >= 60 else "failure"
print(message)
```

3.2 *for* Statement

The `for` statement iterates over the items of any sequence (a list or a string), in order that they appear in the sequence. For example:

```python
# Measure some strings:
words = ['cat', 'window', 'defenestate']
for w in words:
    print(w,len(w))

# Another example
a = [(1,2),(3,4),(5,6)]
for (first, last) in a:
    print(first + last)
```

```
cat 3
window 6
defenestate 11
3
7
11
```

Code that modifies a collection while iterating over that same collection can be tricky to get right. Instead, it is usually more straight-forward to loop over a copy of the collection or to create a new collection:

```python
# Strategy: Iterate over a copy
for user, status in user.copy().items():
    if status == 'inactive':
        del users[user]

# Strategy: Create a new collection
active_users = {}
for user, status in users.items():
    if status == 'active':
        active_users[user] = status
```

```python
for i in range(2,10):
    for j in range(1,10):
        print(i*j, end=" ") # end=" " for gap between values
    print('') # print('') for \n
```

```
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

3.2.1 List Comprehension

```python
# Without list comprehension
a= [1,2,3,4]
result1 = []
for num in a:
    result1.append(num*3)
print(result1)

# With list comprehension
a= [1,2,3,4]
result2 = [num*3 for num in a]
print(result2)

# Another example
a = [1,2,3,4]
result3 = [num*3 for num in a if num%2==0]
print(result3)
```

```
[3, 6, 9, 12]
[3, 6, 9, 12]
[6, 12]
```

3.3 The `range()` Function

If you do need to iterate over a sequence of numbers, the built-in function `range` comes in handy. It generates arithmetic progressions:

The given end point is never part of the generated sequence; `range(10)` generates 10 values starting from 0. It is also possible to count numbers from specified point, or even negative values; this is called 'step'

```python
# 1
for i in range(5, 10):
    print(i)

# 2
for i in range(0,10,3):
    print(i)

# 3
for i in range(-10,-100,-30):
    print(i)
```

To iterate over the indices of a sequence, you can combine `range()` and `len()` as follows:

```python
a = ['Mary', 'had', 'a', 'little', 'lamb']
for i in range(len(a)):
    print(i, a[i])
```

With the combination of `range()` and `len()`, it is possible to get all list members, even though we don't know the numbers of list's total member quantity. However, in most cases, it's pretty much easier to use `enumerate()` function than using the combination of `range()` and `len()` function

```python
print(range(10))
```

`range()` function seems like list, but actually it isn't. It is an object which returns the successive items of the desired sequence when you iterate over it, but it doesn't really make the list, thus saving space

we say such an object is *iterable*

```python
# 1
sum(range(4)) # 0 + 1 + 2 + 3

# 2
list(range(4))
```

Those `range()` function can be switched into `list` as below:

3.4 *break* and *continue* Statements and *else* Clauses on Loops

The `break` statement, like in C, breaks out of the innermost enclosing `for` or `while` loop

```python
for n in range(2,10):
    for x in range(2,n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
         # Loop fell through without finding a factor
        print(n, 'is a priome number')
```

The `continue` statement, also borrowed from C, continues with the next iteration of the loop:

```python
for num in range(2,10):
    if num % 2 ==0:
        print("Found and even number", num)
        continue
    print("Found and oddn number", num)
```

3.5 *pass* Statements

The pass statement does nothing. It can be used when a statement is required syntactically but the program requieres no action. Fro example:

```python
while True:
    pass # Busy-wait for keyboard interrupt (Ctrl + C)
```

This is commonly used for creating minimal `classes`

```python
class MyEmptyClass:
    pass
```

Another place pass can be used is as a place-holder for a function or conditional body when you are working on new code, allowing you to keep thinking at a more abstract level. The `pass` is silently ignored.

```python
def initlog(*args):
    pass # Remeber to implement this!
```

** *While* Statement

Loop statement which is used at cycle.

```python
treeHit = 0
while treeHit < 10:
    treeHit = treeHit +1
    print("나무를 %d번 찍었습니다." %treeHit)
    if treeHit == 10:
        print("나무 넘어갑니다.")
```

`while` statement example :

```python
# while
prompt = '''
1. Add
2. Del
3. List
4. Quit

Enter number : '''
number = 0W
while number != 4:
    print(prompt)
    number = int(input())

# break
coffee = 10
money = 300
while money:
    print("돈을 받았으니 커피를 줍니다.")
    coffee -= 1
    print("남은 커피의 양은 %d개입니다." %coffee)
    if coffee == 0:
        print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
        break

# break 2
coffee = 10
while True:
    money = int(input("돈을 넣어 주세요: "))
    if money == 300:
        print("커피를 줍니다.")
        coffee = coffee -1
    elif money > 300:
        print("거스름돈 %d를 주고 커피를 줍니다." % (money -300))
        coffee = coffee -1
    else:
        print("돈을 다시 돌려주고 커피를 주지 않습니다.")
        print("남은 커피의 양은 %d개 입니다." % coffee)
    if coffee == 0:
        print("커피가 다 떨어졌습니다. 판매를 중지 합니다.")
        break

# continue
a = 0
while a < 10:
    a = a + 1
    if a % 2 == 0: continue
    print(a)
```

**Infinite loop

```python
while True:
    print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
```

**연습문제

```python
# Q1
a = "Life is too short, you need python"

if "wife" in a: print("wife")
elif "python" in a and "you" not in a: print("python")
elif "shirt" not in a: print("shirt")
elif "need" in a: print("need")
else: print("none")
```

```
shirt
```

```python
# Q2
sum = 0
number = 1
while number <= 1000:
    if number % 3 == 0:
        sum = sum + number
        print("sum is : %d " % sum)
    number += 1
print("total sum is : %d" % sum)
```

```
sum is : 3
sum is : 9
sum is : 18
sum is : 30
sum is : 45
sum is : 63
sum is : 84
sum is : 108
sum is : 135
sum is : 165
sum is : 198
sum is : 234
sum is : 273
sum is : 315
sum is : 360
sum is : 408
sum is : 459
sum is : 513
sum is : 570
sum is : 630
sum is : 693
sum is : 759
sum is : 828
sum is : 900
sum is : 975
sum is : 1053
sum is : 1134
sum is : 1218
sum is : 1305
sum is : 1395
sum is : 1488
sum is : 1584
sum is : 1683
sum is : 1785
sum is : 1890
sum is : 1998
sum is : 2109
sum is : 2223
sum is : 2340
sum is : 2460
sum is : 2583
sum is : 2709
sum is : 2838
sum is : 2970
sum is : 3105
sum is : 3243
sum is : 3384
sum is : 3528
sum is : 3675
sum is : 3825
sum is : 3978
sum is : 4134
sum is : 4293
sum is : 4455
sum is : 4620
sum is : 4788
sum is : 4959
sum is : 5133
sum is : 5310
sum is : 5490
sum is : 5673
sum is : 5859
sum is : 6048
sum is : 6240
sum is : 6435
sum is : 6633
sum is : 6834
sum is : 7038
sum is : 7245
sum is : 7455
sum is : 7668
sum is : 7884
sum is : 8103
sum is : 8325
sum is : 8550
sum is : 8778
sum is : 9009
sum is : 9243
sum is : 9480
sum is : 9720
sum is : 9963
sum is : 10209
sum is : 10458
sum is : 10710
sum is : 10965
sum is : 11223
sum is : 11484
sum is : 11748
sum is : 12015
sum is : 12285
sum is : 12558
sum is : 12834
sum is : 13113
sum is : 13395
sum is : 13680
sum is : 13968
sum is : 14259
sum is : 14553
sum is : 14850
sum is : 15150
sum is : 15453
sum is : 15759
sum is : 16068
sum is : 16380
sum is : 16695
sum is : 17013
sum is : 17334
sum is : 17658
sum is : 17985
sum is : 18315
sum is : 18648
sum is : 18984
sum is : 19323
sum is : 19665
sum is : 20010
sum is : 20358
sum is : 20709
sum is : 21063
sum is : 21420
sum is : 21780
sum is : 22143
sum is : 22509
sum is : 22878
sum is : 23250
sum is : 23625
sum is : 24003
sum is : 24384
sum is : 24768
sum is : 25155
sum is : 25545
sum is : 25938
sum is : 26334
sum is : 26733
sum is : 27135
sum is : 27540
sum is : 27948
sum is : 28359
sum is : 28773
sum is : 29190
sum is : 29610
sum is : 30033
sum is : 30459
sum is : 30888
sum is : 31320
sum is : 31755
sum is : 32193
sum is : 32634
sum is : 33078
sum is : 33525
sum is : 33975
sum is : 34428
sum is : 34884
sum is : 35343
sum is : 35805
sum is : 36270
sum is : 36738
sum is : 37209
sum is : 37683
sum is : 38160
sum is : 38640
sum is : 39123
sum is : 39609
sum is : 40098
sum is : 40590
sum is : 41085
sum is : 41583
sum is : 42084
sum is : 42588
sum is : 43095
sum is : 43605
sum is : 44118
sum is : 44634
sum is : 45153
sum is : 45675
sum is : 46200
sum is : 46728
sum is : 47259
sum is : 47793
sum is : 48330
sum is : 48870
sum is : 49413
sum is : 49959
sum is : 50508
sum is : 51060
sum is : 51615
sum is : 52173
sum is : 52734
sum is : 53298
sum is : 53865
sum is : 54435
sum is : 55008
sum is : 55584
sum is : 56163
sum is : 56745
sum is : 57330
sum is : 57918
sum is : 58509
sum is : 59103
sum is : 59700
sum is : 60300
sum is : 60903
sum is : 61509
sum is : 62118
sum is : 62730
sum is : 63345
sum is : 63963
sum is : 64584
sum is : 65208
sum is : 65835
sum is : 66465
sum is : 67098
sum is : 67734
sum is : 68373
sum is : 69015
sum is : 69660
sum is : 70308
sum is : 70959
sum is : 71613
sum is : 72270
sum is : 72930
sum is : 73593
sum is : 74259
sum is : 74928
sum is : 75600
sum is : 76275
sum is : 76953
sum is : 77634
sum is : 78318
sum is : 79005
sum is : 79695
sum is : 80388
sum is : 81084
sum is : 81783
sum is : 82485
sum is : 83190
sum is : 83898
sum is : 84609
sum is : 85323
sum is : 86040
sum is : 86760
sum is : 87483
sum is : 88209
sum is : 88938
sum is : 89670
sum is : 90405
sum is : 91143
sum is : 91884
sum is : 92628
sum is : 93375
sum is : 94125
sum is : 94878
sum is : 95634
sum is : 96393
sum is : 97155
sum is : 97920
sum is : 98688
sum is : 99459
sum is : 100233
sum is : 101010
sum is : 101790
sum is : 102573
sum is : 103359
sum is : 104148
sum is : 104940
sum is : 105735
sum is : 106533
sum is : 107334
sum is : 108138
sum is : 108945
sum is : 109755
sum is : 110568
sum is : 111384
sum is : 112203
sum is : 113025
sum is : 113850
sum is : 114678
sum is : 115509
sum is : 116343
sum is : 117180
sum is : 118020
sum is : 118863
sum is : 119709
sum is : 120558
sum is : 121410
sum is : 122265
sum is : 123123
sum is : 123984
sum is : 124848
sum is : 125715
sum is : 126585
sum is : 127458
sum is : 128334
sum is : 129213
sum is : 130095
sum is : 130980
sum is : 131868
sum is : 132759
sum is : 133653
sum is : 134550
sum is : 135450
sum is : 136353
sum is : 137259
sum is : 138168
sum is : 139080
sum is : 139995
sum is : 140913
sum is : 141834
sum is : 142758
sum is : 143685
sum is : 144615
sum is : 145548
sum is : 146484
sum is : 147423
sum is : 148365
sum is : 149310
sum is : 150258
sum is : 151209
sum is : 152163
sum is : 153120
sum is : 154080
sum is : 155043
sum is : 156009
sum is : 156978
sum is : 157950
sum is : 158925
sum is : 159903
sum is : 160884
sum is : 161868
sum is : 162855
sum is : 163845
sum is : 164838
sum is : 165834
sum is : 166833
total sum is : 166833
```

```python
# Q3
n = 1
while n < 6:
    print("*" * n)
    n += 1
```

```
*
**
***
****
*****
```

```python
# Q4
for n in range(1,101):
    print(n)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```python
# Q5

sum = 0
score = [ 70, 60, 55, 75, 95, 90, 80, 80, 85, 100]
for i in score:
    sum += i
print(sum/len(score))
```

```
79.0
```

```python
# Q6

# without list comprehension
numbers = [1,2,3,4,5]
result = []
for n in numbers:
    if n % 2 ==1:
        result.append(n*2)
print(result)
print("*" * 50)

# with list comprehension
numbers2 = [1,2,3,4,5]
result2 = [n*2 for n in numbers2 if n%2 ==1]
print(result2)
```

```
[2, 6, 10]
**************************************************
[2, 6, 10]
```