Week 2

① >>> "hello, I'm good"
"hello, I'm good"

>>> 'hello, I'm good'
invalid syntax

>>> 'hello, I\'m good'
"hello, I'm good"

>>> "hello, I'm good"
"hello, I'm good"

② >>> "ba" + "na" * 2 + "muffin"
bananamuffin
no space when '' + ''

③ print("hello")
hello
no quotation marks

④ >>> print("hello", "there")
hello there
Space by default

>>> print("hello" + "there")
hellothere
no space
first concate two str then print

⑤ ☆
```
>>> def square_return(num: int) -> int:
        return num ** 2
>>> def square_print(num: int) -> int:
        print("The square of num is", num ** 2)
>>> a_return = square_return(4)
>>> a_return
16
>>> a_print = square_print(4)
The square of num is 16
>>> a_print
>>> a_return * 5
80
>>> a_print * 5
can't 'NoneType' and 'int'
```

Store the answer in a_return

excute print but not store in a_print

a_return refers numeric value

a_print refers None since no numeric value is stored when excuting print

⑥ >>> age = input("How old are you?")
How old are you 7
>>> age
'7'
input is a str, '7' is a str.

⑦ \n - new line character
>>> print('3\t4\t5')
3 4 5
>>> print('\\')
\
>>> print("He says, \"hi\".")
He says, "hi".
>>> print("\\n is good")
\n is good
>>> print("OK, \n is good")
OK,
is good

⑧ >>> "'that's okay'" >>> '"nice", she said.' >>> print('"nice", she said.')
"that's okay" '"nice", she said.' "nice", she said.

⑨ The last step of Design Recipe is Test

Week 3

---

① >>> 'lines' in 'smiles' >>> 'I' in 'smiles' >>> 'mile' in 'smiles'
False False True
consider the whole string and lower case (uppercase)

② >>> s = 'nice'
>>> s[-0]
n
[-0] ⇐⇒ [0]

③ 8 == 8.0 ✓ 8 == 8.00000 ✓

④ >>> int(99.9)
99
int( ) of any float will be round down

⑤ >>> len('') ⇒ 0 not None

Week 4

① `>>> '123'.isdigit()` → False `>>> '12.34'.isalnum()` → False
need to consider all characters

②
```
digits= '0123456789'
result = ''
for digit in digits:
    result = result + digit *2
print(result)
```
digits refers to a string, so when the string *2, the number of occurence of that digit time 2, not the value.
Since we have print(), the '' quotation is remove

ans: 00112233445566778899

Week 5

①
```
>>>t =[1, 2, 3, 4]
>>> t[-1]   >>>t[3:]   >>> t[3:4]
4           [4]        [4]
```
When slicing, always gives a list that include the wanted element even when one element is sliced.

② `>>> len([1, 2, 3, 4])` → 4 `>>> len("math")` → 4 `>>> min([10, 8, 4])` → 4 `>>> sum([1, 2, 1])` → 4

③
```
>>> L = ['a', 'b', 'c']
>>> L.append('d')
>>> L
['a', 'b', 'c', 'd']
```
```
>>> L = ['a', 'b']
>>> L.append(['e', 'f'])
>>> L
['a', 'b', ['e', 'f']]
```
```
>>> L = ['a', 'b']
>>> L.extend(['c', 'd'])
>>> L
['a', 'b', 'c', 'd']
```

differeces between append and extend method,
if extend or append takes two or more argument, an error occurs.

④
```
>>> 'a' in ['mom', 'dad']
False
```

Week 6

1.
```
def mystery(lst):
    for sublist in lst:
        total = 0
        for num in sublist:
            total = total + num
    return total
```

>>> print(mystery([[10, 20], [20], [40, 10]]))

50 not 100 since total = 0 happens in sublist which means that every time I create a new sublist, total goads to 0. in this case, it only gives the total for the last sublist

2. >>> ['a'] + ['b']

['a', 'b']

no nested list

>>> ['a'].append['b']

None

>>> S = ['a']
>>> S.append(['b'])
>>> S

['a', ['b']]

nested list

if S.extend(['b'])

then S == ['a', 'b']

Week 7

To open a file

open(filename, mode)

mode

'r' to open for reading

'w' to open for writing (erasing what is already in the file)

'a' to open for appending (adding to what is already in the file)

Method

readline()

return the next line from the file, including newline character(if exist)

ex. 'aaaa\n'

'\n'

'' end of the file

for loop

```
for line in file:
    print(line, end='')
```

.....

.....

.....

file.read()

```
print(file.read())
```

.....

.....

.....

file.readlines()

['...\n', '...\n', '...\n', '...\n']

① dictionary keys

{1: 2, 3: 4} ('single',)✓['a', 'b'] (1, 'fred', 2.0)✓

Keys have to be immutable!

② dic = {'a': 1, 'b': 2, 'c': 3}.

ignore "" and '' { 'b' in dic ✓ 'B' in dic ✗ } not ignore uppercase and lowercase.

"b" in dic ✓

③ >>> {1: 10, 1: 50, 1: 30}

{1: 30}

when there're duplicated keys, no error will occur but the key will automatically refers to the last one

Week 8

Palindrome

① reverse the entire string and compare that to the original string

② reverse half the string and compare that to the another half

③ compare pair the first one to the last one until middle is reached.

Week 9

Doctest

```
>>> import doctest
>>> doctest.testmode
```

① __name__ refers to a string containing the name of the module.

② >>> import animals
>>> print(animals.__name__)
animals.?

③ use of __name:

```
if __name__ == '__main__':
    ......
    ......
```

④ >>> import math
>>> math.sqrt(math.pi)

⑤ function calls Vs Method calls

math.trunc(3.2)
len('abc')
pow(3,4)

list.count(['a','b'], 'a')
'moogah'.swapcase()

Week 10

linear search

```
i = 0

while i != len(L) and v != L[i]:
    i = i + 1

if i == len(L):
   return -1
else:
    return i
```

After

binary search

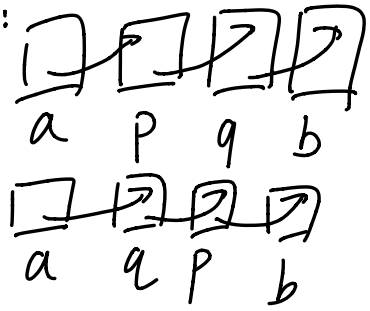
precondition: sorted list

```
b = 0
e = len(L) - 1

while b <= e:
    m = (b + e) // 2
    if L[m] < v:
        b = m + 1
    else:
        e = m - 1

if b == len(L) or L[b] != v:
    return -1
else:
    return b
```

Sorting Algorithms

Bubble sort

```
def bubble_sort(L: list) -> NoneType:
    end = len(L) - 1
    while end != 0
        for i in range(end):
            if L[i] > L[i+1]:
                L[i], L[i+1] = L[i+1], L[i]
        end = end - 1
```



Selection Sort

```
def get_index_of_smallest(L: list, i: int) -> int:
    index_of_smallest = i
    for j in range(i+1, len(L)):
        if L[j] < L[index_of_smallest]:
            index_of_smallest = j
    return index_of_smallest

def selection_sort(L: list) -> None:
    for i in range(len(L)):
        index_of_smallest = get_index_of_smallest(L, i)
        L[index_of_smallest], L[i] = L[i], L[index_of_smallest]
```

Insertion Sort

```
def insert(L: list, i: int) -> None:
    value = L[i]
    while j != 0 and L[j-1] > value:
        L[j] = L[j-1]
        j = j-1
    L[j] = value

def insertion_sort(L: list) -> None:
    for i in range(len(L)):
        insert(L, i)
```