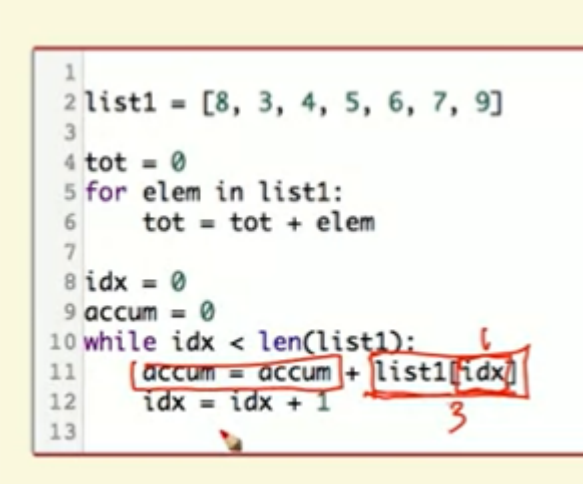
2.4 while loop+advanced functions

- while loop
  - while是更general的for loop for是知道自己要loop几遍 while不确定到底有几次
    - 举例 累加数字
      - def sumTo(aBound):
      - """ Return the sum of 1+2+3 ... n """
      - theSum = 0
      - aNumber = 1
      - while aNumber <= aBound:
      - theSum = theSum + aNumber
      - aNumber = aNumber + 1
      - return theSum
      - print(sumTo(4))
      - print(sumTo(1000))
    - 案例：从0-15 挑出偶数
      - count=0
      - eve_nums=[]
      - while count<=15:
      - if count%2==0:
      - eve_nums.append(count)
      - count=count+1
      - 首先其实是 count=0 while count<=15 count=count+1这个限制了count是从0-15的
      - 之后看下一个even num的条件 也就是module 2为0
      - 再add这些num去list called eve_nums
    - 另一个案例
      - 改写for loop 去while loop



```
list1 = [8, 3, 4, 5, 6, 7, 9]

tot = 0
for elem in list1:
    tot = tot + elem

idx = 0
accum = 0
while idx < len(list1):
    accum = accum + list1[idx]
    idx = idx + 1
```

- 不会：Write a function called stop_at_four that iterates through a list of numbers. Using a while loop, append each number to a new list until the number 4 appears. The function should return the new list.

- the listener loop
  - 只能用whileloop listener就是一直在等某个出现 等到那个出现就停止loop
  - 例子
    - 这里就是 会让你输入很多个数字 当你最终输入0的时候 loop就会停止 然后会把之前写过的数字都加总在一起输出
    - theSum = 0
    - x = -1
    - while (x != 0):
    - x = int(input("next number to add up (enter 0 if no more numbers): "))
    - theSum = theSum + x
    - print(theSum)
    - 问题：为啥前面要有个x=-1呢：为了保证最开始的while loop能够至少执行一次 所以选择了一个初始值 其他任何不是-1的值都可以
  - 另一个超市售货员的例子sentinel values
    - 如果morreItems是True的话 就ask for price 如果price不等于0就count+1 total是total+price 当price=0的时候就会终止while loop了 所以price=0是个sentinel value

```python
def checkout():
    total = 0
    count = 0
    moreItems = True
    while moreItems:
        price = float(input('Enter price of item (0 when done): '))
        if price != 0:
            count = count + 1
            total = total + price
            print('Subtotal: $', total)
        else:
            moreItems = False
    average = total / count
    print('Total items:', count)
    print('Total $', total)
    print('Average price per item: $', average)

checkout()

```

- 输出
- Subtotal: $ 4.0
- Subtotal: $ 8.0
- Subtotal: $ 13.0
- Subtotal: $ 19.0
- Subtotal: $ 26.0
- Total items: 5
- Total $ 26.0
- Average price per item: $ 5.2

- 另一个validating input 回答是非问题
  - d

```python
def get_yes_or_no(message):
    valid_input = False
    while not valid_input:
        answer = input(message)
        answer = answer.upper() # convert to upper case
        if answer == 'Y' or answer == 'N':
            valid_input = True
        else:
            print('Please enter Y for yes or N for no.')
    return answer

response = get_yes_or_no('Do you like lima beans? Y)es or N)o: ')
if response == 'Y':
    print('Great! They are very healthy.')
else:
    print('Too bad. If cooked right, they are quite tasty.')

```

- 乌龟random walk
  - d

```
import random
import turtle


def isInScreen(w, t):
    if random.random() > 0.1:
        return True
    else:
        return False


t = turtle.Turtle()
wn = turtle.Screen()

t.shape('turtle')
while isInScreen(wn, t):
    coin = random.randrange(0, 2)
    if coin == 0:               # heads
        t.left(90)
    else:                       # tails
        t.right(90)

```

```
def isInScreen(wn,t):
    leftBound = -(wn.window_width() / 2)
    rightBound = wn.window_width() / 2
    topBound = wn.window_height() / 2
    bottomBound = -(wn.window_height() / 2)

    turtleX = t.xcor()
    turtleY = t.ycor()

    stillIn = True
    if turtleX > rightBound or turtleX < leftBound:
        stillIn = False
    if turtleY > topBound or turtleY < bottomBound:
        stillIn = False

    return stillIn
```

- Break and continue
  - break要skipped东西直接跳到最后
  - continue也是skipped但是跳回原来condition top of loop 的while那句话
  - 例子
    - 当x《10的时候 print一句话 如果x是偶数就加三再continue回到while loop的句子上 如果是三的倍数就加五再加一 这里好像就算下面的if没有continue还是回到了最初始的地方?

```
x = 0
while x < 10:
    print("we are incrementing x")
    if x % 2 == 0:
        x += 3
        continue
    if x % 3 == 0:
        x += 5
    x += 1
print("Done with our loop! X has the value: " + str(x))
```

```
we are incrementing x
we are incrementing x
we are incrementing x
Done with our loop! X has the value: 15
```

- infinite while loop的例子
  - b = 15
  - while b < 60:
  - b = 5
  - print("Bugs")
  - b = b + 7
  - 这里就是 每次while loop的时候b都要重新回到5上，意味着就算你后面加了7还是重新回到5 永远停不下来

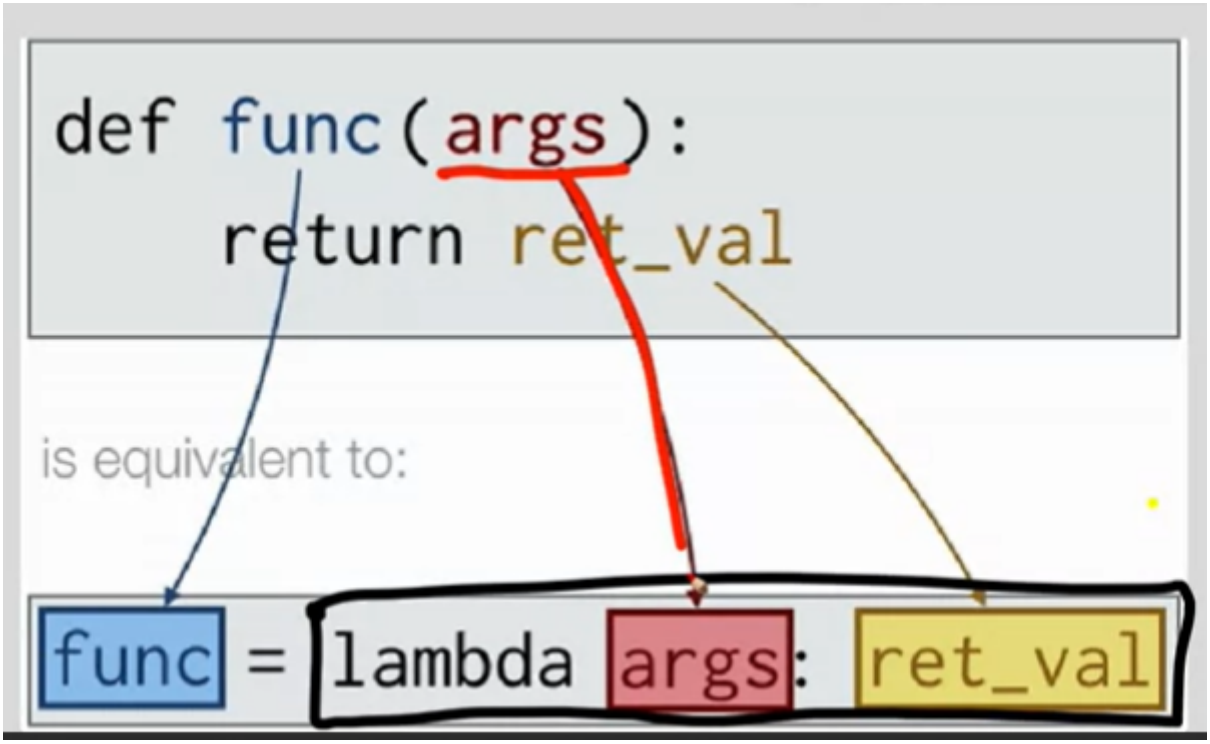
- 考试题
  - Write a function called check_nums that takes a list as its parameter, and contains a while loop that only stops once the element of the list is the number 7. What is returned is a list of all of the numbers up until it reaches 7.
    - 要求 创建一个function 用while loop 这个循环只在选中list的7时停止 要求return一个到7之前轮过的数字的list
    - def check_nums(list):
    - i=0
    - sub_list=[]
    - while i<len(list) and list[i] !=7:
    - sub_list.append(list[i])
    - i=i+1
    - return(sub_list)

- define一个function 用while loop 因为这个while是需要在list中选数字 所以我总共选的次数就是length of list 这是条件1，第二个条件维持while loop的就是 需要选中的数字不为7 然后有一个list去承接这些不是7的数字 使list扩大 这里需要一个有点像forloop里面accum的东西维持每一次下去 最后return一个list

- Below is a for loop that works. Underneath the for loop, rewrite the problem so that it does the same thing, but using a while loop instead of a for loop. Assign the accumulated total in the while loop code to the variable sum2. Once complete, sum2 should equal sum1.
  - sum1 = 0
  - lst = [65, 78, 21, 33]
  - for x in lst:
  - sum1 = sum1 + x
  -
  - sum2=0
  - i=0
  - while i<len(lst):
  - sum2=sum2+lst[i]
  - i=i+1

- advanced functions
  - optional parameters
    - 例子
      - print(int("100"))
      - print(int("100", 10)) # same thing, 10 is the default value for the base
      - print(int("100", 8)) # now the base is 8, so the result is 1*64 = 64 以8为基数
      - 输出
      - 100
      - 100
      - 64
    - 例子
      - initial = 7
      - def f(x, y =3, z=initial): y有3叫做default value
      - print("x, y, z, are: " + str(x) + ", " + str(y) + ", " + str(z))
      - f(2)
      - f(2, 5)
      - f(2, 5, 8)
      - 输出
      - x, y, z, are: 2, 3, 7

        - x, y, z, are: 2, 5, 7
        - x, y, z, are: 2, 5, 8
        - 默认值是在定义函数的时候确定的 而不是在调用函数时确定的the default value for z is determined at the time the function is defined; at that time initial has the value 0.
        - initial=7
        - def f(x,y=3, z=initial):
        - print("x,y,z, are:"+str(x)+", " + str(y) + ", " + str(z))
        - initial=10
        - f(2)
        - 输出的是x, y, z, are: 2, 3, 7
    - 案例
        - 1 def f(a, L=[]):
        - 2 L.append(a)
        - 3 return L
        - 4
        - 5 print(f(1))
        - 6 print(f(2))
        - 7 print(f(3))
        - 8 print(f(4, ["Hello"]))
        - 9 print(f(5, ["Hello"])) 8行和9行这两个是不一样的list of “Hello”
        - 输出
        - [1]
        - [1, 2]
        - [1, 2, 3]
        - ['Hello', 4]
        - ['Hello', 5]
- key word parameter
    - 就是把f()里的设定成 z=8 但是不specify y的值 这样的话就是y取define的值 z取8
    - 但是要注意z=8这种difene了的一定要在没有define的variable后面
- anonymous functions with lambda expression
    - lambda是另一种define function的方式
    -



- 案例
  - def f(x):
  - return x - 1
  - print(f)
  - print(type(f))
  - print(f(3))
  - 输出
  - <function f>
  - <class 'function'>
  - 2
  - print(lambda x: x-2)
  - print(type(lambda x: x-2))
  - print((lambda x: x-2)(6)) 这里6一定要用另一个括号扩开
  - 输出
  - <function <lambda>>
  - <class 'function'>
  - 4
  - 改写
  - def last_char(s):
  - return s[-1]
  - last_char = (lambda s: s[-1])
- prgramming style

- 缩进4格
- import在最前面
- 用两个空行分隔函数定义
- 将函数定义放在一起
- 将顶层语句(包括函数调用)放在程序的底部

- 考试题
  -

Create a function called mult that has two parameters, the first is required and should be an integer, the second is an optional parameter that can either be a number or a string but whose default is 6. The function should return the first parameter multiplied by the second.

In [0]:
```
def mult(x = int() , y = 6):
    return x*y
```

The following function, greeting, does not work. Please fix the code so that it runs without error. This only requires one change in the definition of the function.

In [0]:
```
def greeting(name, greeting="Hello ", excl="!"):
    return greeting + name + excl

print(greeting("Bob"))
print(greeting(""))
print(greeting("Bob", excl="!!!"))
```

Below is a function, sum, that does not work. Change the function definition so the code works. The function should still have a required parameter, intx, and an optional parameter, intz with a defualt value of 5.

In [0]:
```
def sum(intx, intz=5):
    return intz + intx
```

Write a function, test, that takes in three parameters: a required integer, an optional boolean whose default value is True, and an optional dictionary, called dict1, whose default value is {2:3, 4:5, 6:8}. If the boolean parameter is True, the function should test to see if the integer is a key in the dictionary. The value of that key should then be returned. If the boolean parameter is False, return the boolean value "False".

In [0]:
```
def test(x = int(), bo = True, dict1 = {2:3,4:5,6:8}):
    if bo == True:
        if x in dict1.keys():
            return dict1[x]
    else:
        return(bo)
```

Write a function called checkingIfIn that takes three parameters. The first is a required parameter, which should be a string. The second is an optional parameter called direction with a default value of True. The third is an optional parameter called d that has a default value of {'apple': 2, 'pear': 1, 'fruit': 19, 'orange': 5, 'banana': 3, 'grapes': 2, 'watermelon': 7}. Write the function checkingIfIn so that when the second parameter is True, it checks to see if the first parameter is a key in the third parameter; if it is, return True, otherwise return False.

But if the second paramter is False, then the function should check to see if the first parameter is not a key of the third. If it's not, the function should return True in this case, and if it is, it should return False.

In [0]:
```
def checkingIfIn(x = '', direction = True, d = {'apple': 2, 'pear': 1, 'fruit': 19, 'orange': 5, 'banana': 3, 'grape
s': 2, 'watermelon': 7}):
    if direction == True:
        if x in d.keys():
            return True
        else:
            return False
    else:
        if x not in d.keys():
            return True
        else:
            return False
```

We have provided the function checkingIfIn such that if the first input parameter is in the third, dictionary, input parameter, then the function returns that value, and otherwise, it returns False. Follow the instructions in the active code window for specific variable assignmemts.

In [0]:
```
def checkingIfIn(a, direction = True, d = {'apple': 2, 'pear': 1, 'fruit': 19, 'orange': 5, 'banana': 3, 'grapes': 2,
 'watermelon': 7}):
    if direction == True:
        if a in d:
            return d[a]
        else:
            return False
    else:
        if a not in d:
            return True
        else:
            return d[a]

# Call the function so that it returns False and assign that function call to the variable c_false
c_false = checkingIfIn('strawberry')
# Call the fucntion so that it returns True and assign it to the variable c_true
c_true = checkingIfIn('strawberry',False)
# Call the function so that the value of fruit is assigned to the variable fruit_ans
fruit_ans = checkingIfIn('fruit')
# Call the function using the first and third parameter so that the value 8 is assigned to the variable param_check
param_check = checkingIfIn('orange') + checkingIfIn('banana')
```