3.2 Map Filter List Comprehension and ZIP

- Map Function
  - 用map的话前面是function 后面是某list的名字

```
def tripleStuff(a_list):
    new_seq = map(triple, a_list)
    return list(new_seq)

def quadrupleStuff(a_list):
    new_seq = map(lambda value: 4*value, a_list)
    return list(new_seq)

things = [2, 5, 9]
things3 = tripleStuff(things)
print(things3)
things4 = quadrupleStuff(things)
print(things4)

```

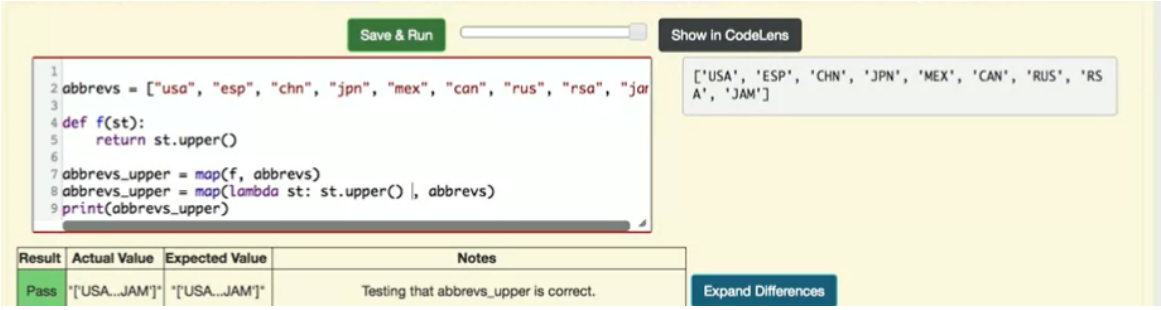
```
[6, 15, 27]
[8, 20, 36]
```

  - 所以要先define一个double or triple function

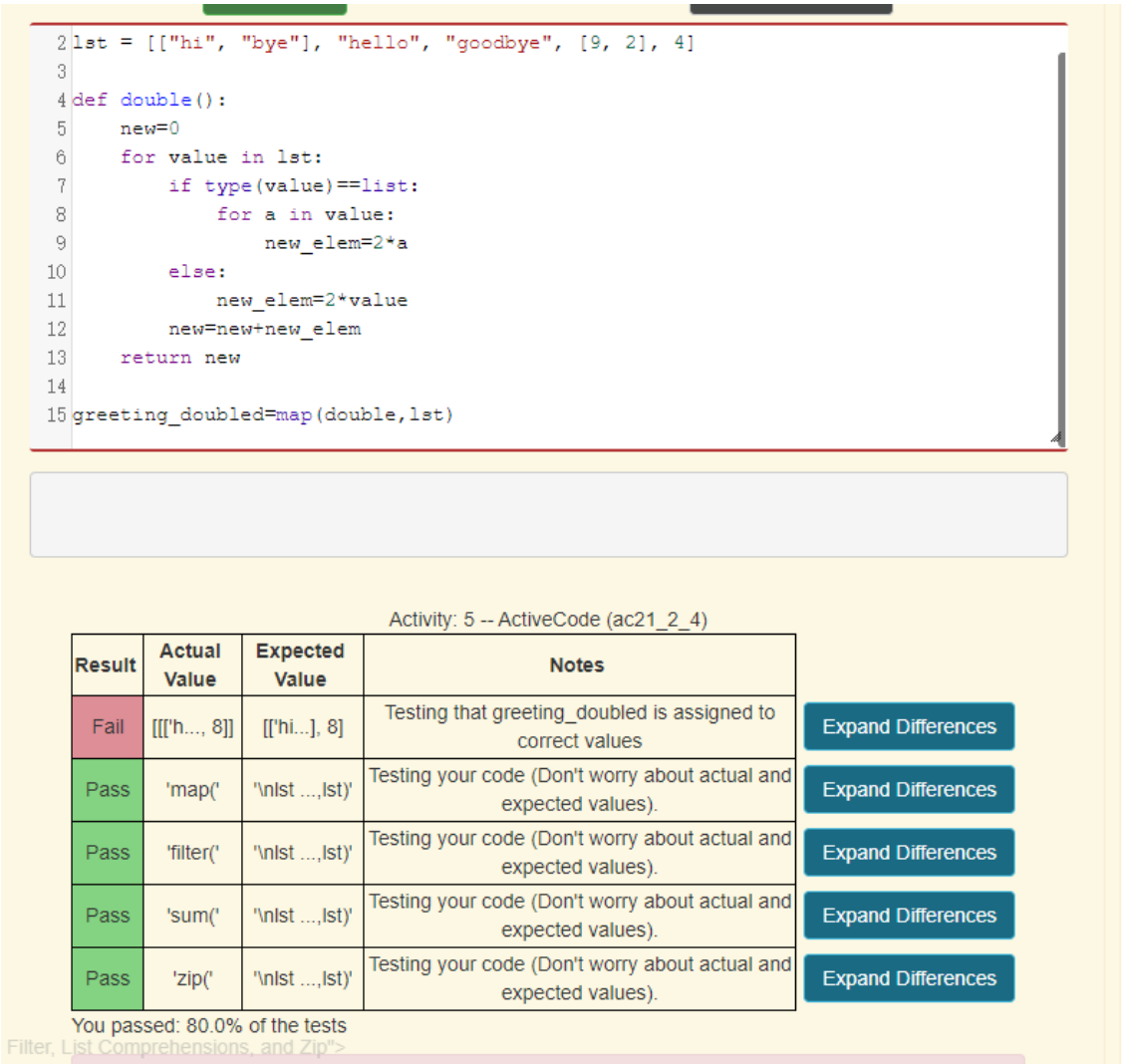
```
def doubleStuff(a_list):
    """ Return a new list in which contains doubles of the elements in a_list. """
    new_list = []
    for value in a_list:
        new_elem = 2 * value
        new_list.append(new_elem)
    return new_list

things = [2, 5, 9]
print(things)
things = doubleStuff(things)
print(things)

```

```
[2, 5, 9]
[4, 10, 18]
```

  - 有个改成大写的例子
    - d



- 有个死活写不出来错的例子
  - 应该是因为list中list



```
lst = [["hi", "bye"], "hello", "goodbye", [9, 2], 4]

def double():
    new=0
    for value in lst:
        if type(value)==list:
            for a in value:
                new_elem=2*a
        else:
            new_elem=2*value
        new=new+new_elem
    return new

greeting_doubled=map(double,lst)
```

Activity: 5 -- ActiveCode (ac21_2_4)

Result	Actual Value	Expected Value	Notes	
Fail	[['h..., 8]]	['hi...], 8]	Testing that greeting_doubled is assigned to correct values	Expand Differences
Pass	'map('	'\nlst ...,lst)'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'filter('	'\nlst ...,lst)'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'sum('	'\nlst ...,lst)'	Testing your code (Don't worry about actual and expected values).	Expand Differences
Pass	'zip('	'\nlst ...,lst)'	Testing your code (Don't worry about actual and expected values).	Expand Differences

You passed: 80.0% of the tests

Filter, List Comprehensions, and Zip">

- Filter
  - 找出最后是偶数的list

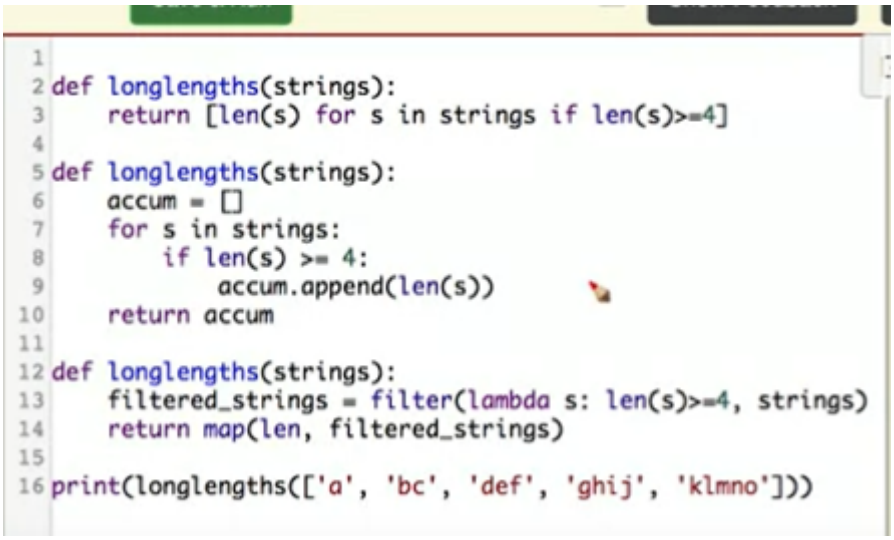
```
def keep_evens(nums):
    new_list = []
    for num in nums:
        if num % 2 == 0:
            new_list.append(num)
    return new_list

print(keep_evens([3, 4, 6, 7, 0, 1]))
```

```
[4, 6, 0]
```

- 或者还有其他的写法
- def keep_evens(nums):
- 	new_seq = filter(lambda num: num % 2 == 0, nums)这里前半部分相当于一个是否题 如果是的话就print往下
- 	return list(new_seq)
- print(keep_evens([3, 4, 6, 7, 0, 1]))

- List Comprehensions
  - 格式 [<transformer_expression> for <loop_var> in <sequence> if <filtration_expression>]
  - 例子
    - things = [2, 5, 9]
    - yourlist = [value * 2 for value in things] 左右create a list 然后里面就是value*2其实就是for value in things都*2 for前面写要做的变换
    - print(yourlist)
    - 这个写法跟 yourlist=map(lambda value:value*2,things)
  - 用filter的也可以改写
    - def keep_evens(nums):
    - 	new_list = [num for num in nums if num % 2 == 0] 第一个num是本来应该填transform的东西 但是也不想tran它 就直接num for num这里是variable in nums是list if是条件后面的东西满足 也就是 filter里的True条件
    - 	return new_list
    - print(keep_evens([3, 4, 6, 7, 0, 1]))
    - 本来的话就是 new_list=filter(lambda num:num % 2==0,nums)

- 也可以合在一起变成 【要转换的方式 for 谁 in 哪里 if 条件】or map(lambda s:len(s),strings)



```
1
2 def longlengths(strings):
3     return [len(s) for s in strings if len(s)>=4]
4
5 def longlengths(strings):
6     accum = []
7     for s in strings:
8         if len(s) >= 4:
9             accum.append(len(s))
10    return accum
11
12 def longlengths(strings):
13    filtered_strings = filter(lambda s: len(s)>=4, strings)
14    return map(len, filtered_strings)
15
16 print(longlengths(['a', 'bc', 'def', 'ghij', 'klmno']))
```

- Zip
  - 把长度相等的list同位点的数字放在一起
  - 看这个
    - L1 = [3, 4, 5]
    - L2 = [1, 2, 3]
    - L4 = list(zip(L1, L2))
    - print(L4)
    - 输出
    - [(3, 1), (4, 2), (5, 3)]
    - 如果想要让上下分别相加的话
    - 用之前学的第一种 最常用最推荐的trans for 谁 in 哪里 if　L3 = [x1 + x2 for (x1, x2) in list(zip(L1, L2))]
    - 第二种 map：L3 = map(lambda x: x[0] + x[1], zip(L1, L2))
    -
  - 例子 猜词hang小人 人吊死了还没猜出来就是死了
    - 之前写法

```python
def possible(word, blanked, guesses_made):
    if len(word) != len(blanked):
        return False
    for i in range(len(word)):
        bc = blanked[i]
        wc = word[i]
        if bc == '_' and wc in guesses_made:
            return False
        elif bc != '_' and bc != wc:
            return False
    return True

print(possible("wonderwall", "_on__r__ll", "otnqurl"))
print(possible("wonderwall", "_on__r__ll", "wotnqurl"))

```

```
True
False
```

- word这个自己想的word

- 用zip改写

```python
def possible(word, blanked, guesses_made):
    if len(word) != len(blanked):
        return False
    for (bc, wc) in zip(blanked, word):
        if bc == '_' and wc in guesses_made:
            return False
        elif bc != '_' and bc != wc:
            return False
    return True

print(possible("wonderwall", "_on__r__ll", "otnqurl"))
print(possible("wonderwall", "_on__r__ll", "wotnqurl"))


```

```
True
False
```

-

```python
def possible(word, blanked, guesses_made):
    if len(word) != len(blanked):
        return False
#   for i in range(len(word)):
#       bc = blanked[i]
#       wc = word[i]
    for (bc, wc) in zip(blanked, word):
        if not compatible_char(bc, wc, guesses_made):
            return False
    return True
```