Task:1

1.1 & 1.2

def myreduce(func,my\_list):

    result = my\_list[0]

    for item in my\_list[1:]:

     result = func(result,item)

    return result

def myfilter(func,my\_list):

    result =[]

    for item in my\_list:

      if func(item):

        result.append(item)

    return result

def sum(x,y):return x+y

def ispositive(x):

    if(x<=0):

        return False

    else:

        return True

    print("Sum on list [1,2,3] using custom reduce function" + str(myreduce(sum,{1,2,3})))

2

word = "ACADGILD"

alphabet\_list = [ alphabet for alphabet in word ]

print ("ACADGILD => " + str(alphabet\_list))

input\_list = ['x','y','z']

result = [ item\*num for item in input\_list for num in range(1,5)  ]

print("['x','y','z'] => " +   str(result))

input\_list = ['x','y','z']

result = [ item\*num for num in range(1,5) for item in input\_list  ]

print("['x','y','z'] => " +   str(result))

input\_list = [2,3,4]

result = [ [item+num] for item in input\_list for num in range(0,3)]

print("[2,3,4] =>" +  str(result))

input\_list = [2,3,4,5]

result = [ [item+num for item in input\_list] for num in range(0,4)  ]

print("[2,3,4,5] =>" +  str(result))

input\_list=[1,2,3]

result = [ (b,a) for a in input\_list for b in input\_list]

print("[1,2,3] =>" +  str(result))

3

def find\_longest\_word(words\_list):

    word\_len = []

    for n in words\_list:

        word\_len.append((len(n), n))

    word\_len.sort()

    return word\_len[-1][1]

print(find\_longest\_word(["PHP", "Exercises", "Backend"]))

Task 2

1.1

a = 5

b = 6

c = 7

s = (a + b + c) / 2

area = (s\*(s-a)\*(s-b)\*(s-c)) \*\* 0.5

print('The area of the triangle is %0.2f' %area)

1.2

def filter\_long\_words(words, n):

    return filter(lambda x: len(x) > n, words)

print(filter\_long\_words(['this', 'words', 'are for testing'], 5))

2.1

def map\_to\_lengths\_for(words):

    lengths = []

    for word in words:

        lengths.append(len(word))

    return lengths

def map\_to\_lengths\_map(words):

    return map(len, words)

def map\_to\_lengths\_lists(words):

    return [len(word) for word in words]

if \_\_name\_\_ == "\_\_main\_\_":

    words = ['abv', 'try me', 'test']

    print map\_to\_lengths\_for(words)

    print map\_to\_lengths\_map(words)

    print map\_to\_lengths\_lists(words)

2.2

def is\_vowel(char):

    all\_vowels = 'aeiou'

    return char in all\_vowels

print(is\_vowel('c'))

print(is\_vowel('e'))