ineuron-assignment-02

March 16, 2020

1 Assignment 02

Date: 02/14/2020

Name: Mukunthan Ragavan

2 Task - 1

2.1 1.1:

Write a Python Program to implement your own myreduce() function which works exactly like Python's built-in function reduce()

[1]:
```
def add (a,b):
    return a+b

def mul (a,b):
    return a*b


def myreduce(func, l1):
        if(len(l1))==0:
            return None
        elif(len(l1))==1:
            return l1[0]
        else:
            result=l1[0]
            for i in range(1,len(l1)):
                result= func(result, l1[i])
        return result
```

[2]:
```
myreduce(add, [1,2,3,4])
```

[2]:
```
10
```

[3]:
```
myreduce(lambda x,y: x+y , [1,2,3,4])
```

[3]: 10

[4]:
```
myreduce(mul, [1,2,3,4])
```

[4]: 24

[5]:
```
myreduce(lambda x,y: x*y , [1,2,3,4])
```

[5]: 24

2.2 1.2:

Write a Python program to implement your own myfilter() function which works exactly like Python's built-in function filter()

[6]:
```
def major(age):
    if age > 18: return True
    else: return False


def myfilter(filterfunc, l2):
    result=[]
    for i in l2:
        status= filterfunc(i)
        if status: result.append(i)
    return result
```

[7]:
```
myfilter(major,[15,25,29,45,12])
```

[7]: [25, 29, 45]

[8]:
```
myfilter(lambda x: x>18,[15,25,29,45,12])
```

[8]: [25, 29, 45]

2.3 2:

Implement List comprehensions to produce the following lists. Write List comprehensions to produce the following Lists

['A', 'C', 'A', 'D', 'G', 'I', 'L', ' D'] — Done

['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz'] —Done

['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz'] —Done

[[2], [3], [4], [3], [4], [5], [4], [5], [6]] – Done

[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]] —Done

[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)] – Done

2.4 Expected List Comprehension output

['A', 'C', 'A', 'D', 'G', 'I', 'L', ' D']

[9]:
```
word_one='acadgild'
l1= [ char.upper() for char in word_one]
l1
```

[9]:
```
['A', 'C', 'A', 'D', 'G', 'I', 'L', 'D']
```

2.5 Expected List Comprehension output

['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']

[10]:
```
word_two='xyz'
```

[11]:
```
# Normal For-loop solution

l2=[]
reps=4
for a  in word_two:
     for i in range(1,reps+1):
            l2.append(a*i)

l2
```

[11]:
```
['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']
```

[12]:
```
#Using List comprehension to get solution
reps=4
l3= [ a*i for a in word_two for i in range(1,reps+1)]
l3
```

[12]:
```
['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']
```

2.6 Expected List Comprehension output

['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz']

[13]:
```
# Normal For-loop solution
l9=[]
for a in range(1,5):
    for b in word_two:
        l9.append(a*b)

l9
```

[13]: ['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz']

[14]:
```
#Using List comprehension to get solution
l10= [ item*num for num in range(1,5) for item in word_two  ]
l10
```

[14]: ['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz']

2.7 Expected List Comprehension output

[[2], [3], [4], [3], [4], [5], [4], [5], [6]]

[15]:
```
# Normal For-loop solution
l6=[]
for i in range(1,4):
    for j in range(1,4):
        t=[]
        t.append(j+i)
        l6.append(t)
l6
```

[15]: [[2], [3], [4], [3], [4], [5], [4], [5], [6]]

[16]:
```
#Using List comprehension to get solution
l7=[ [i+j] for i in range(1,4) for j in range(1,4) ]
l7
```

[16]: [[2], [3], [4], [3], [4], [5], [4], [5], [6]]

2.8 Expected List Comprehension output

[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]

[17]:
```
# Normal For-loop solution
l8=[]
for i in range(1,5):
    t=[]
```

```
    for j in range(1,5):
        t.append(j+i)
    l8.append(t)

l8
```

[17]:
```
[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]
```

[18]:
```
#Using List comprehension to get solution
t=[]
l9=[ [(j+i) for i in range(1,5)] for j in range(1,5)]
l9
```

[18]:
```
[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]
```

2.9 Expected List Comprehension output

[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]

[19]:
```
# Normal For-loop solution
l4=[]
for t in range(1,4):
    for v in range(1,4):
        ele=v,t
        l4.append(ele)
l4
```

[19]:
```
[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]
```

[20]:
```
#Using List comprehension to get solution
l5= [(v,t) for t in range(1,4)for v in range(1,4) ]
l5
```

[20]:
```
[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]
```

2.10 3:

Implement a function longestWord() that takes a list of words and returns the longest one.

[21]:
```
from typing import List

def longestWord(word_list:List[str])->str:
    word_dict= { word :len(word) for word in word_list if type(word)==str}
    max_value = max(word_dict.values())
```

```
    list_of_Keys = [key  for (key, value) in word_dict.items() if value ==␣
↪max(word_dict.values())]
    return list_of_Keys[0]
```

[22]:
```
longestWord(['abb', 'ghkhkjhk', 'g25hkjhk','hjghkjkhghjjjkjlk'])
```

[22]:
```
'hjghkjkhghjjjkjlk'
```

3 Task - 2

3.1 1.1

Write a Python Program(with class concepts) to find the area of the triangle using the below formula. area = (s*(s-a)*(s-b)*(s-c)) ** 0.5 Function to take the length of the sides of triangle from user should be defined in the parent class and function to calculate the area should be defined in subclass.

[23]:
```
import math
class Userprompt:

    def __init__(self,a,b,c):
        self.a = a
        self.b = b
        self.c = c

    @property
    def get_traingle_s(self):
        return (self.a+self.b+self.c)/2

class Traingle(Userprompt):
    def __init(self,a,b,c):
        super().__init__(a,b,c)

    @property
    def get_area(self):
        self.s = super().get_traingle_s
        self.area=math.sqrt(self.s*(self.s-self.a)*(self.s-self.b)*(self.s-self.
↪c))
        return self.area
```

[24]:
```
t1 = Traingle(2,3,3)
```

[25]:
```
t1.get_area
```

[25]:
```
2.8284271247461903
```

[26]:
```
t1.get_traingle_s
```

[26]:
```
4.0
```

3.2 1.2

Write a function filter_long_words() that takes a list of words and an integer n and returns the list of words that are longer than n.

[27]:
```
from typing import List

def filter_long_words(wlist:List[str], limit:int) ->List[str]:
    final=[word for word in wlist if len(word) > limit]
    return final
```

[28]:
```
user_list = 'This assignment will help you to consolidate the concepts learnt␣
 ↪in the session'.split()
user_list
```

[28]:
```
['This',
 'assignment',
 'will',
 'help',
 'you',
 'to',
 'consolidate',
 'the',
 'concepts',
 'learnt',
 'in',
 'the',
 'session']
```

[29]:
```
filter_long_words(user_list, 4)
```

[29]:
```
['assignment', 'consolidate', 'concepts', 'learnt', 'session']
```

3.3 2.1

Write a Python program using function concept that maps list of words into a list of integers representing the lengths of the corresponding words. Hint: If a list [ab,cde,erty] is passed on to the python function output should come as [2,3,4] Here 2,3 and 4 are the lengths of the words in the list.

[30]:
```
from typing import List

l11=[ 'ab','cde','erty','mukunthan ragavan']


def get_len_wordlist(wlist:List[str])->List[str]:
    len_word = map(lambda x: len(x), wlist)
    return list(len_word)
```

[31]:
```
get_len_wordlist(l11)
```

[31]:
```
[2, 3, 4, 17]
```

3.4 2.2

Write a Python function which takes a character (i.e. a string of length 1) and returns True if it is a vowel, False otherwise.

[32]:
```
def is_vowel(char:str)->bool:
    if(len(char)>1):
        print('Entered more than on character, however first letter will be␣
 ↪considered.')
    if char[0].lower() in (list('aeiou')):
        print('Given character {} is vowel'.format(char[0]))
        return True
    else:
        print('Given character {} is not vowel'.format(char[0]))
        return False
```

[33]:
```
is_vowel('e')
```

```
Given character e is vowel
```

[33]:
```
True
```

[34]:
```
is_vowel('E')
```

```
Given character E is vowel
```

[34]:
```
True
```

[35]:
```
is_vowel('english')
```

```
Entered more than on character, however first letter will be considered.
Given character e is vowel
```

[35]: True

[36]:
```
is_vowel('T')
```

```
Given character T is not vowel
```

[36]: False

[37]:
```
is_vowel('$')
```

```
Given character $ is not vowel
```

[37]: False