

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

All Medium tasks

Python

28/115 challenges solved
Rank: 58935 | Points: 930



Write a function

Medium, Python (Basic), Max Score: 10, Success Rate: 90.33%



Solved

The Minion Game

Medium, Python (Basic), Max Score: 40, Success Rate: 86.80%



Solved

Merge the Tools!

Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.75%



Solved

Time Delta

Medium, Python (Basic), Max Score: 30, Success Rate: 91.36%



Solved

Find Angle MBC

Medium, Python (Basic), Max Score: 10, Success Rate: 89.15%



Solved

No Idea!

Medium, Python (Basic), Max Score: 50, Success Rate: 88.01%



Solved

Word Order

Medium, Python (Basic), Max Score: 50, Success Rate: 90.23%



Solved

Compress the String!

Medium, Python (Basic), Max Score: 20, Success Rate: 97.15%



Solved

Company Logo

Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.83%



Solved

Piling Up!

Medium, Python (Basic), Max Score: 50, Success Rate: 90.64%



Solved



ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

1/2/24, 10:14 PM

Solve Python | HackerRank

Iterables and Iterators

Medium, Python (Basic), Max Score: 40, Success Rate: 96.62%



Solved

Triangle Quest

Medium, Python (Basic), Max Score: 20, Success Rate: 93.84%



Solved

Classes: Dealing with Complex Numbers

Medium, Python (Basic), Max Score: 20, Success Rate: 90.92%



Solved

Athlete Sort

Medium, Python (Basic), Max Score: 30, Success Rate: 95.53%



Solved

ginortS

Medium, Python (Basic), Max Score: 40, Success Rate: 97.63%



Solved

Validating Email Addresses With a Filter

Medium, Python (Basic), Max Score: 20, Success Rate: 90.82%



Solved

Reduce Function

Medium, Max Score: 30, Success Rate: 98.38%



Solved

Regex Substitution

Medium, Python (Basic), Max Score: 20, Success Rate: 94.11%



Solved

Validating Credit Card Numbers

Medium, Python (Basic), Max Score: 40, Success Rate: 95.46%



Solved

Words Score

Medium, Max Score: 10, Success Rate: 94.94%



Solved

Default Arguments

Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.82%



Solved

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

1. Write a function

```
1 def is_leap(year):
2     leap = False
3     if (year % 400 == 0):
4         return True
5     if (year % 100 == 0):
6         return leap
7     if (year % 4 == 0):
8         return True
9     else:
10        return False
11    # Write your logic here
12
13    return leap
14
15 ✓ year = int(input())
16 print(is_leap(year))
```

write a function

2. Game of minion

```
1 def minion_game(string):
2     vowels = "AEIOU"
3     length = len(string)
4     kevin_score = 0
5     stuart_score = 0
6
7     for i in range(length):
8         if string[i] in vowels:
9             kevin_score += length - i
10        else:
11            stuart_score += length - i
12
13    # Determine the winner
14    if kevin_score > stuart_score:
15        print("Kevin", kevin_score)
16    elif stuart_score > kevin_score:
17        print("Stuart", stuart_score)
18    else:
19        print("Draw")
20
21
22
23 ✓ if __name__ == '__main__':
24     s = input()
25     minion_game(s)
```

the minion game

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

3. Merging Tool

```
1 def merge_the_tools(string, k):
2     # your code goes here
3     temp = []
4     len_temp = 0
5     for item in string:
6         len_temp += 1
7         if item not in temp:
8             temp.append(item)
9         if len_temp == k:
10            print(''.join(temp))
11            temp = []
12            len_temp = 0
13 if __name__ == '__main__':
14     string, k = input(), int(input())
15     merge_the_tools(string, k)
```

merge the tool

4. Time Delta function

```
1  #!/bin/python3
2
3  from datetime import datetime, timedelta
4
5  def time_delta(t1, t2):
6      # Define the format of the timestamp
7      fmt = "%a %d %b %Y %H:%M:%S %z"
8
9      # Parse the timestamps using the defined format
10     dt1 = datetime.strptime(t1, fmt)
11     dt2 = datetime.strptime(t2, fmt)
12
13     # Calculate the absolute difference in seconds
14     diff_seconds = int(abs((dt1 - dt2).total_seconds()))
15
16     return diff_seconds
17
18 # Number of test cases
19 t = int(input().strip())
20
21 for _ in range(t):
22     # Read the timestamps
23     time1 = input().strip()
24     time2 = input().strip()
25
26     # Calculate and print the absolute difference in seconds
27     result = time_delta(time1, time2)
28     print(result)
29
```

Time Delta

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

5. Find angle of MBC

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 import math
3 ab=int(input())
4 bc=int(input())
5 ca=math.hypot(ab,bc)
6 mc=ca/2
7 bca=math.asin(1*ab/ca)
8 bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
9 mbc=math.asin(math.sin(bca)*mc/bm)
10 print(int(round(math.degrees(mbc),0)),'\u00B0',sep='')
11
```

Find angle MBC

6. None idea

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 # Read input values
3 n, m = map(int, input().split())
4 arr = list(map(int, input().split()))
5 set_a = set(map(int, input().split()))
6 set_b = set(map(int, input().split()))
7
8 # Calculate happiness
9 happiness = 0
10
11 for num in arr:
12     if num in set_a:
13         happiness += 1
14     elif num in set_b:
15         happiness -= 1
16
17 # Print the final happiness
18 print(happiness)
19
```

No idea

7. Correct word order

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

```
1  # Enter your code here. Read input from STDIN. Print output to STDOUT
2  from collections import OrderedDict
3
4  def word_count(words):
5      word_dict = OrderedDict()
6
7      for word in words:
8          # If the word is not in the dictionary, add it with count 1
9          if word not in word_dict:
10             word_dict[word] = 1
11          else:
12             # If the word is already in the dictionary, increment its count
13             word_dict[word] += 1
14
15      return word_dict
16
17 # Read input
18 n = int(input())
19 word_list = [input().strip() for _ in range(n)]
20
21 # Count occurrences
22 word_counts = word_count(word_list)
23
24 # Output the results
25 print(len(word_counts))
26 print(*word_counts.values())
27
```

Word order

8. String Compression

```
1  # Enter your code here. Read input from STDIN. Print output to STDOUT
2  # Enter your code here. Read input from STDIN. Print output to STDOUT
3  from itertools import groupby
4  for k, c in groupby(input()):
5      print("%d, %d" % (len(list(c)), int(k)), end=' ')
6
```

Compress the string

9. Company logo

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8
9  from collections import Counter
10
11 S = input()
12 S = sorted(S)
13 FREQUENCY = Counter(list(S))
14 for k, v in FREQUENCY.most_common(3):
15     print(k, v)
16
17
```

Company logo

10. Piling up

```
1  # Enter your code here. Read input from STDIN. Print output to STDOUT
2  def can_stack_cubes(test_cases):
3      for cubes in test_cases:
4          n = cubes[0]
5          side_lengths = cubes[1]
6
7          left = 0
8          right = n - 1
9          prev_cube = float('inf')
10
11         while left <= right:
12             # Choose the larger cube from the left or right end
13             current_cube = max(side_lengths[left], side_lengths[right])
14
15             # Check if it's not possible to stack the cubes
16             if current_cube > prev_cube:
17                 print("No")
18                 break
19
20             # Update previous cube and adjust pointers
21             prev_cube = current_cube
22             if side_lengths[left] >= side_lengths[right]:
23                 left += 1
24             else:
25                 right -= 1
26
27         else:
28             # If the loop completes without a break, print "Yes"
29             print("Yes")
30
31     # Read the number of test cases
32     t = int(input().strip())
33
34     # Read and store test cases
35     test_cases = []
36     for _ in range(t):
37         _ = int(input().strip()) # ignoring the number of cubes
38         side_lengths = list(map(int, input().split()))
39         test_cases.append(('_', side_lengths))
40
41     # Check if it's possible to stack cubes for each test case
42     can_stack_cubes(test_cases)
43
```

Piling Up

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

11. Triangular quest 2

```
1 ✓ for i in range(1, int(input())+1):  
2     print(((10**i)//9)**2)  
3
```

Triangular quest2

12. Iterables & Iterators

```
1  
2 # Enter your code here. Read input from STDIN. Print output to STDOUT  
3 from itertools import combinations  
4  
5 N = int(input())  
6 LETTERS = list(input().split(" "))  
7 K = int(input())  
8  
9 TUPLES = list(combinations(LETTERS, K))  
10 CONTAINS = [word for word in TUPLES if "a" in word]  
11  
12 print(len(CONTAINS)/len(TUPLES))  
13
```

iterables and iterators

13. Triangular quest 1

```
1 ✓ for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank line also  
2     print((10**(i)//9)*i)  
3
```

Triangle quest

14. Classes: Complex Numbers Handling

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

```
1 import math
2
3 class Complex(object):
4     def __init__(self, real, imaginary):
5         self.real = real
6         self.imaginary = imaginary
7
8     def __add__(self, no):
9         return Complex((self.real+no.real), self.imaginary+no.imaginary)
10
11     def __sub__(self, no):
12         return Complex((self.real-no.real), (self.imaginary-no.imaginary))
13
14     def __mul__(self, no):
15         r = (self.real*no.real)-(self.imaginary*no.imaginary)
16         i = (self.real*no.imaginary+no.real*self.imaginary)
17         return Complex(r, i)
18
19     def __truediv__(self, no):
20         conjugate = Complex(no.real, (-no.imaginary))
21         num = self*conjugate
22         denom = no*conjugate
23         try:
24             return Complex((num.real/denom.real), (num.imaginary/denom.real))
25         except Exception as e:
26             print(e)
27
28     def mod(self):
29         m = math.sqrt(self.real**2+self.imaginary**2)
30         return Complex(m, 0)
31
32     def __str__(self):
33         if self.imaginary == 0:
34             result = "%.2f+0.00i" % (self.real)
35         elif self.real == 0:
36             if self.imaginary >= 0:
37                 result = "0.00+%.2fi" % (self.imaginary)
38             else:
39                 result = "0.00-%.2fi" % (abs(self.imaginary))
40         elif self.imaginary > 0:
41             result = "%.2f+%.2fi" % (self.real, self.imaginary)
42         else:
43             result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))
44         return result
45
46 if __name__ == '__main__':
47     c = map(float, input().split())
48     d = map(float, input().split())
49     x = Complex(*c)
50     y = Complex(*d)
51     print(*map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]), sep='\n')]
```

Classes:dealing with complex number

Activate Window
Go to Settings to activate

15. Athelete sorting

```
1 #!/bin/python3
2
3 import math
4 import os
5 import random
6 import re
7 import sys
8
9 # Read the first input for rows and columns
10 n, m = map(int, input().split())
11
12 # Read the matrix of numbers
13 rows = [list(map(int, input().split())) for _ in range(n)]
14
15 # Read the index for sorting
16 k = int(input())
17
18 # Sort rows based on the k-th column
19 for row in sorted(rows, key=lambda x: x[k]):
20     print(' '.join(map(str, row)))
21
```

Athelete sort

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

16. Ginortx

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 def custom_sort(c):
3     if c.islower():
4         return (0, c)
5     elif c.isupper():
6         return (1, c)
7     elif c.isdigit() and int(c) % 2 != 0:
8         return (2, c)
9     elif c.isdigit() and int(c) % 2 == 0:
10        return (3, c)
11
12 def sort_string(s):
13     sorted_str = ''.join(sorted(s, key=custom_sort))
14     return sorted_str
15
16 # Read input
17 s = input().strip()
18
19 # Output the sorted string
20 result = sort_string(s)
21 print(result)
22
```

ginortS

17. Validating Email address via filter

```
1 def fun(email):
2     try:
3         username, url = email.split('@')
4         website, extension = url.split('.')
5     except ValueError:
6         return False
7     if username.replace('-', '').replace('_', '').isalnum() is False:
8         return False
9     elif website.isalnum() is False:
10        return False
11    elif len(extension) > 3:
12        return False
13    else:
14        return True
15
16 def filter_mail(emails):
17     return list(filter(fun, emails))
18
19 > def filter_mail(emails):---
```

VALIDITY EMAIL ADDRESS

18. Reduction function

```
1 > from fractions import Fraction---
2
3 def product(fracs):
4     t = Fraction(reduce(lambda x, y: x * y, fracs))# complete this line with a reduce statement
5     return t.numerator, t.denominator
6
7
8 if __name__ == '__main__':
9     fracs = []
10    for _ in range(int(input())):
11        fracs.append(Fraction(*map(int, input().split())))
12    result = product(fracs)
13    print(*result)
```

Reduced function

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

19. Regex substitution

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2
3 import re
4 for _ in range(int(input())):
5     print(re.sub(r'(?<= )(&&|\\|\\|)(?= )', lambda x: 'and' if x.group() == '&&' else 'or', input()))
6
```

Regex substitution

20. Validating the Credit card number

```
1
2 # Enter your code here. Read input from STDIN. Print output to STDOUT
3 import re
4 n = int(input())
5 for t in range(n):
6     credit = input().strip()
7     credit_removed_hiphen = credit.replace('-', '')
8     valid = True
9     length_16 = bool(re.match(r'^[4-6]\d{15}$', credit))
10    length_19 = bool(re.match(r'^[4-6]\d{3}-\d{4}-\d{4}-\d{4}$', credit))
11    consecutive = bool(re.findall(r'(?=\d)\1\1\1', credit_removed_hiphen))
12    if length_16 == True or length_19 == True:
13        if consecutive == True:
14            valid = False
15        else:
16            valid = True
17        if valid == True:
18            print('Valid')
19    else:
20        print('Invalid')
21
```

Validating credit card numbers

21. Word score

```
1 def is_vowel(letter):
2     return letter in ['a', 'e', 'i', 'o', 'u', 'y']
3 def is_vowel(letter):
4     return letter in ['a', 'e', 'i', 'o', 'u', 'y']
5 def score_words(words):
6     score = 0
7     for word in words:
8         num_vowels = 0
9         for letter in word:
10             if is_vowel(letter):
11                 num_vowels += 1
12             if num_vowels % 2 == 0:
13                 score += 2
14             else:
15                 score += 1
16     return score
17
18 > ...
```

Word Score

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

22. Default argument

```
1 > class EvenStream(object): ...
18
19 def print_from_stream(n, stream=EvenStream()):
20     stream.__init__()
21     for _ in range(n):
22         print(stream.get_next())
23
24
25 queries = int(input())
26 for _ in range(queries):
27     stream_name, n = input().split()
28     n = int(n)
29     if stream_name == "even":
30         print_from_stream(n)
31     else:
32         print_from_stream(n, OddStream())
33
```

Default argument

Some hard task

Python

28/115 challenges solved
Rank: 58928 | Points: 930



Maximize It!

Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.25%



Solved

Validating Postal Codes

Hard, Max Score: 80, Success Rate: 87.39%



Solved

Matrix Script

Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.97%



Solved

1. Maximization

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
2 # Enter your code here. Read input from STDIN. Print output to STDOUT
3 import itertools
4
5 NUMBER_OF_LISTS, MODULUS = map(int, input().split())
6 LISTS_OF_LISTS = []
7
8 for i in range(0, NUMBER_OF_LISTS):
9     new_list = list(map(int, input().split()))
10    del new_list[0]
11    LISTS_OF_LISTS.append(new_list)
12
13 def squared(element):
14     return element**2
15
16 COMBS = list(itertools.product(*LISTS_OF_LISTS))
17 RESULTS = []
18
19 for i in COMBS:
20     result1 = sum(map(squared, [a for a in i]))
21     result2 = result1 % MODULUS
22     RESULTS.append(result2)
23
24 print(max(RESULTS))
```

Maximize it

2. Validation of postal codes

```
1 regex_integer_in_range = r"^[1-9][\d]{5}$" # Do not delete 'r'.
2 regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"
3
4 import re
5 P = input()
6
7 print(bool(re.match(regex_integer_in_range, P))
8 and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

Validity Postal codes

3. Matrix script function

ASSIGNMENT # 3

AI - USMAN MUJTABA_399619

```
1  #!/bin/python3
2
3  import math
4  import os
5  import random
6  import re
7  import sys
8  import re
9  n, m = map(int,input().split())
10 character_ar = [''] * (n*m)
11 for i in range(n):
12     line = input()
13     for j in range(m):
14         character_ar[i+(j*n)]=line[j]
15 decoded_str = ''.join(character_ar)
16 final_decoded_str = re.sub(r'(?<=[A-Za-z0-9])([ !@#%&]+)(?=[A-Za-z0-9])',' ',decoded_str)
17 print(final_decoded_str)
18
19
20
21 first_multiple_input = input().rstrip().split()
22
23 n = int(first_multiple_input[0])
24
25 m = int(first_multiple_input[1])
26
27 matrix = []
28
29 for _ in range(n):
30     matrix_item = input()
31     matrix.append(matrix_item)
32
```

Matrix script