مسائل تمرینی توابع

مسائل به ترتیب افزایش درجه سختی مرتب شده اند:

- دست گرمی - اینها می توانند با استفاده از مقایسه های پایه و روش های ساده حل شوند
- سطح 1 - اینها ممکن است شامل عبارات شرطی if / then و روش های ساده باشد
- سطح 2 - اینها ممکن است نیاز به تکرار بر روی دنباله ها داشته باشد ، معمولاً با نوعی حلقه
- چالش برانگیز - حل این مسائل نیاز به خلاقیت دارد

دست گرمی:

کمتر از دو زوج: تابعی بنویسید که از میان دو عدد داده را بعنوان ورودی دریافت می کند و * اگر * هر دو عدد زوج باشند مقدار کوچکتر را بر میگرداند ، اما اگر حداقل یکی از اعداد فرد باشد مقدار بزرگتر را برمی گرداند

```
lesser_of_two_evens (2،4) -> 2
lesser_of_two_evens (2،5) -> 5
```

In [1]:

```
def lesser_of_two_evens(a, b):
    if a % 2 == 0 and b % 2 == 0:
        return min(a, b)
    else:
        return max(a, b)
```

In [2]:

```
# Check
lesser_of_two_evens(2, 4)
```

Out[2]:

```
2
```

In [3]:

```
# Check
lesser_of_two_evens(2, 5)
```

Out[3]:

```
5
```

ANIMAL CRACKERS: تابعی بنویسید که یک رشته دو کلمه ایی از اسامی حیوانات را بعنوان ورودی دریافت کند و اگر حرف اول دو کلمات داخل رشته با یکدیگر برابر باشد، مقدار `True` را برمیگرداند

```
animal_crackers('Levelheaded Llama') --> True
animal_crackers('Crazy Kangaroo') --> False
```

In [7]:

```
def animal_crackers(text):
    wordlist = text.split()
    return wordlist[0][0].lower() == wordlist[1][0].lower()
```

In [8]:

```
# Check
animal_crackers('Levelheaded Llama')
```

Out[8]:

```
True
```

In [9]:

```
# Check
animal_crackers('Crazy Kangaroo')
```

Out[9]:

```
False
```

MAKES TWENTY: تابعی بنویسید که دو عدد را بعنوان ورودی دریافت می کند و اگر یکی از اعداد 20 باشد یا مجموعه دو عدد 20 باشد مقدار `True` را برگرداند.

```
makes_twenty(20,10) --> True
makes_twenty(12,8) --> True
makes_twenty(2,3) --> False
```

In [10]:

```
def makes_twenty(a, b):
    return a == 20 or b == 20 or (a + b) == 20
```

In [11]:

```
# Check
makes_twenty(20,10)
```

Out[11]:

```
True
```

In [12]:

```
# Check
makes_twenty(12,8)
```

Out[12]:

```
True
```

In [13]:

```
#Check
makes_twenty(2,3)
```

Out[13]:

```
False
```

مسائل سطح 1

OLD MACDONALD: تابعی بنویسید تا یک رشته را بعنوان ورودی دریافت کند و اگر طول رشته بزرگتر از 3 باشد حرف اول و چهارم را با حروف بزرگ انگلیسی بنویسد

```
old_macdonald('macdonald') --> MacDonald

'Note: 'macdonald'.capitalize() returns 'Macdonald
```

In [18]:

```
def old_macdonald(name):
    if len(name) > 3:
        return name[:3].capitalize() +  name[3:].capitalize()
    else:
        return 'Name is too short'
```

In [19]:

```
# Check
old_macdonald('macdonald')
```

Out[19]:

```
'MacDonald'
```

In [20]:

```
old_macdonald('ma')
```

Out[20]:

```
'Name is too short'
```

MASTER YODA: تابعی بنویسید تا یک رشته را دریافت نمایید و ترتیب کلمات را در آن مطابق مثال های زیر معکوس نماید.

```
master_yoda('I am home') --> 'home am I'
master_yoda('We are ready') --> 'ready are We'
```

In [25]:

```
def master_yoda(text):
    return ' '.join(text.split()[::-1])
```

In [26]:

```
# Check
master_yoda('I am home')
```

Out[26]:

```
'home am I'
```

In [28]:

```
# Check
master_yoda('We are ready')
```

Out[28]:

```
'ready are We'
```

ALMOST THERE: تابع بنویسید که یک عدد را دریافت کند و اگر در فاصله 10 تا از 100 یا 200 باشد `True` و در غیر اینصورت `False` را برگرداند

```
almost_there(90) --> True
almost_there(104) --> True
almost_there(150) --> False
almost_there(209) --> True
```

NOTE: `abs(num)` returns the absolute value of a number

In [29]:

```
def almost_there(num):
    return ((abs(100 - num) <= 10) or (abs(200 - num) <= 10))
```

In [30]:

```
# Check
almost_there(90)
```

Out[30]:

```
True
```

In [31]:

```
# Check
almost_there(104)
```

Out[31]:

```
True
```

In [32]:

```
# Check
almost_there(150)
```

Out[32]:

```
False
```

In [33]:

```
# Check
almost_there(209)
```

Out[33]:

```
True
```

مسائل سطح 2

FIND 33:

تابعی بنویسید که لیستی از اعداد صحیح را دریافت نماید و اگر دو عضو با مقدار 3 پشت سرهم پیدا نماید مقدار `True` را برگرداند و در غیر اینصورت مقدار `False`

```
has_33([1, 3, 3]) → True
has_33([1, 3, 1, 3]) → False
has_33([3, 1, 3]) → False
```

In [35]:

```
def has_33(nums):
    for i in range(0, len(nums) -1 ):

        if nums[i:i+2] == [3,3]:
            return True

    return False
```

In [36]:

```
# Check
has_33([1, 3, 3])
```

Out[36]:

```
True
```

In [37]:

```
# Check
has_33([1, 3, 1, 3])
```

Out[37]:

```
False
```

In [38]:

```
# Check
has_33([3, 1, 3])
```

Out[38]:

```
False
```

PAPER DOLL: تابعی بنویسید که یک رشته را بعنوان ورودی دریافت نماید و یک رشته جدید بعنوان خروجی برگرداند که در آن به ازای هر حرف در رشته اصلی سه بار در رشته جدید تکرار شده باشد.

```
paper_doll('Hello') --> 'HHHeeellllllooo'
paper_doll('Mississippi') --> 'MMMiiissssssiiipppppppiii'
```

In [39]:

```
def paper_doll(text):
    result = ''

    for char in text:
        result += char * 3
    return result
```

In [40]:

```
# Check
paper_doll('Hello')
```

Out[40]:

```
'HHHeeellllllooo'
```

In [41]:

```
# Check
paper_doll('Mississippi')
```

Out[41]:

```
'MMMiiissssssiiissssssiiipppppppiii'
```

BLACKJACK: تابعی بنویسید که سه عدد صحیح در بازه 1 تا 11 دریافت کند در صورتیکه مجموعه اعداد کمتر مساوی 21 باشد مقدار مجموع را برگرداند، در صورتیکه مجموعه بیشتر از 21 باشد، ؛ مجموع را منهای 10 کند اگر مقدار جدید کمتر از 21

باشد و در میان اعداد 11 باشد، مجموع جدید آن را گزارش کند و در غیراینصورت عبارت Bust را چاپ کند.

```
blackjack(5,6,7) --> 18
blackjack(9,9,9) --> 'BUST'
blackjack(9,9,11) --> 19
```

In [43]:

```
def blackjack(a, b, c):

    if sum((a, b, c)) <= 21:
        return sum((a, b, c))
    elif sum((a, b, c)) <= 31 and 11 in (a, b, c):
        return sum((a, b, c)) - 10
    else:
        return 'BUST'
```

In [44]:

```
# Check
blackjack(5,6,7)
```

Out[44]:

18

In [45]:

```
# Check
blackjack(9,9,9)
```

Out[45]:

'BUST'

In [46]:

```
# Check
blackjack(9,9,11)
```

Out[46]:

19

SUMMER OF '69: تابعی بنویسید که یک لیست از اعداد صحیح را بعنوان ورودی دریافت کند و مجموع اعداد را با در نظر گرفتن شرایط مقابل محاسبه نماید، اگر در هنگام محاسبه مجموع به عضوی در لیست با مقدار 6 برسیم خود آن عضو و اعضای بعدی را رسید به عضوی که مقدار آن 9 است در مجموع لحاظ نخواهیم کرد. (در لیست ورودی حتما پس از 6 باید حداقل یک 9 بیاید)

```
summer_69([1, 3, 5]) --> 9
summer_69([4, 5, 6, 7, 8, 9]) --> 9
summer_69([2, 1, 6, 9, 11]) --> 14
```

In [32]:

```
def summer_69(lst):

    total = 0
    add = True

    for num in lst:
        while add:
            if num != 6:
                total += num
                break
            else:
                add = False

        while not add:
            if num != 9:
                break
            else:
                add = True
                break
    return total
```

In [33]:

```
# Check
summer_69([1, 3, 5])
```

Out[33]:

```
9
```

In [34]:

```
# Check
summer_69([4, 5, 6, 7, 8, 9])
```

Out[34]:

```
9
```

In [35]:

```
# Check
summer_69([2, 1, 6, 9, 11])
```

Out[35]:

```
14
```

مسائل چالش برانگیز

SPY GAME: تابعی بنویسید تا لیست از اعداد صحیح را بعنوان ورودی دریافت نماید و اگر در میان اعضا بتوان سه عضو پشت سرهم یا با فاصله پیدا نماید که الگوی `007` را ایجاد کند مقدار `True` را برگرداند در غیر اینصورت `False` را

برگرداند

```
spy_game([1,2,4,0,0,7,5]) --> True
spy_game([1,0,2,4,0,5,7]) --> True
spy_game([1,7,2,0,4,5,0]) --> False
```

In [47]:

```
def spy_game(nums):

    code = [0, 0, 7, 'x']

    for num in nums:
        if num == code[0]:
            code.pop(0)
    return len(code) == 1
```

In [48]:

```
# Check
spy_game([1,2,4,0,0,7,5])
```

Out[48]:

```
True
```

In [49]:

```
# Check
spy_game([1,0,2,4,0,5,7])
```

Out[49]:

```
True
```

In [50]:

```
# Check
spy_game([1,7,2,0,4,5,0])
```

Out[50]:

```
False
```

COUNT PRIMES: تابعی بنویسید که یک عدد صحیح را بعنوان ورودی دریافت کند و لیست اعداد اول تا خود عدد ورودی را چاپ وتعداد آنها را چاپ کند

```
count_primes(100) --> 25
```

.By convention, 0 and 1 are not prime

In [51]:

```
def count_primes(num):
    primes = [2]
    x = 3
    if num < 2:
        return 0
    while x <= num:
        for y in range(3, x, 2):
            if x % y == 0:
                x += 2
                break
        else:
            primes.append(x) #primes => [2, 3, 5, 7]
            x += 2
    print(primes)
    return len(primes)

```

In [52]:

```
# Check
count_primes(100)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 7
1, 73, 79, 83, 89, 97]
```

Out[52]:

```
25
```

جایزه: راه حل سریعتر که اعداد اول را در طول مسیر پیمایش جمع آوری می نماید!

In [53]:

```
def count_primes2(num):
    primes = [2]
    x = 3
    if num < 2:
        return 0
    while x <= num:
        for y in primes:  # use the primes list!
            if x%y == 0:
                x += 2
                break
        else:
            primes.append(x)
            x += 2
    print(primes)
    return len(primes)
```

In [43]:

```
count_primes2(100)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 7
1, 73, 79, 83, 89, 97]
```

Out[43]:

25

صرفا جهت فان :)

PRINT BIG: تابعی بنویسید که یک حرف انگلیسی را دریافت کند و آن را با حرف بزرگ و با استفاده از علامت های * چاپ کند.

```
print_big('a')

out:   *
      * *
     *****
     *   *
     *   *
```

"راهنما: در نظر داشته باشید که یک فرهنگ لغت از الگوهای ممکن ایجاد کنید و حروف الفبا را به ترکیبات خاص 5 خطی از الگوهای مرتبط نگاشت کنید. برای اهداف این تمرین، اگر فرهنگ لغت شما در "E" متوقف شود، مشکلی نیست."

In [67]:

```
def print_big(letter):
    patterns = {1:'  *  ',2:' * * ',3:'*   *',4:'*****',5:'**** ',6:'   * ',7:' *
    alphabet = {'A':[1,2,4,3,3],'B':[5,3,5,3,5],'C':[4,9,9,9,4],'D':[5,3,3,3,5],'E':
    for pattern in alphabet[letter.upper()]:
        print(patterns[pattern])
```

In [70]:

```
print_big('c')
```

```
*****
*
*
*
*****
```

عالی!