Submitted By - Kirti

Function Practice Exercises

Problems are arranged in increasing difficulty:

- Warmup - these can be solved using basic comparisons and methods
- Level 1 - these may involve if/then conditional statements and simple methods
- Level 2 - these may require iterating over sequences, usually with some kind of loop
- Challenging - these will take some creativity to solve

WARMUP SECTION:

LESSER OF TWO EVENS: Write a function that returns the lesser of two given numbers *if* both numbers are even, but returns the greater if one or both numbers are odd

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

In [ ]:

```
def lesser_of_two_evens(a,b):
    if a%2==0 and b%2==0:
        if a<b:
            print(a)
        else:
            print(b)
    else:
        if a>b:
            print(a)
        else:
            print(b)
```

In [ ]:

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

```
2
```

In [ ]:

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

```
5
```

ANIMAL CRACKERS: Write a function takes a two-word string and returns True if both words begin with same letter

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

In [ ]:

```
def animal_crackers(text):
    l=text.split()
    if l[0][0]==l[1][0]:
        print("True")
    else:
        print("False")
```

In [ ]:

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

```
True
```

In [ ]:

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

```
False
```

MAKES TWENTY: Given two integers, return True if the sum of the integers is 20 *or* if one of the integers is 20. If not, return False

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

In [ ]:

```
def makes_twenty(n1,n2):
    if n1==20 or n2==20 or n1+n2==20:
        return True
    else:
        return False
```

In [ ]:

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

Out[ ]:

```
True
```

In [ ]:

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

Out[ ]:

```
False
```

LEVEL 1 PROBLEMS

OLD MACDONALD: Write a function that capitalizes the first and fourth letters of a name

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

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

In [ ]:

```
def old_macdonald(name):
    str=""
    for i in range(0,len(name)):
        if i==0 or i==3:
            s=name[i].upper()
            str+=s
        else:
            str+=name[i]
    print(str)
```

In [ ]:

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

```
MacDonald
```

MASTER YODA: Given a sentence, return a sentence with the words reversed

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

Note: The .join() method may be useful here. The .join() method allows you to join together strings in a list with some connector string. For example, some uses of the .join() method:

```
>>> "--".join(['a','b','c'])
>>> 'a--b--c'
```

This means if you had a list of words you wanted to turn back into a sentence, you could just join them with a single space string:

```
>>> " ".join(['Hello','world'])
>>> "Hello world"
```

In [ ]:

```
def master_yoda(text):
    l1=[]
    l=text.split()
    for i in range(len(l)-1,-1,-1):
        l1.append(l[i])
    print(" ".join(l1))
```

In [ ]:

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

```
home am I
```

In [ ]:

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

```
ready are We
```

ALMOST THERE: Given an integer n, return True if n is within 10 of either 100 or 200

```
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 [ ]:

```
def almost_there(n):
    if n>=90 and n<=110 or n>=190 and n<=210:
        return True
    else:
        return False
```

In [ ]:

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

Out[ ]:

```
True
```

In [ ]:

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

Out[ ]:

```
False
```

In [ ]:

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

Out[ ]:

```
True
```

LEVEL 2 PROBLEMS

FIND 33:

Given a list of ints, return True if the array contains a 3 next to a 3 somewhere.

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

In [ ]:

```
def has_33(nums):
    f=0
    for i in range(0,len(nums)):
        if nums[i]==3 and i!=len(nums)-1:
            if nums[i+1]==3:
                f=1
                return True
    if f==0:
        return False
```

In [ ]:

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

Out[ ]:

```
True
```

In [ ]:

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

Out[ ]:

```
False
```

In [ ]:

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

Out[ ]:

```
False
```

PAPER DOLL: Given a string, return a string where for every character in the original there are three characters

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

In [ ]:

```
def paper_doll(text):
    s=""
    for i in text:
        for j in range(0,3):
            s+=i
    print(s)
```

In [ ]:

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

```
HHHeeellllllooo
```

In [ ]:

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

```
MMMiiissssssiiissssssiiippppppiii
```

Great Job!