Python Screening Task 3

About these tasks

These tasks are designed to challenge your python skills that you should have picked up during your Udacity course.

Feel free to use google to research any topics below, however, do not copy and paste any code directly.

Problem 1

In this problem, we will complete a function that can determine if a string is a palindrome

1. Complete the function `is_palindrome`.
2. The function `is_palindrome` should take one argument named `text`.
3. The function `is_palindrome` should return true if the value of `text` is a palindrome, and should otherwise return false.

Note: A a **palindrome** is a word, number, phrase, or other sequence of characters which reads the same backward as forward.

```
def is_palindrome(text):
    text=text.lower()
    if(len(text)==1):
        return True
    elif(len(text)==2):
        if(text[0]==text[1]):
            return True
        else:
            return False
    elif(len(text)>2 and text[0]==text[-1]):
        text=text[1:-1]
        result=is_palindrome(text)
        if(result):
            return True
        else:
            return False
    else:
        return False


# Do not modify code below the code below this line
assert is_palindrome("redivider") == True
assert is_palindrome("drive") == False
assert is_palindrome("kayak") == True
assert is_palindrome("11211") == True
assert is_palindrome("1115544") == False
```

Problem 2

In this problem, we will try to complete a function that will be able to tell us how many times a word is found in a collection of sentences.

1. Complete the function `find_occurrences`.
2. The function should take two arguments `sentence_list` and `search_term`.
3. `sentence_list` should be an array of strings
4. `search_term` should be a string.
5. The function should return an integer greater than or equal to zero.
6. The number returned should be the number of occurrences of the `search_term` found within `sentence_list`

You can look at the **assertions** at the bottom of the problem to see how your program is expected to perform. For example:

```
assert find_occurrences(["welcome to our Python program", "Python is my favorite language!", "I a
```

Using this we can see, given the list of strings and searching for `"Python"` we expect the function to return `4`

```
def find_occurrences(sentence_list, search_term):
  # Write your code here
      return search_term.count(sentence_list)


  return 5

# Do not modify code below the code below this line
assert find_occurrences(["welcome to our Python program", "Python is my favorite language!
assert find_occurrences(["this is the best day", "Python is the best language for learning
assert find_occurrences(["welcome", "language", "I am", "I love"], "Python") == 0
assert find_occurrences(["What are you doing?", "you like programming?", "We are students"
assert find_occurrences(["welcome welcome", "wikipedia", "wonderland", "we"], "w") == 5
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-30-a1d058e658ac> in <module>
      9
     10 # Do not modify code below the code below this line
---> 11 assert find_occurrences(["welcome to our Python program", "Python is my favo
Pythons", "I love Python"], "Python") == 4
     12 assert find_occurrences(["this is the best day", "Python is the best languag
learning", "I love learning"], "learning") == 3
     13 assert find_occurrences(["welcome", "language", "I am", "I love"], "Python")

AssertionError:
```

SEARCH STACK OVERFLOW

Problem 3

In this problem, we will replace a function with an equivalent lambda function

1. Create a **lambda function** called `module_2_lambda` that does the same task as `module_2`
2. Remove the `module_2` function
3. Modify the assertion so that it evaluates your lambda

```
def module_2(num):
  return num % 2

# Write your code here


# Do not modify the function below
def is_a_lambda(v):
  LAMBDA = lambda:0
  return isinstance(v, type(LAMBDA)) and v.__name__ == LAMBDA.__name__

# Modify this assertion
assert is_a_lambda(module_2) == True
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-28-5f06f8ad1e03> in <module>
```

Problem 4

In this problem identify the error in the following code and fix it.

1. There should be a function called `increase_task_count()`

```
task_count = 5

def increase_task_count():
    task_count += 1

increase_task_count()

# Do not modify below the code below this line
assert task_count == 6
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
<ipython-input-27-33fb44120687> in <module>
      4     task_count += 1
      5
----> 6 increase_task_count()
      7
      8 # Do not modify below the code below this line

<ipython-input-27-33fb44120687> in increase_task_count()
      2
      3 def increase_task_count():
----> 4     task_count += 1
      5
      6 increase_task_count()

UnboundLocalError: local variable 'task_count' referenced before assignment
```

SEARCH STACK OVERFLOW

Problem 5

In this problem, we have a function that performs a division operation, but it crashes when dividing by zero.

1. Modify the following function to catch and handle the ZeroDivisionError

```
def divide(divisor, dividend):
  return (dividend / divisor)
```

```
# Do not modify below the code below this line
assert divide(0, 10) == "Division by zero not allowed"
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-20-0c18ed2a36a1> in <module>
      3 
      4 # Do not modify below the code below this line
----> 5 assert divide(0, 10) == "Division by zero not allowed"

<ipython-input-20-0c18ed2a36a1> in divide(divisor, dividend)
      1 def divide(divisor, dividend):
----> 2   return (dividend / divisor)
      3 
      4 # Do not modify below the code below this line
      5 assert divide(0, 10) == "Division by zero not allowed"

ZeroDivisionError: division by zero
```

SEARCH STACK OVERFLOW

Problem 6

In this problem, someone has written some code using the NumPy library, but has not imported the library correctly.

1. Import the NumPy library to be able to run the code.

```
import numpy as np

arrange = np.arange(15).reshape(3, 5)

# Do not modify below the code below this line
assert np.allclose(arrange, [[0,1,2,3,4],[5,6,7,8,9],[10,11,12,13,14]]) ==  True
```