Practice Worksheet

1. Write a function `avoids` that takes a word (a string) and a string of forbidden characters. It returns False if the word contains any forbidden characters and returns True of the word avoids all forbidden characters.

   Hint: Remember you can check to see if something is ‘in’ a string, list, or tuple.

   Solution:

   ```python
def avoids(word, forbidden):
    for letter in word:
        if letter in forbidden:
            return False
    return True
```

2. Write a function `guess_a_number` function that takes a min and a max integer as input. The function generates a random secret integer between min and max (inclusive) and prompts the user to guess the number until they guess correctly. Then print, “That’s it!”.

   Hint 1: The random module contains a function `randint(a,b)` that will generate a random integer between a and b (inclusive).

   Hint 2: The `raw_input` function gets input from the user as a string. Convert it to an integer using the `int` function.

   Sample run of the function:

   ```
>>> guess_a_number(0,5)
Guess a number between 0 and 5: 0
Nope! Guess again: 4
Nope! Guess again: 3
That’s it!
```

   Solution:

   ```python
def guess_a_number(min, max):
    secret = randint(min, max)
    num = int(raw_input("Guess a number between "+str(min)+" and "+str(max)+": "))
    while num != secret:
        num = int(raw_input("Nope! Guess again: "))
    print "That’s it!"
```
3. Write a function `shift_digits` that takes a string as input. The function should return a new string with one added to any digit (‘0’-'9') in the string. Any ‘9’s should wrap back around to ‘0’. Any other characters in the string should remain untouched.

Hint: The character ‘0’ is a 48 on the ascii table. The character ‘9’ is a 57.

Hint 2: Don’t forget the `ord`, `chr`, and `join` functions.

Sample Run:

```python
>>> shift_digits("abc123XYZ99")
'abc234XY200'
```

Solution:

```python
def shift_digits(word):
    s = []
    for c in word:
        if c >= '0' and c <= '8':
            c = chr(ord(c) + 1)
        elif c == '9':
            c = '0'
        s.append(c)
    return ''.join(s)
```

4. Write a function that takes, as a parameter, a list of spheres. It will compute the distances between every possible pair of spheres in the list and return a tuple containing copies of the 2 spheres that are farthest apart.

```python
class Point:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

class Sphere:
    def __init__(self, center, radius):
        self.center = center
        self.radius = radius

def distance(s1, s2):
    return sqrt((s1.center.x-s2.center.x)**2 +
                (s1.center.y-s2.center.y)**2 +
                (s1.center.z-s2.center.z)**2)
```
def farthest_spheres(list):
    # return None if the list doesn't have two spheres
    if len(list) < 2:
        return None

    # assume the first two are the farthest apart
    s1 = list[0]
    s2 = list[1]
    farthest = distance(s1, s2)

    # search for a pair that are farther
    for i in range(len(list)-1):
        for j in range(i+1, len(list)):
            dist = distance(list[i], list[j])
            if dist > farthest:
                s1 = list[i]
                s2 = list[j]
                farthest = dist

    # make copies of the spheres
    s1 = Sphere(Point(s1.center.x, s1.center.y, s1.center.z),
                 s1.radius)
    s2 = Sphere(Point(s2.center.x, s2.center.y, s2.center.z),
                 s2.radius)

    return (s1, s2)