## Activity 3 - Code Review Hands One

### Code Snippets to Review

You will need to provide feedback as demonstrated in the assignment description for the following five snippets. Please use this template document to complete this assignment. You will need to provide written feedback and also “corrected” code just like in the examples in the assignment description.

**Name:** Christopher Eckerson

**email:** eckersoc@oregonstate.edu

### Snippet 1

# Pull request 1  
def my\_func(x):  
  
  
 return x\*\*2

**Feedback:** This function is very concise and dry, providing no unnecessary additional code. Continuing with the concise design, improvements can be made with the removal of white spaces. It would also be helpful for readability to include an attribute docstring as well as choosing a function name that is reflective of the nature of the function itself. An example of these suggestions is included below:

def square(x):  
 *“”” Returns the square value of x ”””*  
 return x\*\*2

### Snippet 2

# Pull request 2  
def create\_odds(num):  
 “““Creates a list of len(num) of random odd numbers between 1 and 1000”””  
 num\_list = []  
 for i in range(0, num):  
 new\_num = 2  
 while new\_num % 2 == 0:  
 new\_num = random.randint(1, 1000)  
 num\_list.append(new\_num)  
 return num\_list  
  
def create\_evens(num):  
 “““Creates a list of len(num) of random even numbers between 1 and 1000”””  
 num\_list = []  
 for i in range(0, num):  
 new\_num = 1  
 while new\_num % 2 != 0:  
 new\_num = random.randint(1, 1000)  
 num\_list.append(new\_num)  
 return num\_list

**Feedback:** These two functions have good function names as well as relevant and understandable attribute docstrings. Additionally, the code works to generate the desired outcome of a list of either odd or even number between 1000. It is also great to see singular ideas/tasks assigned to each function (rather than one function that handles both of these). The code includes no comments, soit would be helpful to include more to ease readability. To be more pythonic, the random library includes the random.randrange() function which can allow selections to be made between a range with a given increment. Since odd and even integers are in increments of 2, this would work to reduce extra code. Continuing to minimize additional code, we can remove the minimum parameter of range() since this defaults to 0. An example of these suggestions is included below:

def create\_odds(num):  
 “““Creates a list of len(num) of random odd numbers between 1 and 1000”””  
 num\_list = []  
 for i in range(num):

# Generate an odd number in the range 1-999 and then append to list  
 new\_num = random.randrange(1, 999, 2)

num\_list.append(new\_num)  
 return num\_list  
  
def create\_evens(num):  
 “““Creates a list of len(num) of random even numbers between 1 and 1000”””  
 num\_list = []  
 for i in range(num):  
 # Generate an even number in the range 2-1000 and append to list  
 new\_num = random.randrange(2, 1000, 2)

num\_list.append(new\_num)  
 return num\_list

### Snippet 3

# Pull request 3  
def check\_for\_val(self, val):  
 “““This member function checks to see if val exists in the class member  
 values and returns True if found”””  
 for i in range(len(self.values)):  
 if self.values[i] == val:  
 return True  
 return False

**Feedback:** This function is concise for the operation it is performing, includes a clear docstring, and the name is self-explanatory. To make this function even more pythonic, return the result from checking if value is in self.value. This will reduce the code further and allow the docstring to be plenty of documentation for the function. An example of these suggestions is included below:

def check\_for\_val(self, val):  
 “““This member function checks to see if val exists in the class member  
 values and returns True if found”””  
 return val in self.values

### Snippet 4

# Pull request 4  
def get\_val\_index(arr, val):  
 “““Searches arr for val and returns the index if found, otherwise -1”””  
 index = -1  
 for i in range(len(arr)):  
 if arr[i] == val:  
 index = i  
 break  
 return index

**Feedback:** This function has a good relevant name and clear docstring to explain the task of the function. This also does a good job recognizing that if an index isn’t found that the index returned has to be defaulted to -1. The code could include comments within the code to further improve readability. Is the code is intending to return the first occurrence of the value and forgo later occurrences in the list or would rather return any occurrence of the value (returning a list of indices)? In the first case, there is a helpful build-in method for lists which will return the first occurrence (see suggestion) and further help to minimize code. The list.index() does work well if the value exists to return the index but will throw a ValueError if not found though. Using a try & except can help to handle this error and return the value of -1. An example of these suggestions is included below:

def get\_val\_index(arr, val):  
 “““Searches arr for first occurrence of val and returns the index if found, otherwise -1”””  
 try:

return arr.index(val)

except ValueError:  
 return -1

### Snippet 5

# Pull request 5  
int\_arr = [1, 2, 5, 2, 10, 45, 9, 100]  
  
def print\_sorted(arr):  
 “““Prints the items in the array after sorting”””  
 arr.sort()  
 for num in arr:  
 print(num)

**Feedback:** This function does a really great job using the built-in method for lists to sort the provided array. Additionally, the docstring and function name are very relevant to the task and leave no guess work to the functions purpose. It seems that there is an additional “int\_arr” included in this pull request that may have been included from testing but may be good to remove if not intended. An example of the suggestion is seen below:

def print\_sorted(arr):  
 “““Prints the items in the array after sorting”””  
 arr.sort()  
 for num in arr:  
 print(num)