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: Clinton Hawkes

email: hawkesc@oregonstate.edu

Snippet 1

```
# Pull request 1
def my_func(x):


    return x**2
```

Feedback: This is a very clean and simple function, so nice work. I was wondering, due to its simplicity, if you have thought about discarding this function all together? I was thinking that it takes more time to write out my_func(x) than it does to write x**2. I think it would be more readable if you just used the built in ** operator. If you still want to use this function, I would suggest giving it a more descript name. Such as squared( ) or something similar. I usually like to see some comments in the function, but this one is so simple, I don't think you really need any comments in the code. Maybe just remove the extra blank lines in the function.

Snippet 1

```
# Pull request 1
def squared(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: Good job writing these two functions. You got all the bounds correct for range() and randint(), which something I usually have to look up to see if the start/end values are included or not. Everything looks like it will function correctly. I do want to mention a built-in random library function that I learned recently. It is random.randrange().

Randrange() is a great function to use when you are trying to return just even or odd numbers, because you can step through a range of numbers by your desired number. (alternating every other number in this case) It is used as such: randrange(start num, end num (not included), step). I included sample code with its use below.

I also couldn't help but notice that these two functions are very similar. Have you thought about combining the functions? It would be very easy to pass "even" or "odd" into the function as a parameter. Combining the two may condense and simplify your code, but I don't know if the rest of your code base already relies on these two functions. This may not be something you think needs changed, but I have included some code that combines the two functions for your review.

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(num):
        num_list.append(random.randrange(1, 1001, 2))
    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):
        num_list.append(random.randrange(2, 1001, 2))
    return num_list
```

Or, if you want the two functions combined:

```
# Pull request 2
def create_list(num, parity):
    """Creates a list of len(num) of random even/odd numbers between 1 and
    1000, where the parity is provided"""
    num_list = []
    if parity == even:
        for i in range(num):
            num_list.append(random.randrange(2, 1001, 2))
    elif parity == odd:
        for i in range(num):
            num_list.append(random.randrange(1, 1001, 2))
    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: It looks like your function follows all style guidelines, and it is very easy to read/follow the code. Nice work! Even though your current code functions correctly, I do have one suggestion that may help you out. Did you know that you can return the result of an if statement without needing to explicitly return True or False? You would just check if any values in self.values equals val and return the result of the if statement. (code shown below) It is easy to implement and simplifies the code. I thought I would suggest it to you, because I was grateful when somebody else pointed it out to me. It has helped me clean up my code, and I'm always looking for ways to improve. Hope this helps.

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"""
    return if any(index == val for index 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: I can see this function being very useful. I have needed to use something similar many times. It also looks like everything adheres to the style guide. I think I may have some suggestions that will simplify your function. Personally, I usually try to avoid creating variables if they aren't absolutely necessary. I found that the "index" variable could be eliminated and the index value could be returned directly without assigning it to a variable. By doing this, we are able to remove the break statement as well, because the loop will no longer continue if it finds a match.

If the for loop makes it through all the elements in the array without finding a match to val, -1 is then returned. I think these changes simplify the function and make it a little easier to read. I have included example code with these suggestions below if you would like to review it.

Snippet 4

```
# Pull request 4
def get_val_index(arr, val):
    """Searches arr for val and returns the index if found, otherwise -1"""
    for i in range(len(arr)):
        if arr[i] == val:
            return i
    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: Your function looks good, and I appreciate the descriptive names of the function and variables. I was wondering if there is a reason you need to create a function for this rather than just using the built-in functions, such as print(int_arr.sort())? I can see that you

would want to create a function if you needed the elements in the array to be printed vertically on the screen rather than in the array notation, so I just thought I would ask. If this type of printing is not needed, I personally think it would be easier to just use the built-in print function mentioned above. Just a thought.

Also, I wasn't sure if you wanted the array to be sorted in ascending (default) or descending order. The comment just said sorted, so I suggest adding what kind of sorting in the comments.

I see that there is an array of ints created outside of the print_sorted function, but I don't see anything being done with the array. Did you mean to call the function on that array? I am not sure what needs to be done, so I am assuming the int_arr needs to be printed. If this is not the case, or if you would like me to review any additional code, please don't hesitate to ask.

Snippet 5

```
# Pull request 5
int_arr = [1, 2, 5, 2, 10, 45, 9, 100]

print_sorted(int_arr)

def print_sorted(arr):
    """Prints the items in the array after sorting in ascending order"""
    arr.sort()
    for num in arr:
        print(num)
```

If you just used the built-in functions, you would write this:

```
# Pull request 5
int_arr = [1, 2, 5, 2, 10, 45, 9, 100]

print(int_arr.sort())
```