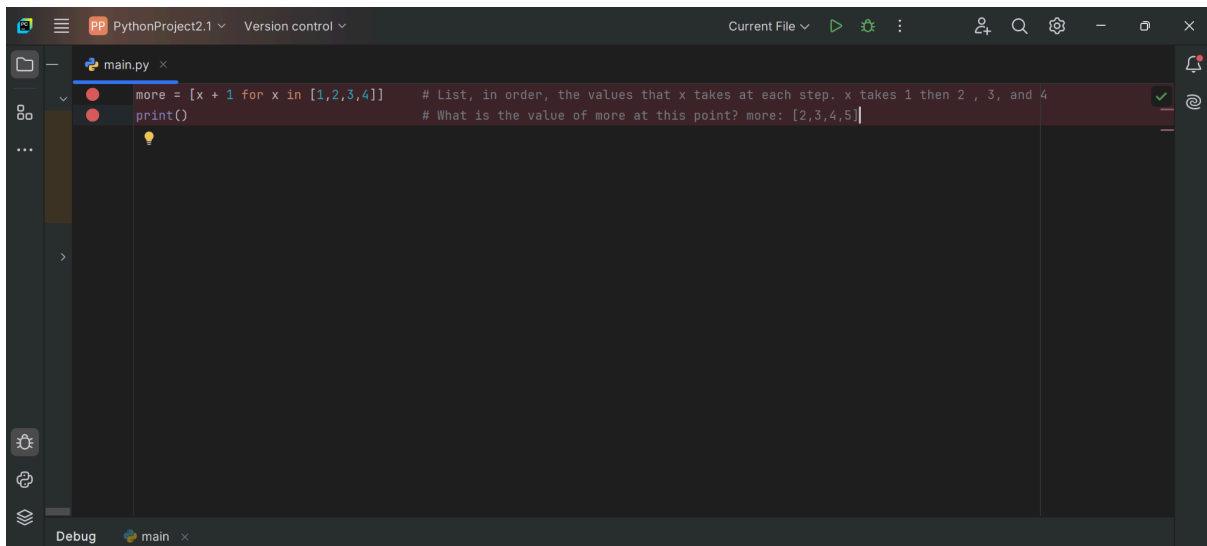


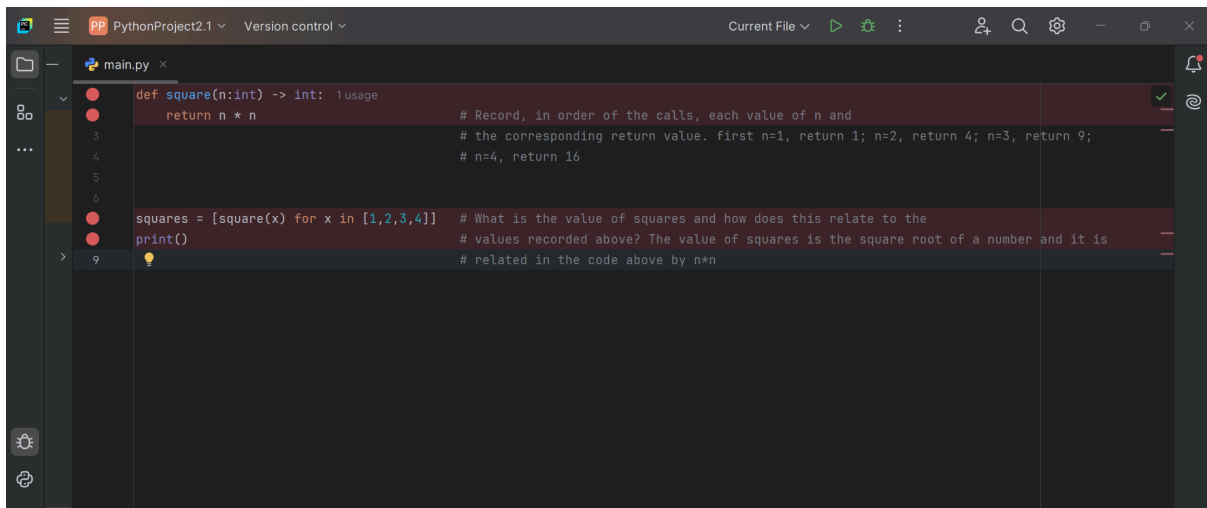
Task 1 : Evaluating Code with List Comprehensions

a.



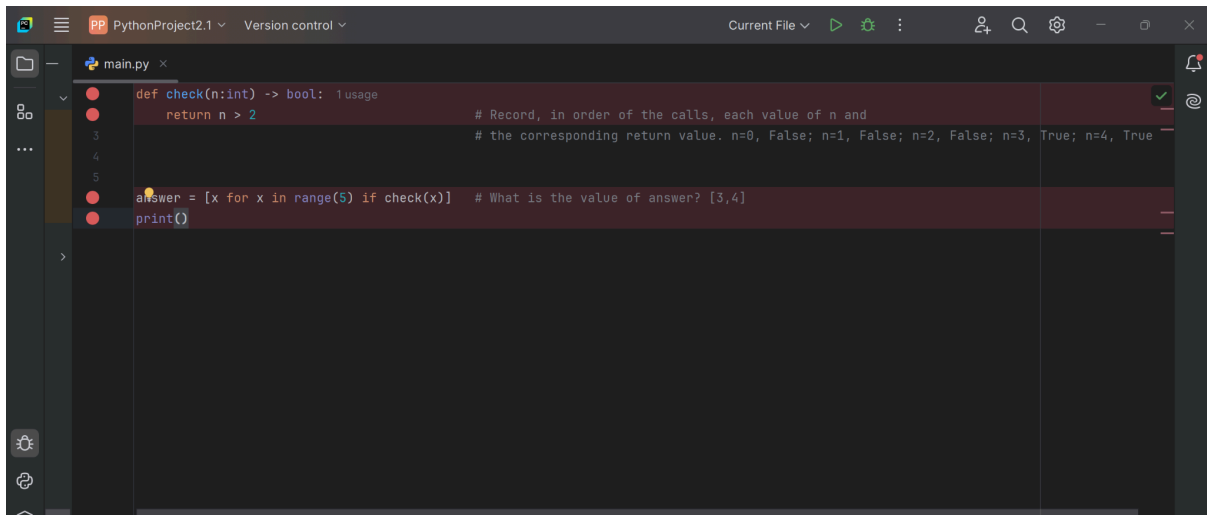
```
main.py
more = [x + 1 for x in [1,2,3,4]] # List, in order, the values that x takes at each step. x takes 1 then 2 , 3, and 4
print() # What is the value of more at this point? more: [2,3,4,5]
```

b.



```
main.py
def square(n:int) -> int: 1usage
    return n * n # Record, in order of the calls, each value of n and
                 # the corresponding return value. first n=1, return 1; n=2, return 4; n=3, return 9;
                 # n=4, return 16
squares = [square(x) for x in [1,2,3,4]] # What is the value of squares and how does this relate to the
print() # values recorded above? The value of squares is the square root of a number and it is
        # related in the code above by n*n
```

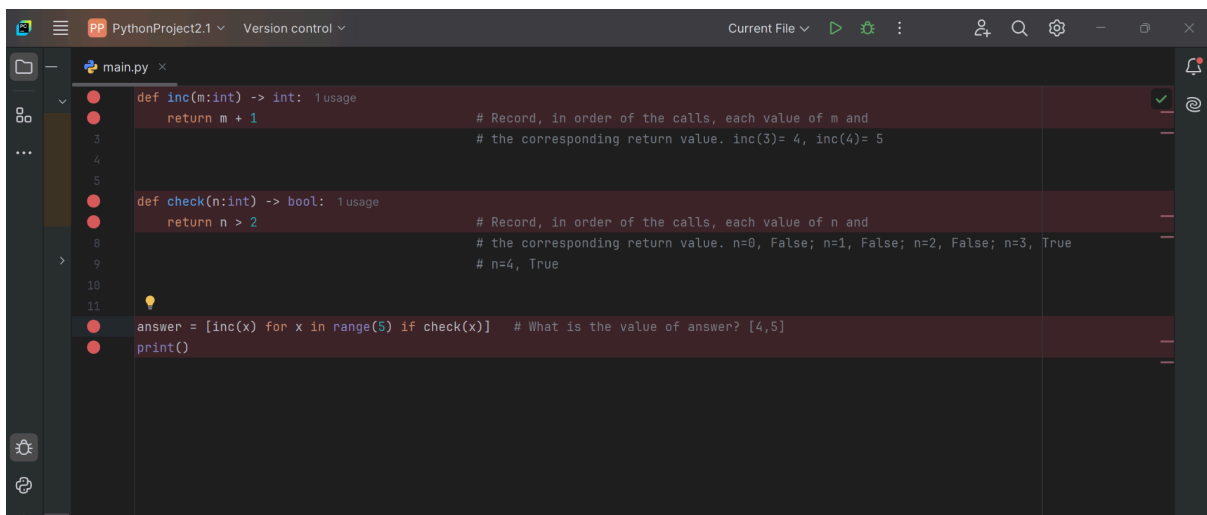
c.



```
def check(n:int) -> bool: 1 usage
    return n > 2           # Record, in order of the calls, each value of n and
                           # the corresponding return value. n=0, False; n=1, False; n=2, False; n=3, True; n=4, True

answer = [x for x in range(5) if check(x)] # What is the value of answer? [3,4]
print()
```

d.



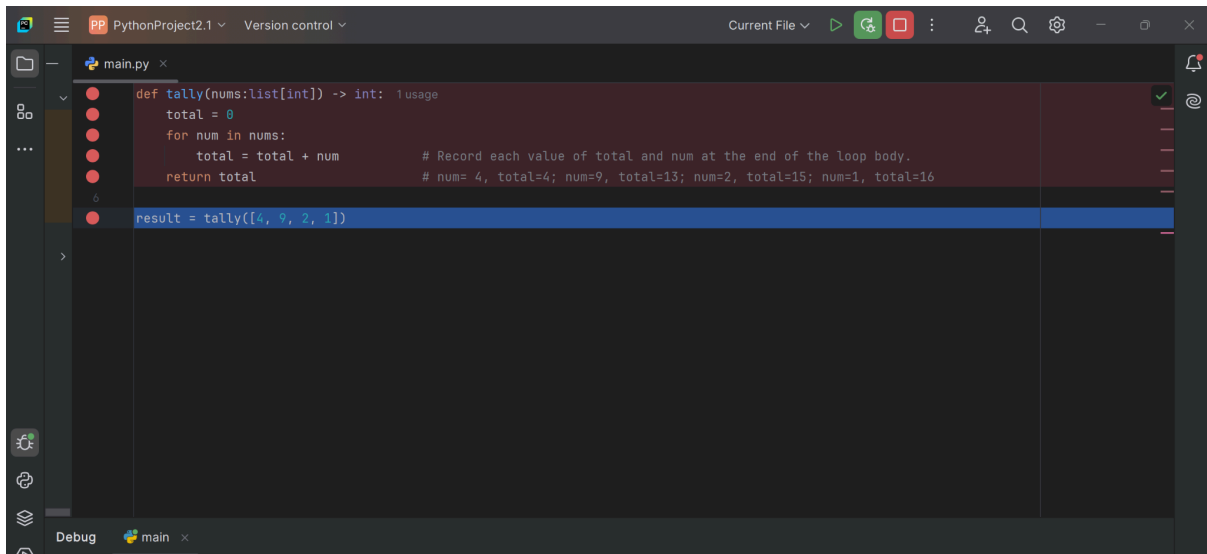
```
def inc(m:int) -> int: 1 usage
    return m + 1         # Record, in order of the calls, each value of m and
                           # the corresponding return value. inc(3)= 4, inc(4)= 5

def check(n:int) -> bool: 1 usage
    return n > 2         # Record, in order of the calls, each value of n and
                           # the corresponding return value. n=0, False; n=1, False; n=2, False; n=3, True
                           # n=4, True

answer = [inc(x) for x in range(5) if check(x)] # What is the value of answer? [4,5]
print()
```

Task 2 Evaluating Code with Loops

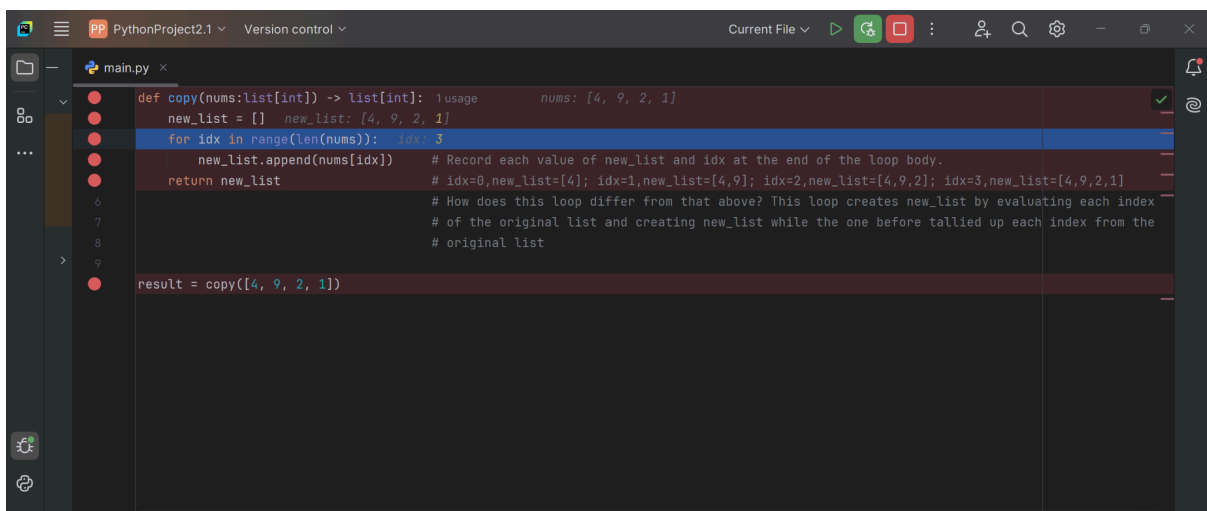
a.



```
def tally(nums:list[int]) -> int: 1 usage
    total = 0
    for num in nums:
        total = total + num      # Record each value of total and num at the end of the loop body.
    return total                # num= 4, total=4; num=9, total=13; num=2, total=15; num=1, total=16

result = tally([4, 9, 2, 1])
```

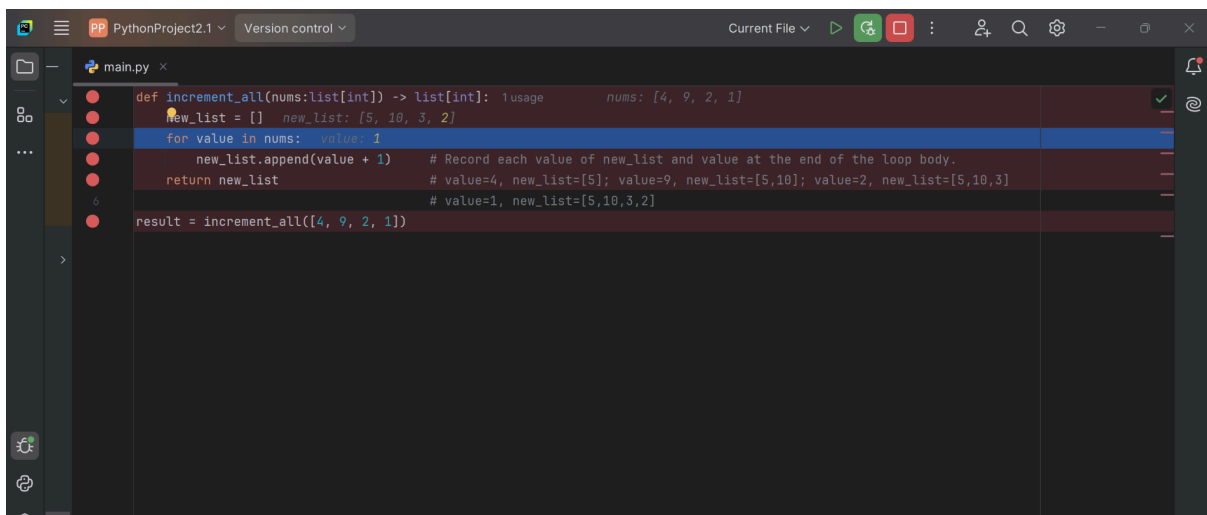
b.



```
def copy(nums:list[int]) -> list[int]: 1 usage      nums: [4, 9, 2, 1]
    new_list = []  new_list: [4, 9, 2, 1]
    for idx in range(len(nums)):  idx: 3
        new_list.append(nums[idx])  # Record each value of new_list and idx at the end of the loop body.
    return new_list                # idx=0,new_list=[4]; idx=1,new_list=[4,9]; idx=2,new_list=[4,9,2]; idx=3,new_list=[4,9,2,1]
    # How does this loop differ from that above? This loop creates new_list by evaluating each index
    # of the original list and creating new_list while the one before tallied up each index from the
    # original list

result = copy([4, 9, 2, 1])
```

c.



```
def increment_all(nums:list[int]) -> list[int]: 1 usage      nums: [4, 9, 2, 1]
    new_list = []  new_list: [5, 10, 3, 2]
    for value in nums:  value: 1
        new_list.append(value + 1)  # Record each value of new_list and value at the end of the loop body.
    return new_list                # value=4, new_list=[5]; value=9, new_list=[5,10]; value=2, new_list=[5,10,3]
    # value=1, new_list=[5,10,3,2]

result = increment_all([4, 9, 2, 1])
```

Task 3 Evaluating Code with Test Cases

<https://github.com/CSC-101/lab3-omardadam.git>