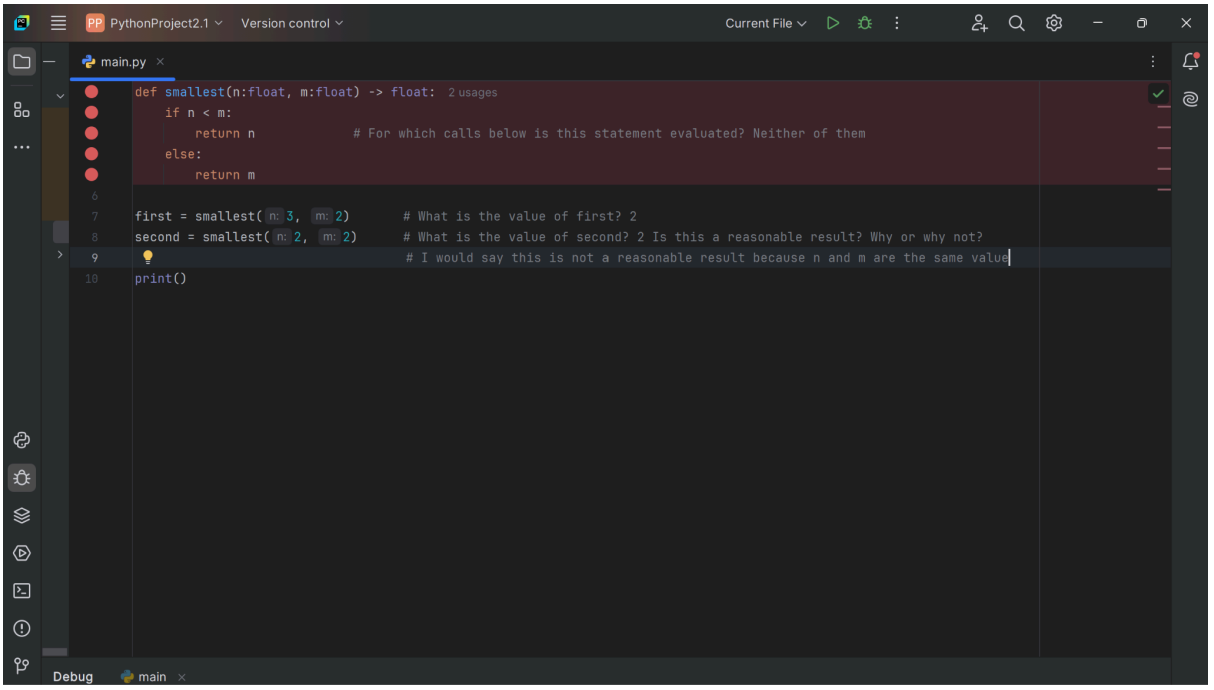
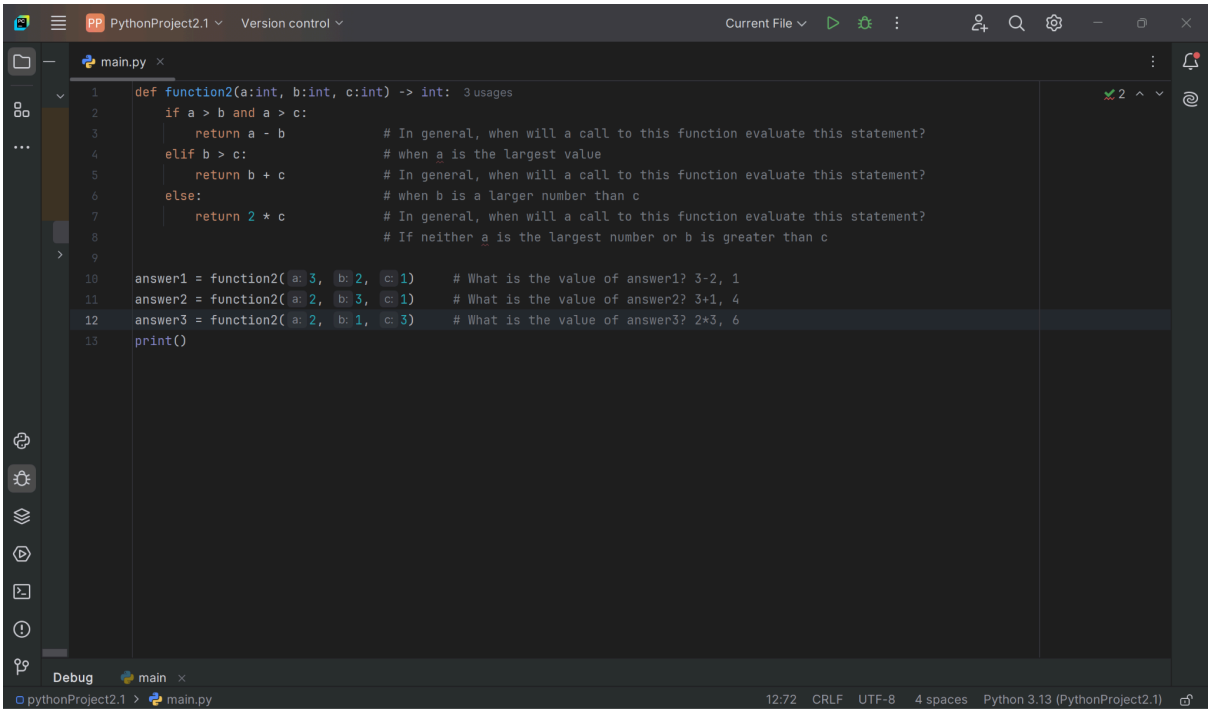
Task 3: Evaluating Code with Conditionals

a.

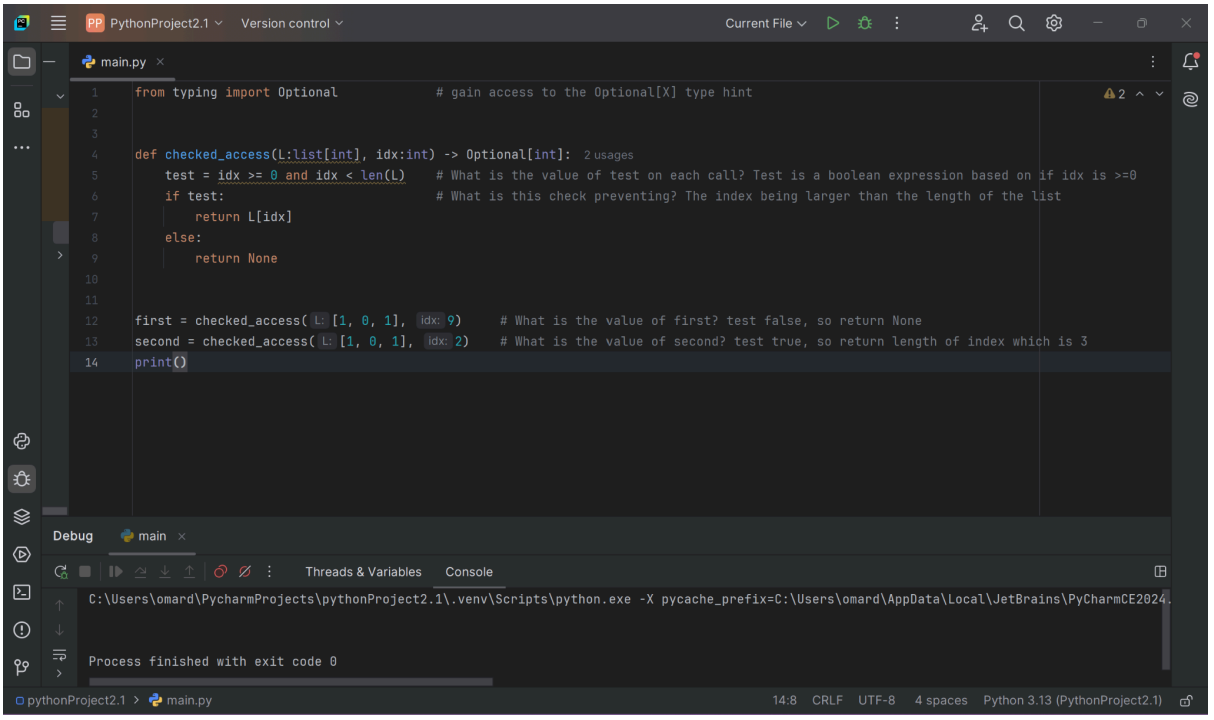


b.

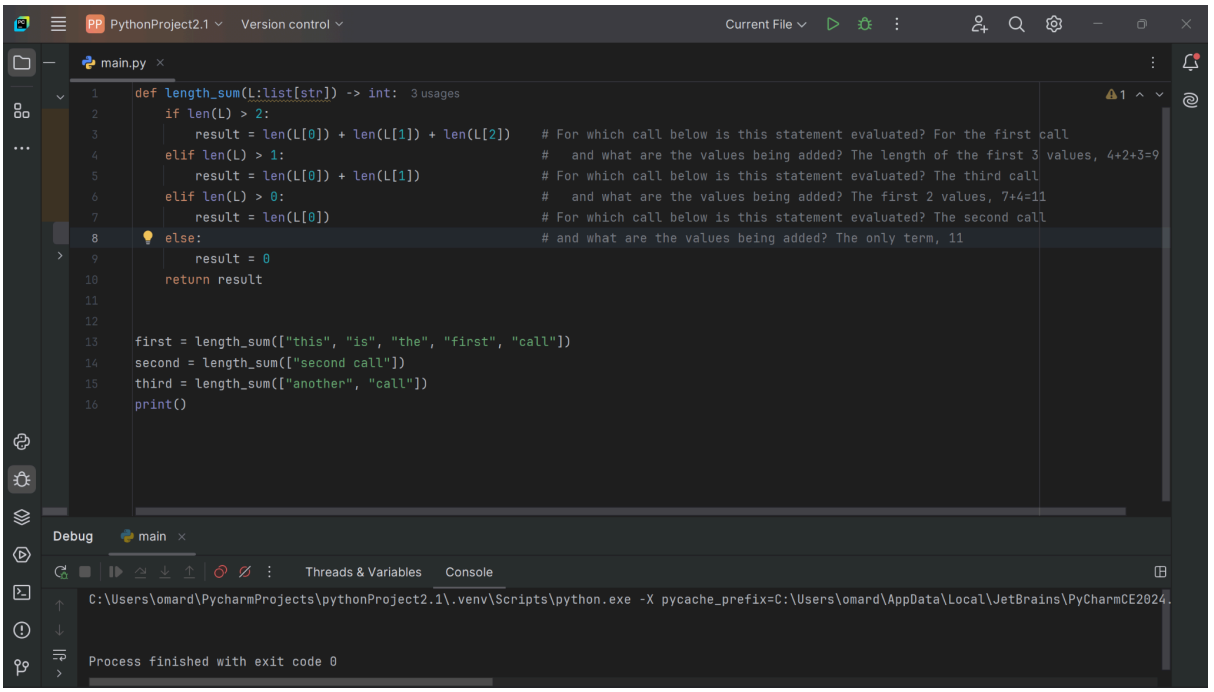


Task 4: Evaluating Code with Lists and Strings

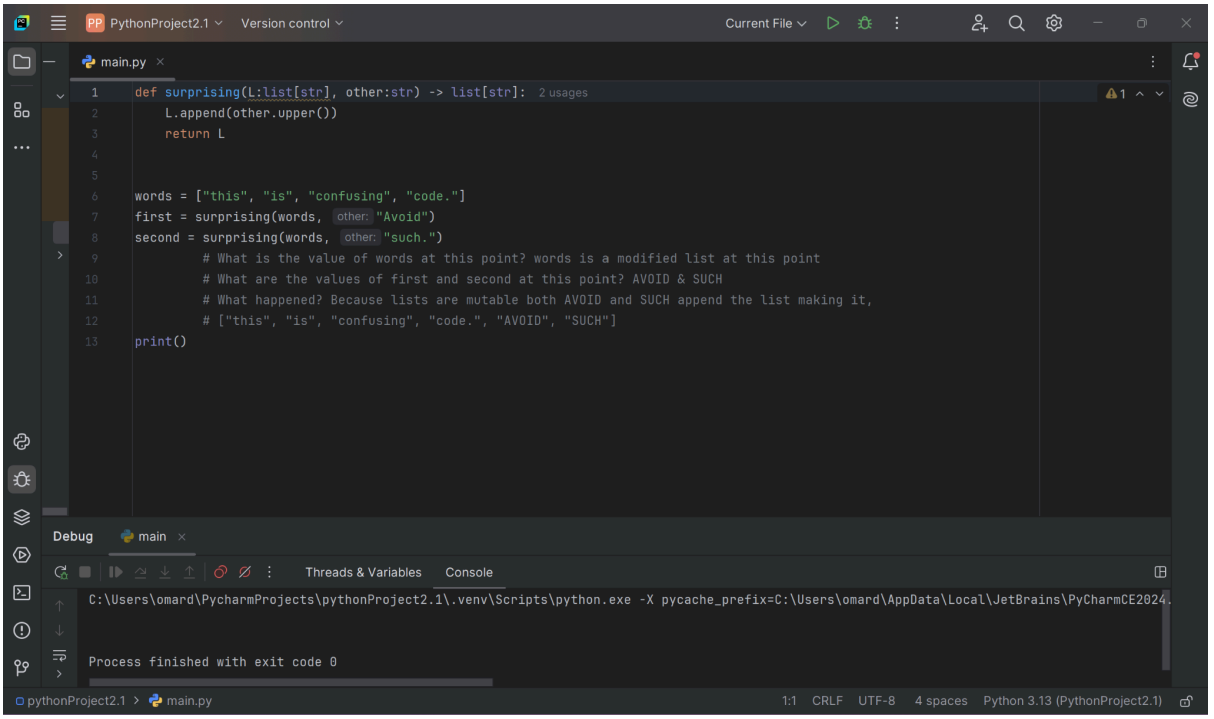
a.



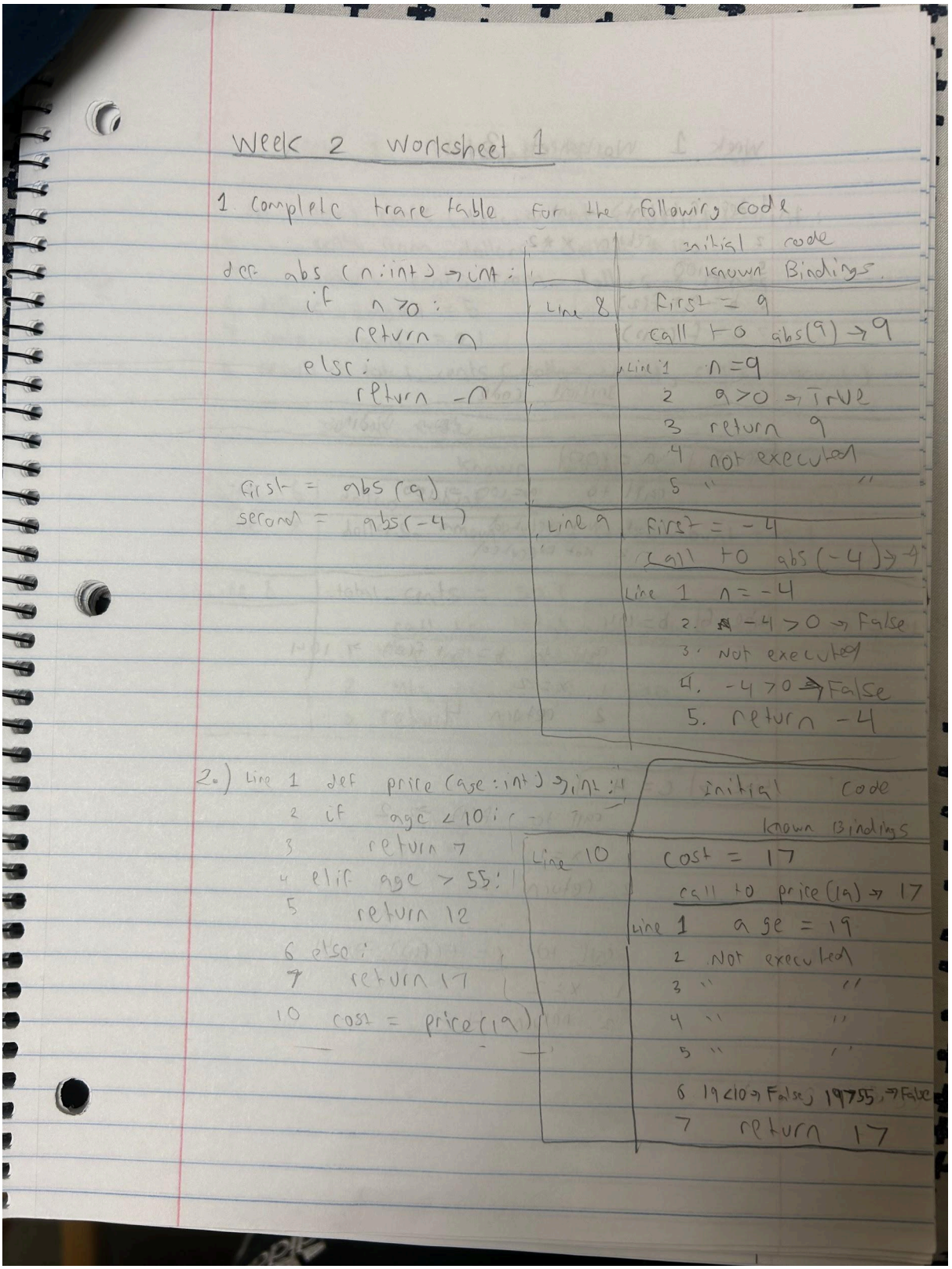
b.



C.


PythonProject2.1
Version control
Current File
main.py
def surprising(L:list[str], other:str) -> list[str]: 2 usages
L.append(other.upper())
return L
words = ["this", "is", "confusing", "code."]
first = surprising(words, other: "Avoid")
second = surprising(words, other: "such.")
# What is the value of words at this point? words is a modified list at this point
# What are the values of first and second at this point? AVOID & SUCH
# What happened? Because lists are mutable both AVOID and SUCH append the list making it,
# ["this", "is", "confusing", "code.", "AVOID", "SUCH"]
print()
Debug
main
Threads & Variables
Console
C:\Users\omard\PycharmProjects\pythonProject2.1\.venv\Scripts\python.exe -X pycache_prefix=C:\Users\omard\AppData\Local\JetBrains\PyCharmCE2024
Process finished with exit code 0
pythonProject2.1
main.py
1:1
CRLF
UTF-8
4 spaces
Python 3.13 (PythonProject2.1)

Worksheets



Week 2 Worksheet 1

1. Complete trace table for the following code

```
def abs(n:int) -> int:
    if n >= 0:
        return n
    else:
        return -n

first = abs(9)
second = abs(-4)
```

	Initial code / known Bindings
Line 8	first = 9
	call to abs(9) → 9
Line 1	n = 9
2	9 >= 0 → True
3	return 9
4	not executed
5	" "
Line 9	first = -4
	call to abs(-4) → 4
Line 1	n = -4
2.	-4 >= 0 → False
3.	Not executed
4.	-4 >= 0 → False
5.	return -4

2.)

```
Line 1  def price(age:int) -> int:
    2       if age < 10:
    3           return 7
    4       elif age > 55:
    5           return 12
    6       else:
    7           return 17
   10   cost = price(19)
```

	Initial code / known Bindings
Line 10	cost = 17
	call to price(19) → 17
Line 1	age = 19
2	Not executed
3	" "
4	" "
5	" "
6	19 < 10 → False; 19 > 55 → False
7	return 17

Week 2 Worksheet 2

```
1 def abs(n:int)->int:
      if n>0:
        return n
      else:
        return -n
```

(a.) n > 0 → true evaluated by line 3.

(b.) n < 0 before evaluation of line 5.

2.)
```
1 def price(age:int)->int:
2     if age<10:
          return 7
      elif age>55
          return 12
      else:
          return 17
```

(a.) age < 10 → true by evaluation of line 3

(b.) age > 55 → true by evaluation of line 5

(c.) age < 55, age > 10 both true by evaluation of line 7.

Week 2 worksheet 3

1. L = [1, 2, 3]
   L[1] + L[2]
   2 + 3
   5

2.) L = [1, 2, 3]
   L[L[0]]
   L[1]
   2

3.) L = [4, 2, 9, 7]
   L[-3] + L[0]
   2 + 4
   6

4.) s = 'hello'
   len(s)
   5

5.) s = 'hello\nthere!'
   s[2] + s[5] + s[-3]
   l + \n + r
   'l\nr'

Week 2 worksheet 4

```
1 def questionable(argL: list[int], idx: int, value: int) -> bool:
2     if idx >= 0 and idx < len(argL):
          argL[idx] = value
          return True
      else:
          return False
```

L = [4, 2, 5, 3, 1, 9]

first = questionable(L, 3, 10)

second = questionable(L, 2, 0)

Known Bindings

Line 9: L = [4, 2, 5, 3, 1, 9]

Line 10: L = [4, 2, 5, 10, 1, 9]

Line 11: L = [4, 2, 0, 10, 1, 9]

Lists

L = [4, 2, 5, 3, 1, 9]

L = [4, 2, 5, 10, 1, 9]

L = [4, 2, 0, 10, 1, 9]

Initial code

call to first = questionable(L, 3, 10) → L = [4, 2, 5, 10, 1, 9]

1. argL = [4, 2, 5, 3, 1, 9], idx = 3, value = 10
2. idx >= 0 → true idx < len(argL) → true
3. argL[3] = 10
4. return true
9. L = [4, 2, 5, 10, 1, 9]

call to questionable(L, 2, 0) → L = [4, 2, 0, 10, 1, 9]

1. argL = [4, 2, 5, 10, 1, 9], idx = 2, value = 0
2. idx >= 0 → true idx < len(argL) → true
3. argL[2] = 0
4. return True
9. L = [4, 2, 0, 10, 1, 9]

Week 2 Worksheet 5

1.)

```
1 listOne = []
2 listOne.append(1)
3 listOne.append(3)
4 listOne.append(9)
5
6 listOne + [4]
```

Lists

L = []
L = [1]
L = [1,3]
L = [1,3,9]
L = [1,3,9,4]

initial code

call to listOne + [4] → listOne = [1,3,9,4]

```
1 listOne = []
2 listOne = [1]
3 listOne = [1,3]
4 listOne = [1,3,9]
6 listOne = [1,3,9,4]
```

Known Bindings	
Line 1	listOne = []
Line 2	listOne = [1]
line 3	listOne = [1,3]
lie 4	listOne = [1,3,9]
line 6	listOne = [1,3,9,4]

2.)

```
1 listOne = [4,2,5]
2 listTwo = [8,9]
3 first = listOne + listTwo
4 second = listOne.append(listTwo)
5 third = listOne.extend(listTwo)
```

Lists

list one = [4,2,5]	listTwo = [8,9]
listNew = [4,2,5,8,9]	
listOne = [4,2,5,8,9]	
listOne = [4,2,5,[8,9]]	

initial code

call to first = listOne + listTwo → L = [4,2,5,8,9]

```
1. listOne = [4,2,5]
2 listTwo = [8,9]
3 L = [4,2,5,8,9]
```

call to second = listOne.append(listTwo) → L = [4,2,5,8,9]

```
1 listOne = [4,2,5]
2 listTwo = [8,9]
4 listOne = [4,2,5,8,9]
```

call to third = listOne.extend(listTwo) → listOne = [4,2,5,[8,9]]

1 listOne = [4,2,5]

2 listTwo = [8,9]

5 listOne = [4,2,5,[8,9]]

Known Bindings

Line 1	listOne = [4,2,5]
Line 2	listTwo = [8,9]
Line 3	L = [4,2,5,8,9]
Line 4	listOne = [4,2,5,8,9]
Line 5	listOne = [4,2,5,[8,9]]

Week 2 Worksheet 5

3.) ListOne = [4, 2, 5]
1 other = listOne
3 listOne.append(0)

Lists
ListOne = [4, 2, 5]
ListOne = [4, 2, 5, 0]

initial code
call to listOne.append(0) → listOne = [4, 2, 5, 0]
1. listOne = [4, 2, 5]
2 other = listOne
3 listOne = [4, 2, 5, 0]

Known Bindings

Line 1	ListOne = [4, 2, 5]
Line 2	ListOne = [4, 2, 5], other = [4, 2, 5]
Line 3	ListOne = [4, 2, 5, 0]

Week 2 worksheet 6

1.)

```
1 sentence = "this is a poorly formatted sentence."
2 words = sentence.split()
3 new = ' '.join(words)
```

Lists

words = ["This", "is", "a", "poorly", "formatted", "sentence"]

Initial code

call to new = ' '.join(words) → "This is a poorly formatted sentence."

1. sentence = "this is a poorly formatted sentence."
2. words = ["This", "is", "a", "poorly", "formatted", "sentence"]
3. new = "this is a poorly formatted sentence."

Known Bindings

Line 1	sentence = "This..."
Line 2	words = ["This", etc.]
Line 3	new = "This..."

2.)

```
1 def capitalize(s:str)->str:
2     chars = list(s)
3     chars[0] = chars[0].upper()
4     return "".join(chars)
7 word = capitalize("hello")
```

Lists

list(s) = ['h', 'e', 'l', 'l', 'o']

list(s) = ['H', 'e', 'l', 'l', 'o']

Initial code

call to word = capitalize("hello") → Hello

1. s = "hello"
2. chars = ['h', 'e', 'l', 'l', 'o']
3. chars = ['H', 'e', 'l', 'l', 'o']
4. return "Hello"

Known Bindings

Line 7	"Hello"