CSCI 150 HW: classes + objects reading practice

Due: Wednesday, November 20

To receive full credit, for each exercise you should do the following:

1. **Predict**: First, complete the exercise *without* using the Python interpreter. *Trace the execution of the code, keeping track of the function stack, the heap, all variables, and any output produced.*

   If you like, you may print out copies of the tracing template found at `http://ozark.hendrix.edu/~yorgey/150/static/heap-tracing-template.pdf` (if you have this PDF open on your computer you may simply click the link above).

2. **Check**: Run the code. Does the actual output agree with what you wrote down in step 1?

3. **Evaluate**: If your answer to part 1 was different than the actual output, keep experimenting with it, consult the textbook or Python documentation, ask a friend or TA or professor, *etc.* until you can explain why the code works the way it does *and* what your misunderstanding(s) were in part 1.

You should consider the code in each exercise separately from the other exercises.

1. Trace the execution of the following code.

```python
class Counter:
    def __init__(self, step: int):
        self.count = 0
        self.step = step

    def inc(self):
        self.count += self.step

    def reset(self):
        self.count = 0

    def get_count(self) -> int:
        return self.count

def incs(c: Counter, n: int):
    for i in range(n):
        c.inc()

def main():
    c = Counter(1)
    d = Counter(2)
    e = c

    incs(c, 3)
    e.reset()
    incs(d, 3)
    incs(e, 4)

    print(f"c: {c.get_count()}, d: {d.get_count()}, e: {e.get_count()}")

main()
```

2. Trace the execution of the following code.

```
from typing import *

class Zipper:
    def __init__(self, words: List[str]):
        self.before = []
        self.after  = words

    def right(self):
        self.before.append(self.after.pop(0))

    def get_cur(self):
        return self.after[0]

    def set_cur(self, new_str: str):
        self.after[0] = new_str

    def unzip(self) -> List[str]:
        return self.before + self.after

def main2():
    z = Zipper(["This", "is", "sorta", "cool"])
    z.right()
    z.right()
    z.set_cur("very")
    print(z.unzip())

main2()
```