On your way in…

Hand-in:

1. HW04 into the 2 folders

Pick-up:

1. POGIL Activity: Classes – 24:Attributes & 25:Methods

Midterm Exam is Thursday, March 12

- TPL 203: 5:45pm-7:45pm OR 8-10pm
- **Exam Review Session: 3/9 at 7-8:30pm in TPL 203.**

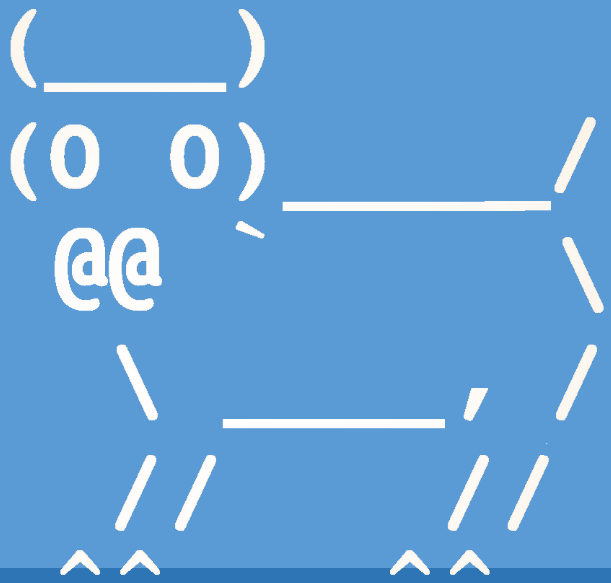
- Closed book exam
- Review your homeworks! POGILs! Slides! Labs!
- Next week's lab will be less intense

Welcome to CS 134!

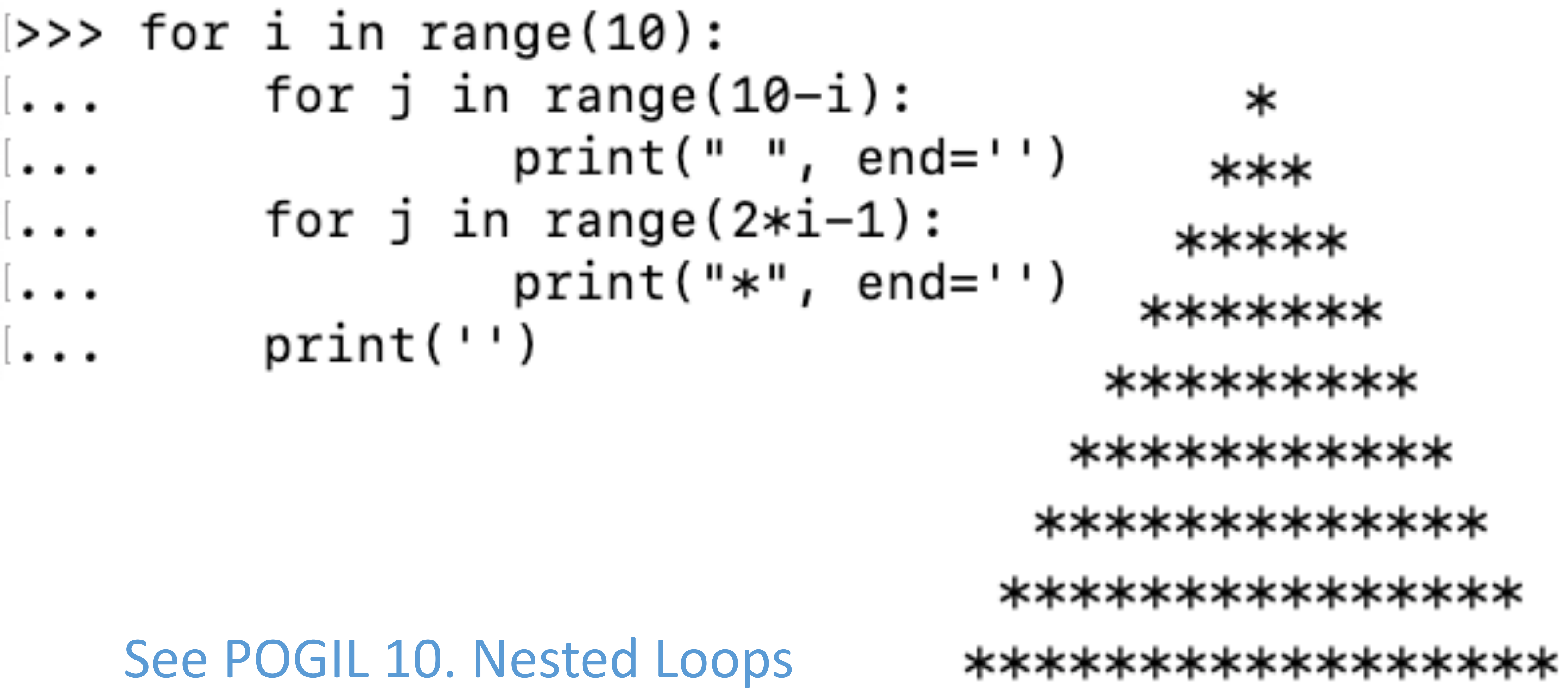
Introduction to Computer Science

Iris Howley

-Classes & Encapsulation-



What does this do?



```
>>> for i in range(10):
...     for j in range(10-i):
...             print(" ", end='')
...     for j in range(2*i-1):
...             print("*", end='')
...     print('')
```

```

         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
```

See POGIL 10. Nested Loops

TODAY'S LESSON

`Abstraction makes programming GREAT`

(Hiding complex implementations behind simpler public interfaces.)

The textbook has really great activities to step through, with exercises to do at the end.

Chapter 4: Case study: interface design

POGIL 25B REPLACES POGIL 25 (WEDNESDAY & FRIDAY)

TODAY'S LESSON

Classes

(Creating new types of objects to help with encapsulation)

Book Chapters 15, 16, 17

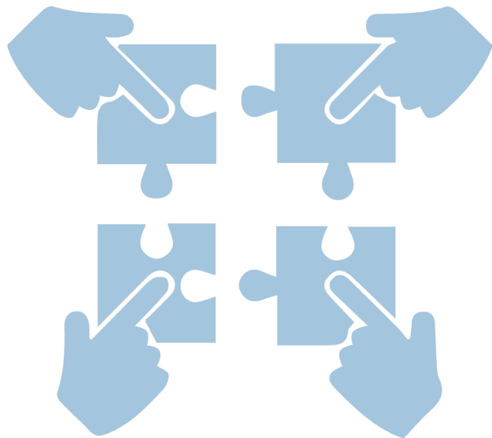
SO INCREDIBLY HELPFUL

Step through it!!!!

Highly, highly, extremely recommended

POGIL Activity 24 – Classes: Attributes

- Look at Python Activity 24, Questions 1-4
- Find a partner and talk through the questions together



POGIL – Activity 24: Question 1

1. Examine the following code from interactive python below using a Flower data structure.

Interactive Python

```
3 >>> iris = Flower()
4 >>> iris.petals = 3
5 >>> iris.petals
6 3
7 >>> iris.color = 'purple'
8 >>> iris.color
```

```
10 flwrList = list()
11 flwrList = [iris]
```

a. What type of object is `flwrList`? How do you know?

____

b. What type of object is `iris`? How do you know?

____

c. On which line do we place `flwrList` on the lefthand side of an assignment operator?

What value is assigned? ____

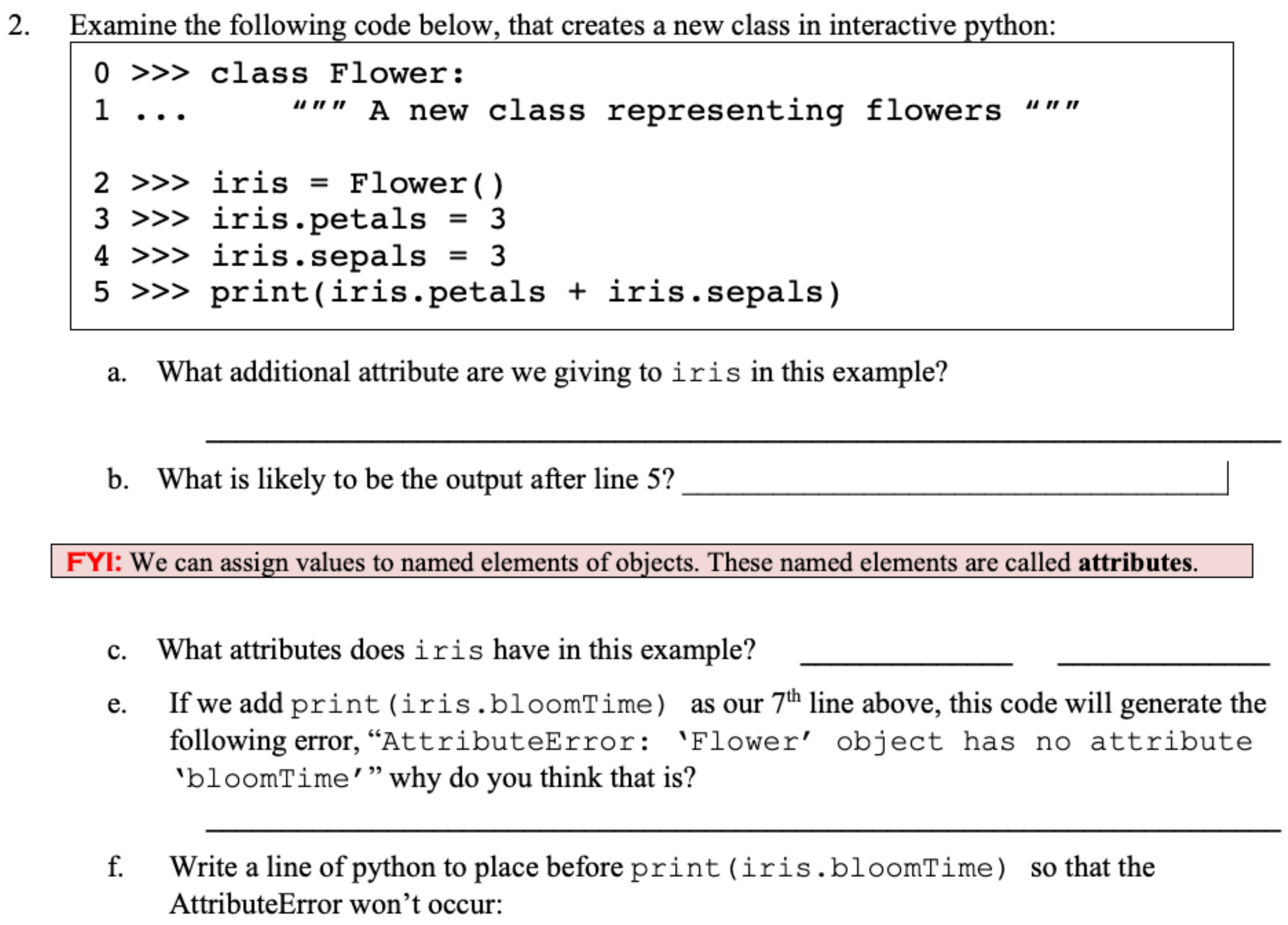
d. On which line is `iris.petals` on the lefthand side of an assignment operator?

What value is assigned? ____

e. What is displayed when we call `iris.petals`? ____________________

f. What will be displayed when we call `iris.color`? ____________________

POGIL – Activity 24: Question 2



2. Examine the following code below, that creates a new class in interactive python:

```
0 >>> class Flower:
1 ...       """ A new class representing flowers """

2 >>> iris = Flower()
3 >>> iris.petals = 3
4 >>> iris.sepals = 3
5 >>> print(iris.petals + iris.sepals)
```

a. What additional attribute are we giving to `iris` in this example?

____

b. What is likely to be the output after line 5? ______________________________

FYI: We can assign values to named elements of objects. These named elements are called **attributes**.

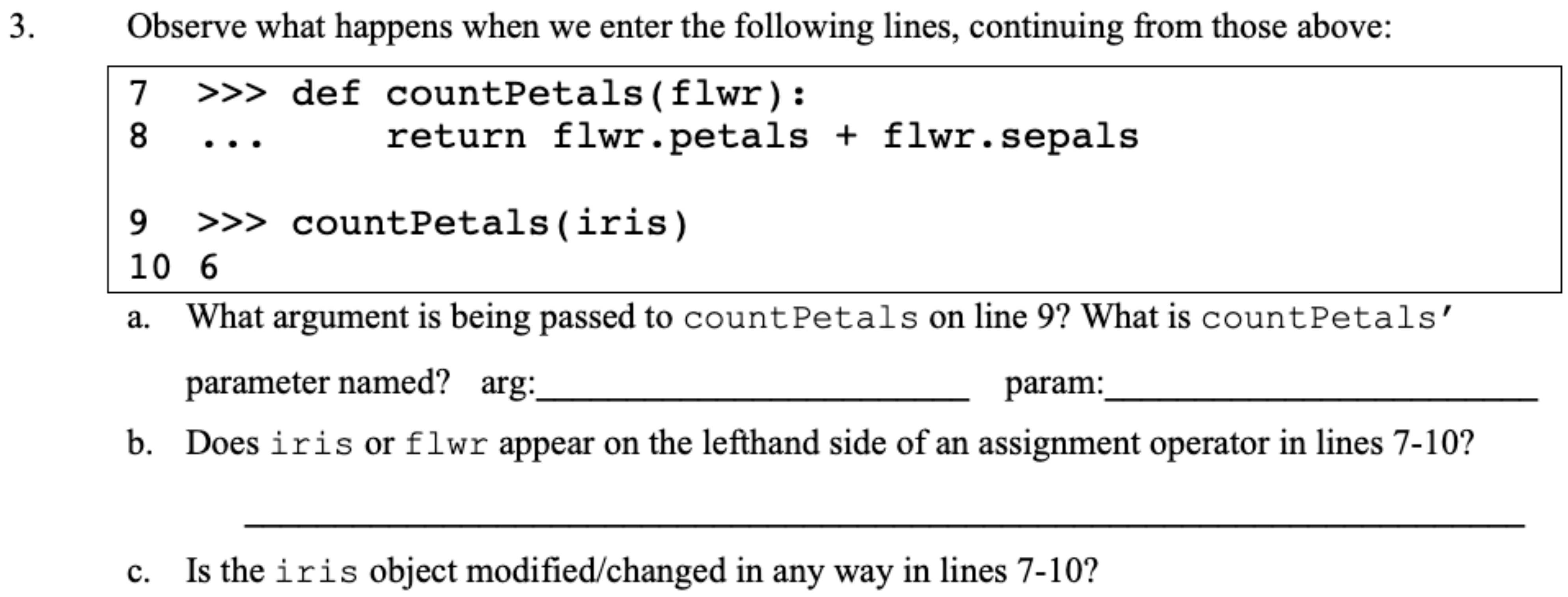
c. What attributes does `iris` have in this example? ____________ ____________

e. If we add `print(iris.bloomTime)` as our 7th line above, this code will generate the following error, "`AttributeError: 'Flower' object has no attribute 'bloomTime'`" why do you think that is?

____

f. Write a line of python to place before `print(iris.bloomTime)` so that the AttributeError won't occur:

POGIL – Activity 24: Question 3



3. Observe what happens when we enter the following lines, continuing from those above:

```
7  >>> def countPetals(flwr):
8  ...      return flwr.petals + flwr.sepals

9  >>> countPetals(iris)
10 6
```

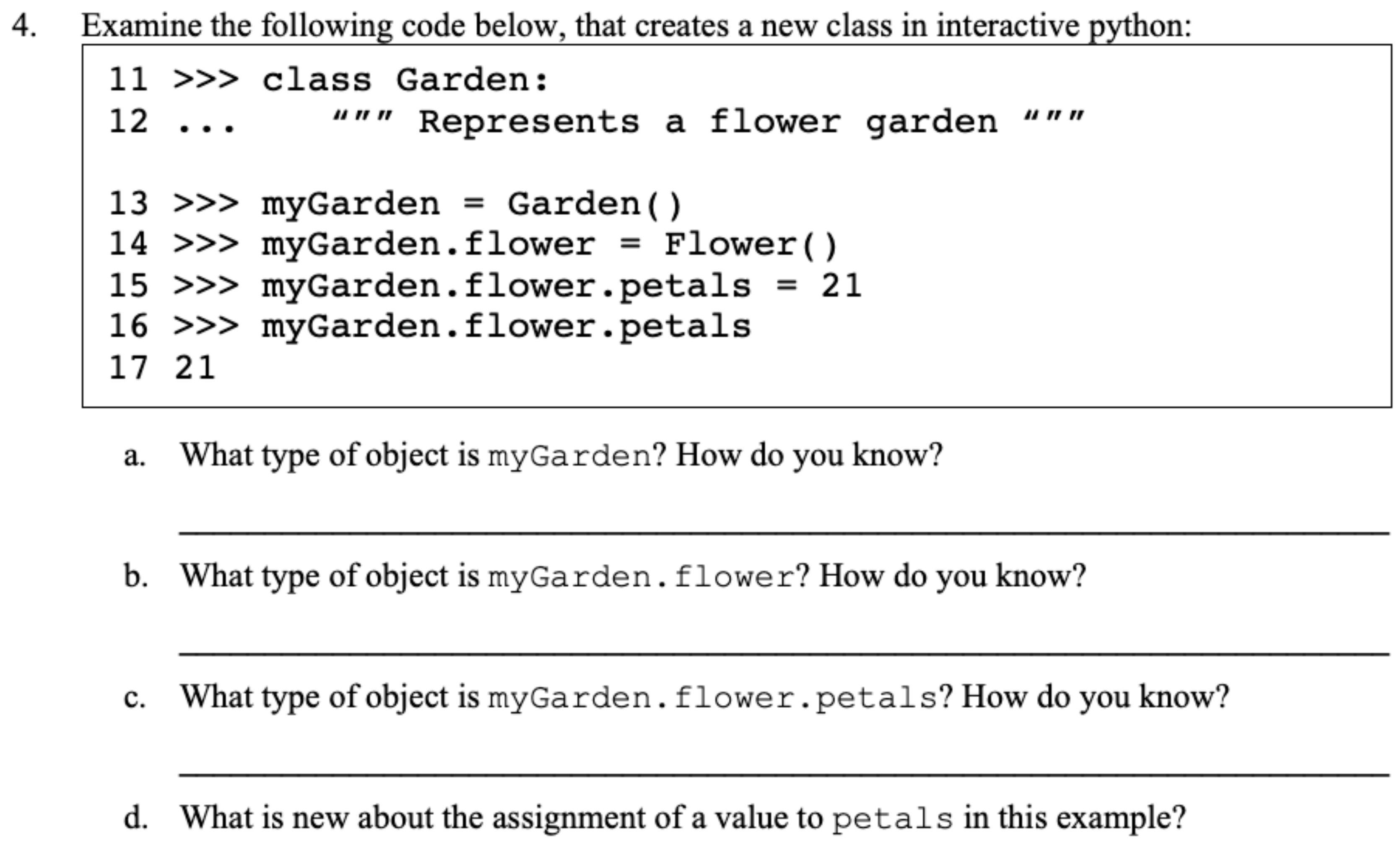
a. What argument is being passed to `countPetals` on line 9? What is `countPetals'` parameter named? arg:________________________ param:________________________

b. Does `iris` or `flwr` appear on the lefthand side of an assignment operator in lines 7-10?

____

c. Is the `iris` object modified/changed in any way in lines 7-10?

POGIL – Activity 24: Question 4



4. Examine the following code below, that creates a new class in interactive python:

```
>>> class Garden:
...     """ Represents a flower garden """

>>> myGarden = Garden()
>>> myGarden.flower = Flower()
>>> myGarden.flower.petals = 21
>>> myGarden.flower.petals
21
```

a. What type of object is `myGarden`? How do you know?

____

b. What type of object is `myGarden.flower`? How do you know?

____

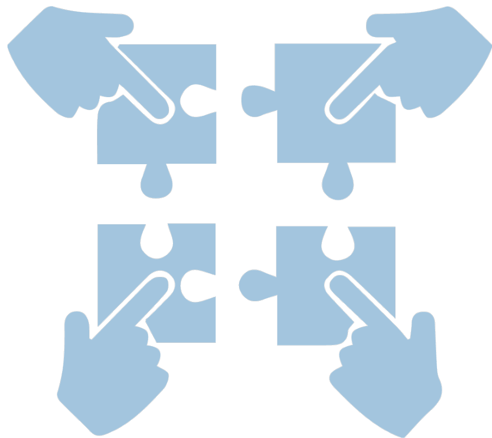
c. What type of object is `myGarden.flower.petals`? How do you know?

____

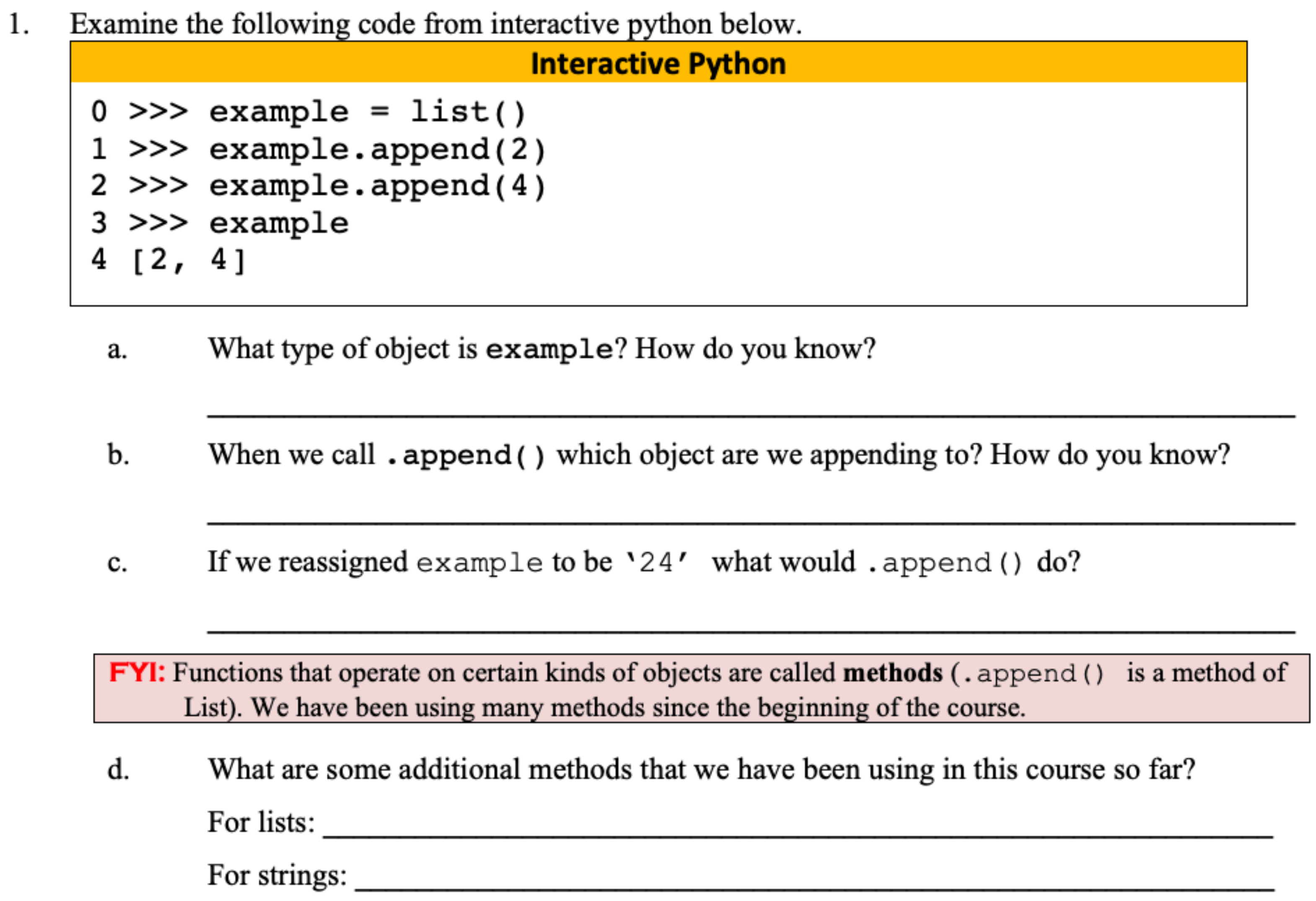
d. What is new about the assignment of a value to `petals` in this example?

POGIL Activity 25b – Classes: Methods

- Look at Python Activity 25b, Questions 1-5
- Find a partner and talk through the questions together



POGIL – Activity 25b: Question 1



1. Examine the following code from interactive python below.

Interactive Python

```
>>> example = list()
>>> example.append(2)
>>> example.append(4)
>>> example
[2, 4]
```

a. What type of object is `example`? How do you know?

____

b. When we call `.append()` which object are we appending to? How do you know?

____

c. If we reassigned `example` to be `'24'` what would `.append()` do?

____

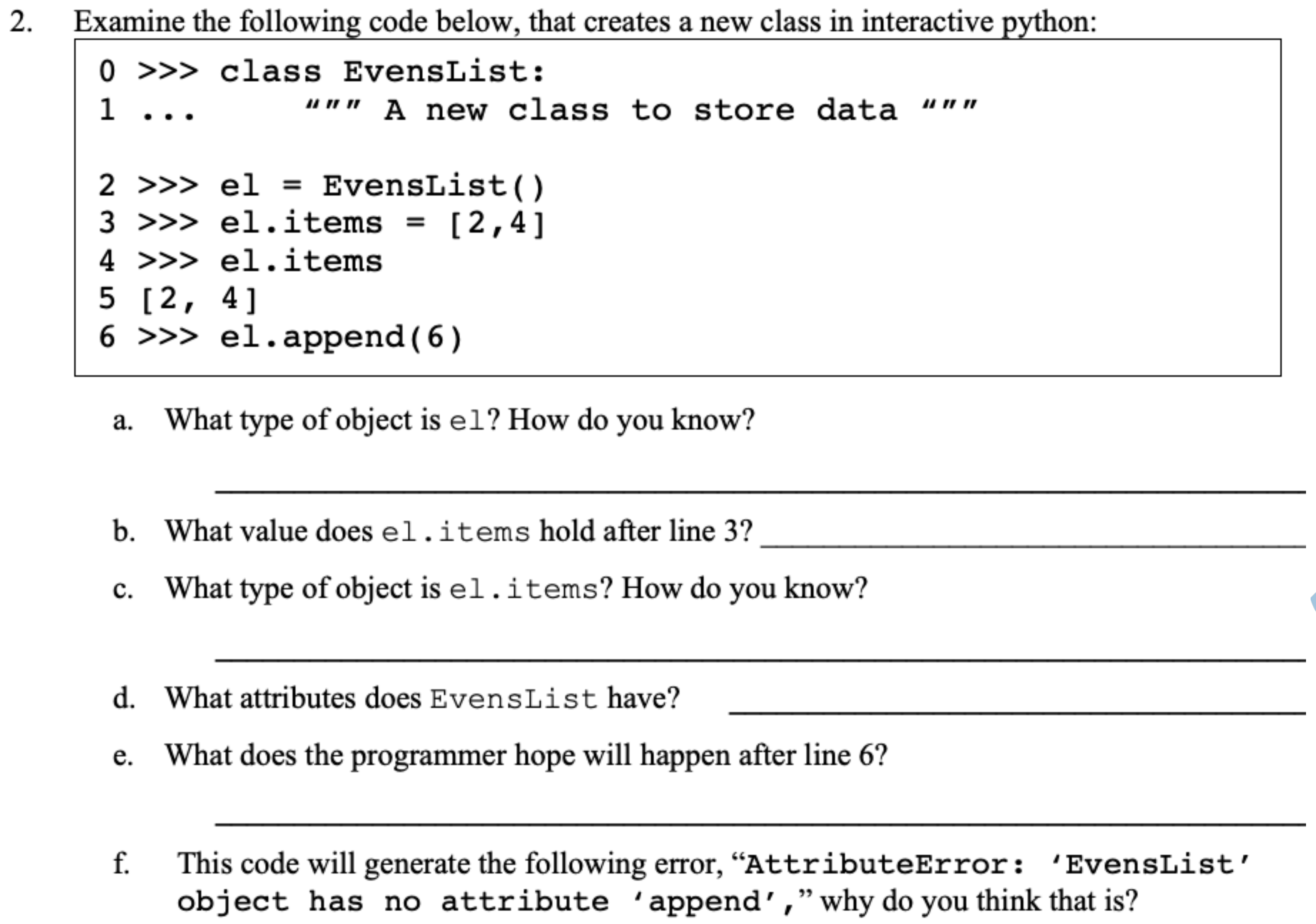
FYI: Functions that operate on certain kinds of objects are called **methods** (`.append()` is a method of List). We have been using many methods since the beginning of the course.

d. What are some additional methods that we have been using in this course so far?

For lists: ____

For strings: ____

POGIL – Activity 25b: Question 2



2. Examine the following code below, that creates a new class in interactive python:

```
0 >>> class EvensList:
1 ...      """ A new class to store data """

2 >>> el = EvensList()
3 >>> el.items = [2,4]
4 >>> el.items
5 [2, 4]
6 >>> el.append(6)
```

a. What type of object is `el`? How do you know?

____

b. What value does `el.items` hold after line 3? ______________________

c. What type of object is `el.items`? How do you know?

____

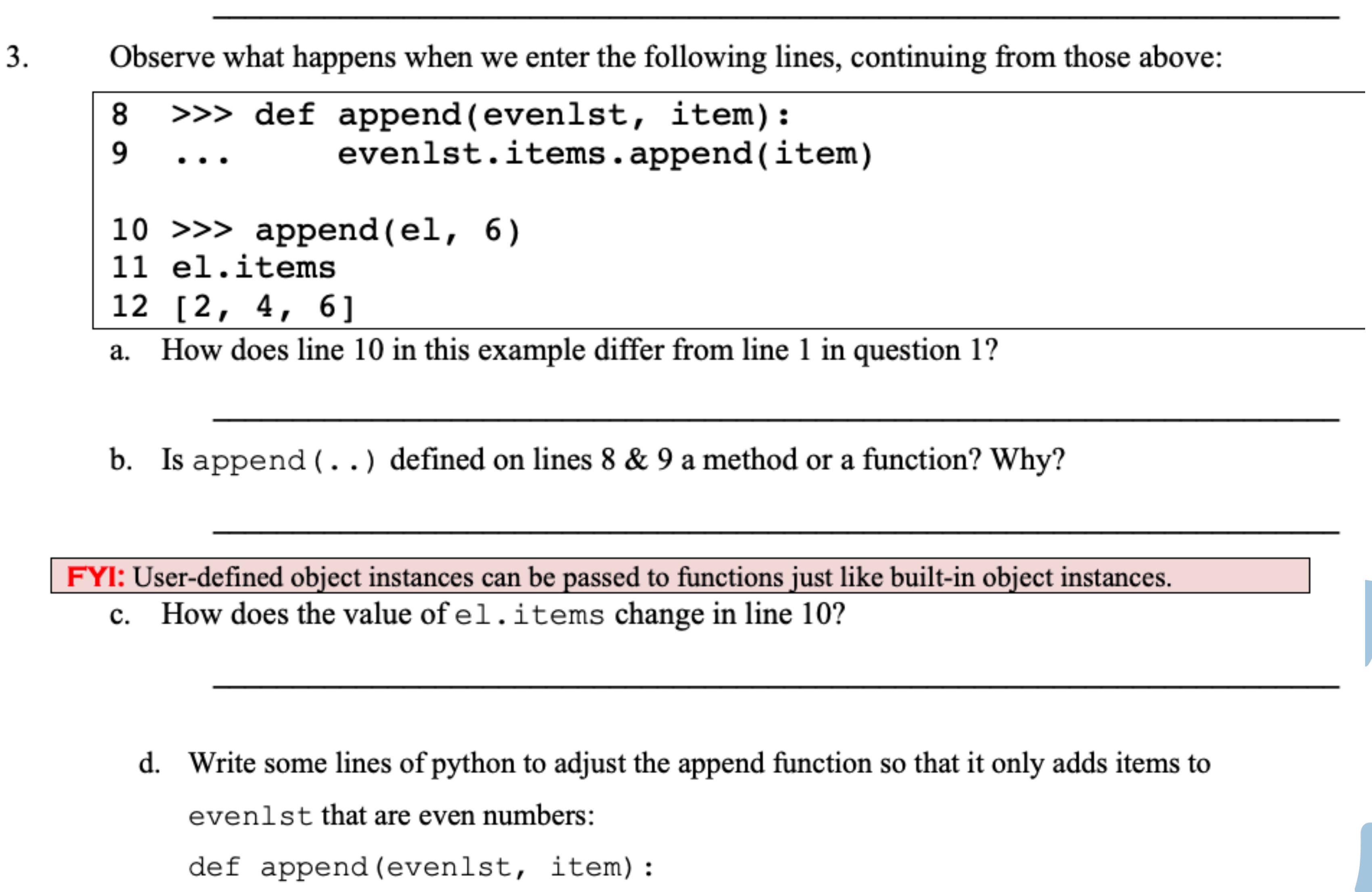
d. What attributes does `EvensList` have? ______________________

e. What does the programmer hope will happen after line 6?

____

f. This code will generate the following error, "**`AttributeError: 'EvensList' object has no attribute 'append',`**" why do you think that is?

POGIL – Activity 25b: Question 3



3. Observe what happens when we enter the following lines, continuing from those above:

```
8  >>> def append(evenlst, item):
9  ...     evenlst.items.append(item)

10 >>> append(el, 6)
11 el.items
12 [2, 4, 6]
```

a. How does line 10 in this example differ from line 1 in question 1?

b. Is `append(..)` defined on lines 8 & 9 a method or a function? Why?

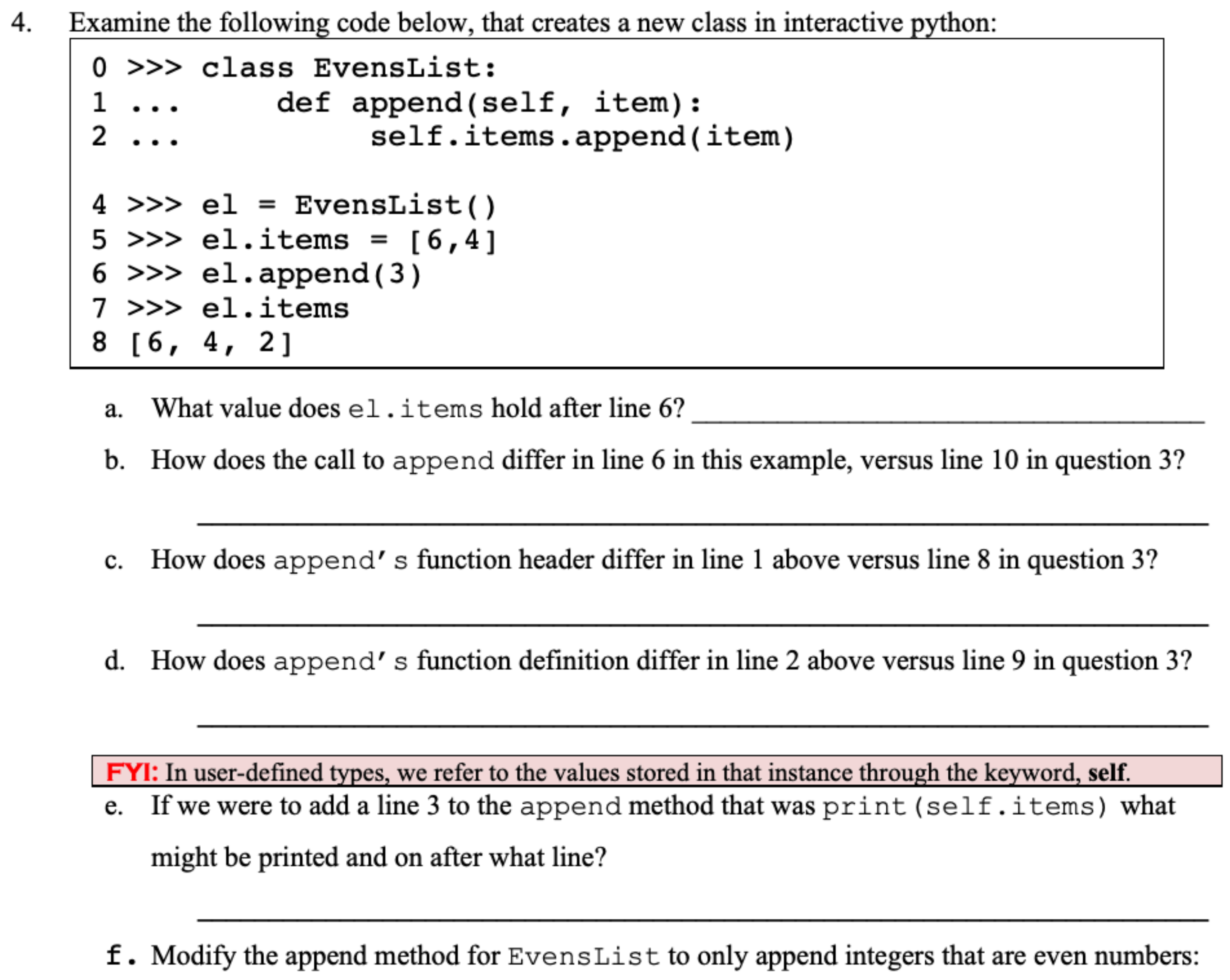
FYI: User-defined object instances can be passed to functions just like built-in object instances.

c. How does the value of `el.items` change in line 10?

d. Write some lines of python to adjust the append function so that it only adds items to `evenlst` that are even numbers:

```
def append(evenlst, item):
```

POGIL – Activity 25b: Question 4



4. Examine the following code below, that creates a new class in interactive python:

```
0 >>> class EvensList:
1 ...     def append(self, item):
2 ...         self.items.append(item)

4 >>> el = EvensList()
5 >>> el.items = [6,4]
6 >>> el.append(3)
7 >>> el.items
8 [6, 4, 2]
```

a. What value does `el.items` hold after line 6? ______________________________

b. How does the call to `append` differ in line 6 in this example, versus line 10 in question 3?

____

c. How does `append's` function header differ in line 1 above versus line 8 in question 3?

____

d. How does `append's` function definition differ in line 2 above versus line 9 in question 3?

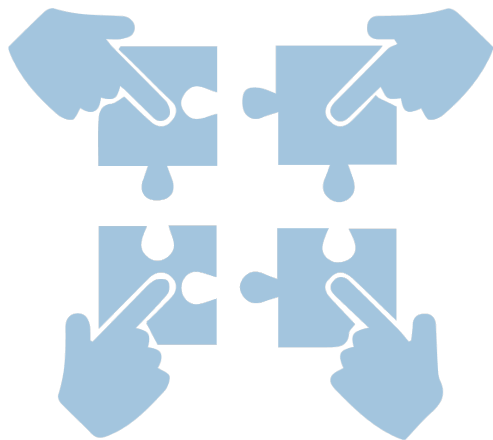
____

FYI: In user-defined types, we refer to the values stored in that instance through the keyword, **self**.

e. If we were to add a line 3 to the `append` method that was `print(self.items)` what might be printed and on after what line?

____

f. Modify the append method for `EvensList` to only append integers that are even numbers:



POGIL – Activity 25b: Question 5

5. Examine the following code below, that creates a different version of `EvensList`, but as a script:

EvensList.py

```
0 class EvensList:
1     def __init__(self, itemList):
2         self._items = itemList
3     def append(self, item):
4         self._items.append(item)

5 if __name__ == ‘__main__’:
6     el = EvensList([88, 12, 4])
7     print(el._items)
8     # prints [88, 12, 4]
9     el._items.append(8)
10    print(el._items)
```

a. What two lines did we add to this definition of `EvensList` that we did not see in the previous question?

b. How does our creation & assignment of the `el` variable differ in this example from `betterEL` in the previous example?

FYI: The __**init**__ method is *implicitly* called when you instantiate a new object. It is very useful for setting up an object with an initial state or initial values.

c. What's stored in `el._items` when line 7 is printed?

d. What's stored in `el._items` after line 9 is executed?

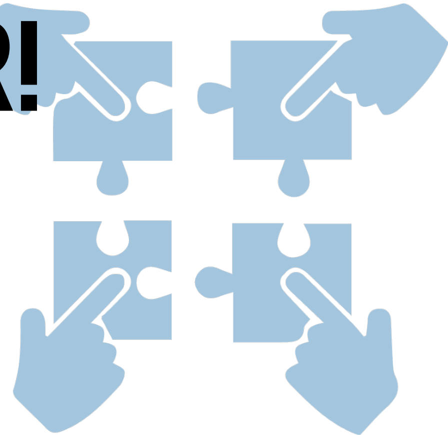
The underscore _ in python

- In python, objects that start with an underscore are “hidden”
  - They’re not really hidden, but it’s a convention to imply that they shouldn’t be accessed publicly
  - If you’re using an object name that starts with an underscore outside of a class definition, you should feel **GUILTY**
  - This goes for double-underscore __<name>__ objects in python too!
- Using a variable name that is an underscore, means you don’t plan to ever use that variable:
  - `for _ in range(5):`
    - `print(“Hello repeat!”)`

YOU SHOULD COMPLETE THE REST OF ALL POGILS OUTSIDE OF CLASS.

BEST DONE WITH A PARTNER OR STUDY GROUP.

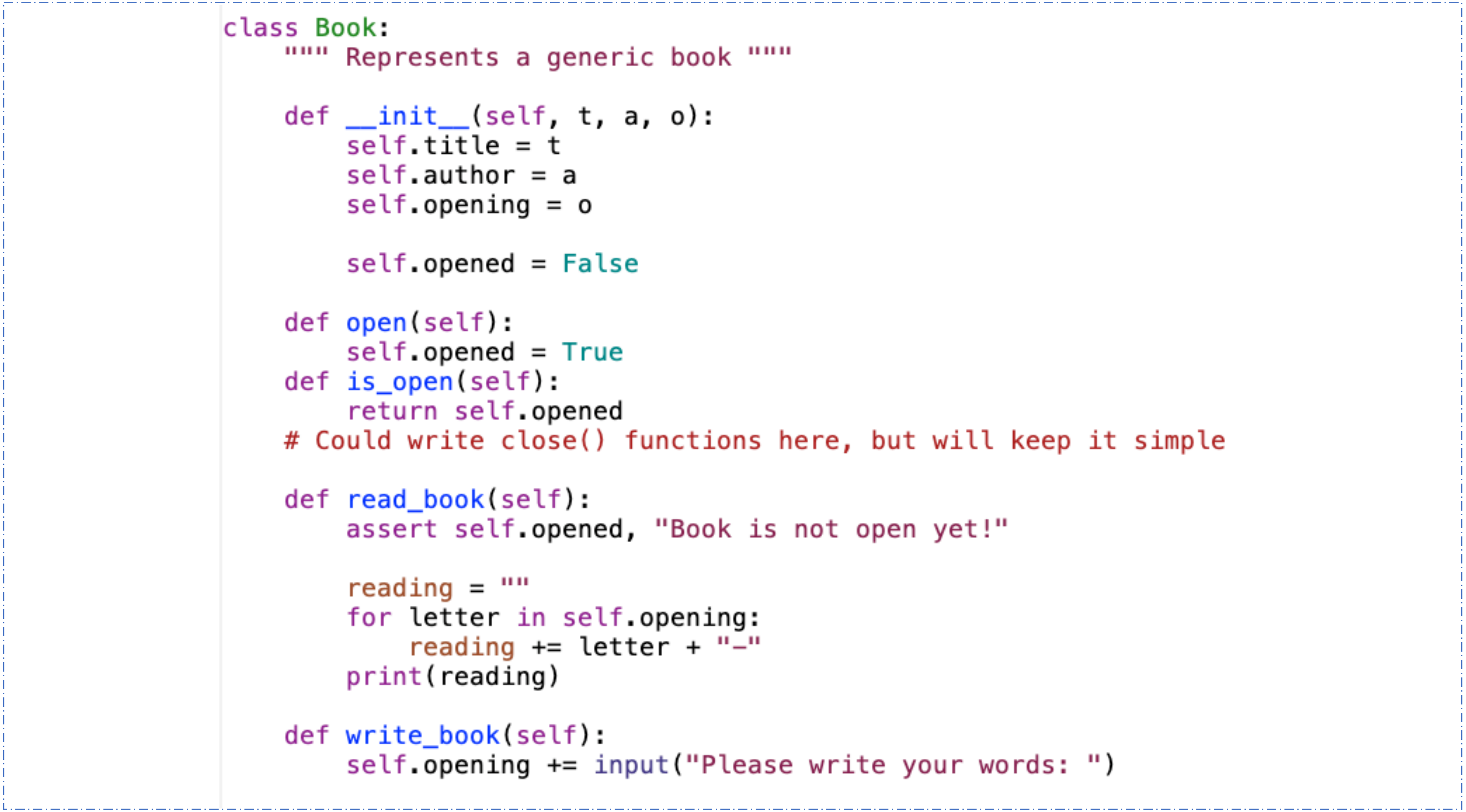
CHECK YOUR ANSWERS ON A COMPUTER!



QUESTIONS?

Leftover Slides

Classes, Objects – See Example Code



```python
class Book:
    """ Represents a generic book """

    def __init__(self, t, a, o):
        self.title = t
        self.author = a
        self.opening = o

        self.opened = False

    def open(self):
        self.opened = True
    def is_open(self):
        return self.opened
    # Could write close() functions here, but will keep it simple

    def read_book(self):
        assert self.opened, "Book is not open yet!"

        reading = ""
        for letter in self.opening:
            reading += letter + "-"
        print(reading)

    def write_book(self):
        self.opening += input("Please write your words: ")
```

Everything in Python is an object!

Everything in Python is an Object

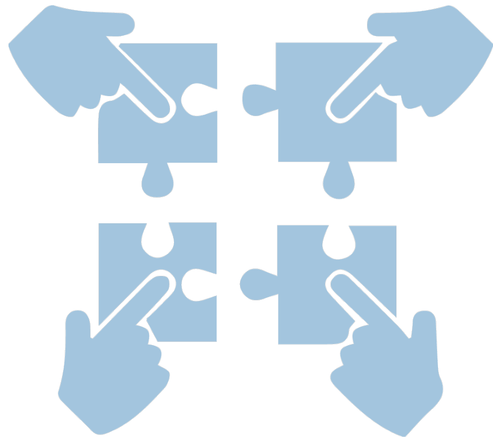
- Even functions!

```
def do_something():
   return 'hello world'
def run_this_func(new_func):
   result = new_func()
   return result

run_this_func(do_something)
```

POGIL Activity 24b – Classes: Slots

- Look at Python Activity 24b, Questions 1-5
- Find a partner and talk through the questions together



POGIL – Activity 24b: Question 1

1. Examine the following code from interactive python below using a Flower data structure.

Interactive Python

```
0 >>> class Flower:
1 ...     """ A new class representing flowers """

2 >>> iris = Flower()
3 >>> iris.petals = 3
4 >>> iris.petals
5 3
6 >>> iris.bloomTime
7 AttributeError: 'Flower' object has no attribute
'bloomTime'
```

a. What type of object is `iris`? How do you know?

____

b. On which line is `iris.petals` on the lefthand side of an assignment operator?

What value is assigned? ____

c. On which line is `iris.bloomTime` on the lefthand side of an assignment operator?

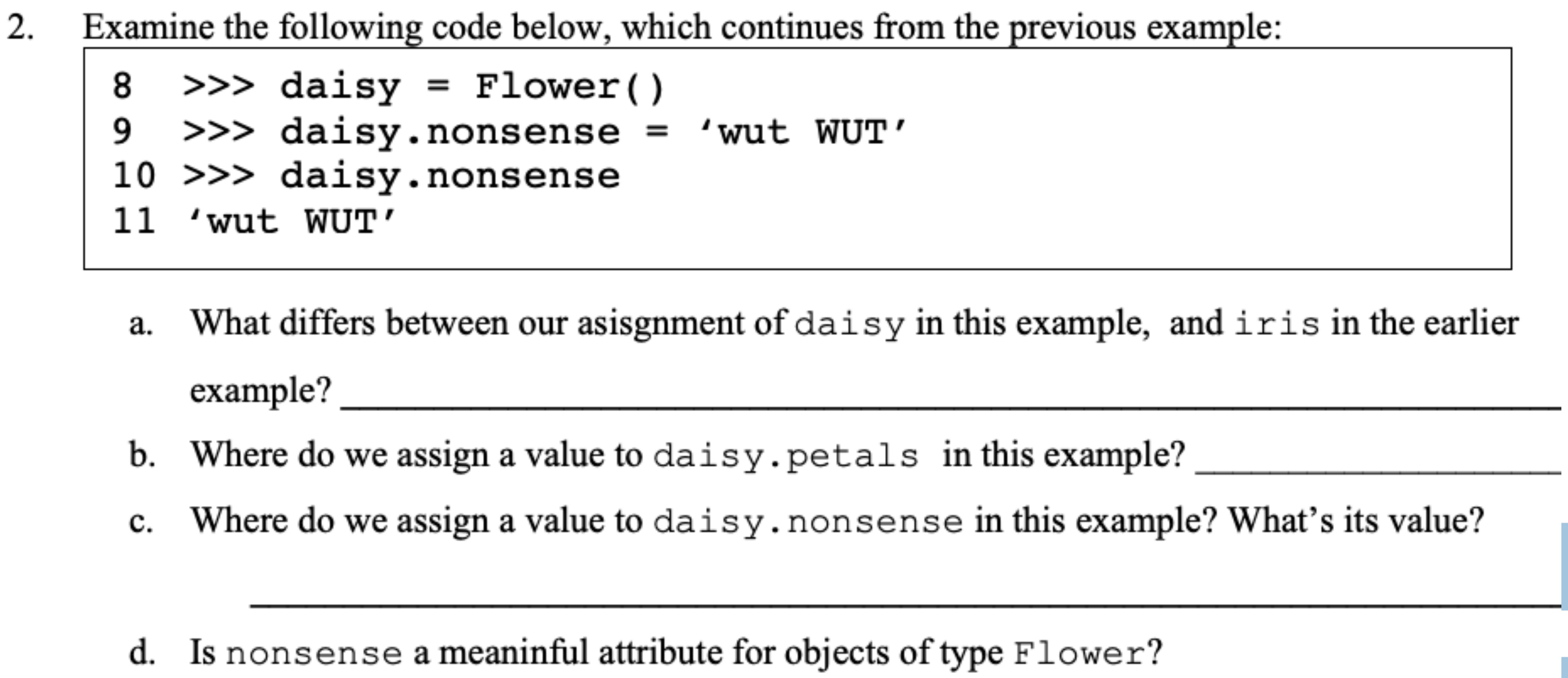
____

d. Why might `iris.bloomTime` on line 7 throw an error?

____

e. Write a line of python to enter before line 6, to fix the error:

POGIL – Activity 24b: Question 2



2. Examine the following code below, which continues from the previous example:

```
>>> daisy = Flower()
>>> daisy.nonsense = 'wut WUT'
>>> daisy.nonsense
'wut WUT'
```

a. What differs between our asisgnment of `daisy` in this example, and `iris` in the earlier example? ______________________________

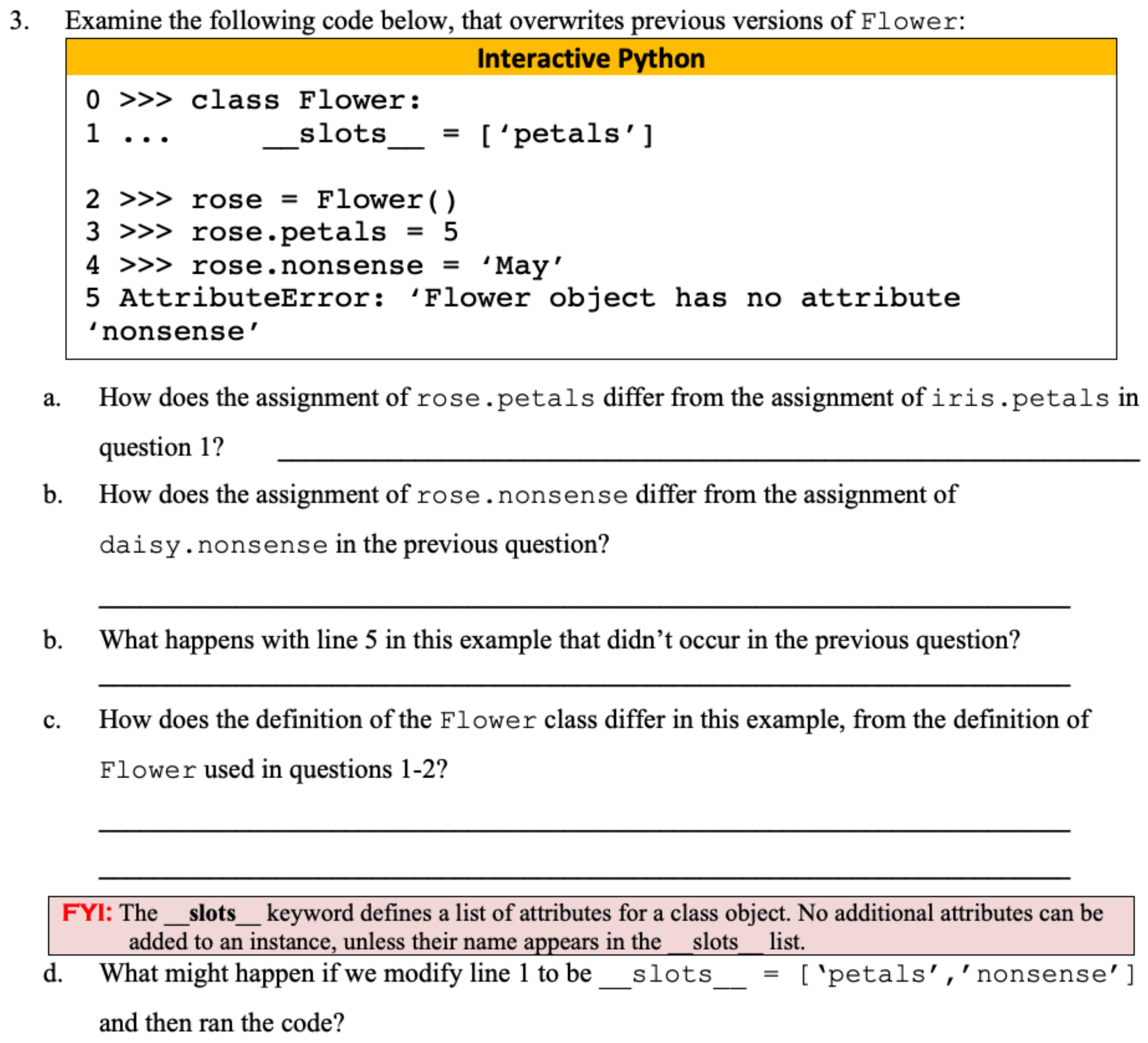
b. Where do we assign a value to `daisy.petals` in this example? ______________

c. Where do we assign a value to `daisy.nonsense` in this example? What's its value?

____

d. Is `nonsense` a meaninful attribute for objects of type `Flower`?

POGIL – Activity 24b: Question 3



3. Examine the following code below, that overwrites previous versions of `Flower`:

Interactive Python

```
0 >>> class Flower:
1 ...       __slots__ = [‘petals’]

2 >>> rose = Flower()
3 >>> rose.petals = 5
4 >>> rose.nonsense = ‘May’
5 AttributeError: ‘Flower object has no attribute
‘nonsense’
```

a. How does the assignment of `rose.petals` differ from the assignment of `iris.petals` in question 1? ____

b. How does the assignment of `rose.nonsense` differ from the assignment of `daisy.nonsense` in the previous question?

____

b. What happens with line 5 in this example that didn't occur in the previous question?

____

c. How does the definition of the `Flower` class differ in this example, from the definition of `Flower` used in questions 1-2?

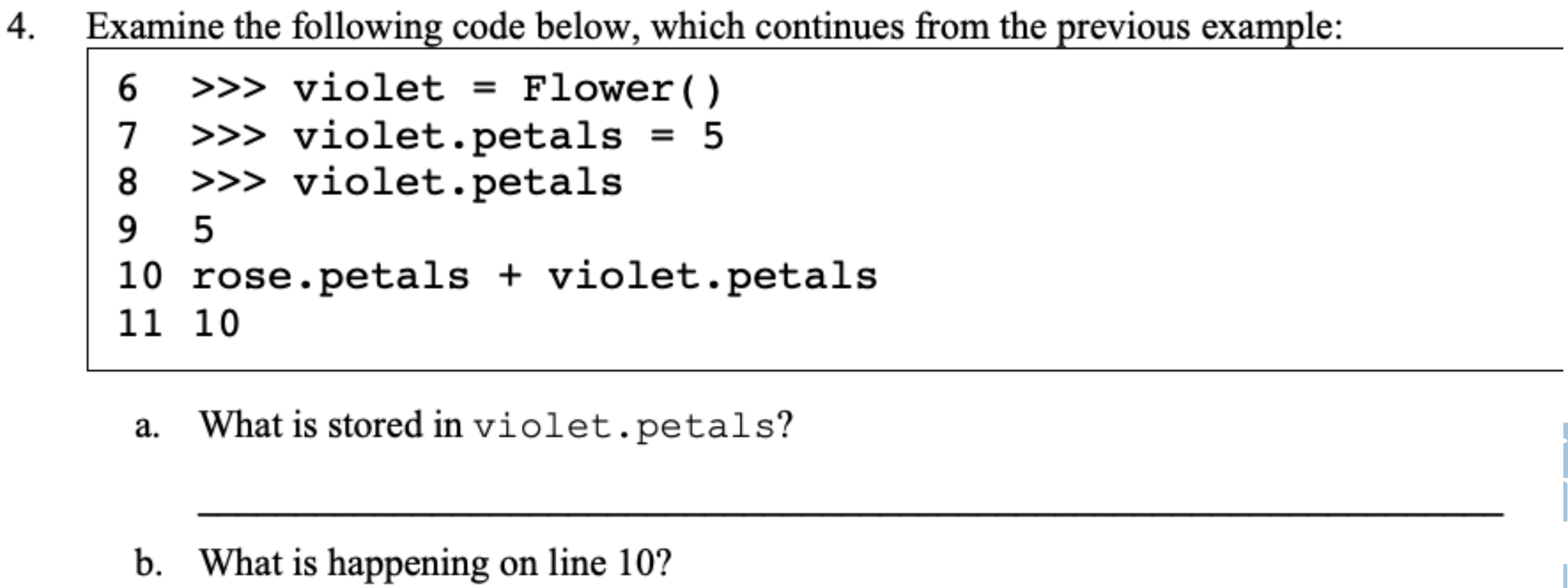
____

____

FYI: The __**slots**__ keyword defines a list of attributes for a class object. No additional attributes can be added to an instance, unless their name appears in the __slots__ list.

d. What might happen if we modify line 1 to be `__slots__ = [‘petals’,’nonsense’]` and then ran the code?

POGIL – Activity 24b: Question 4



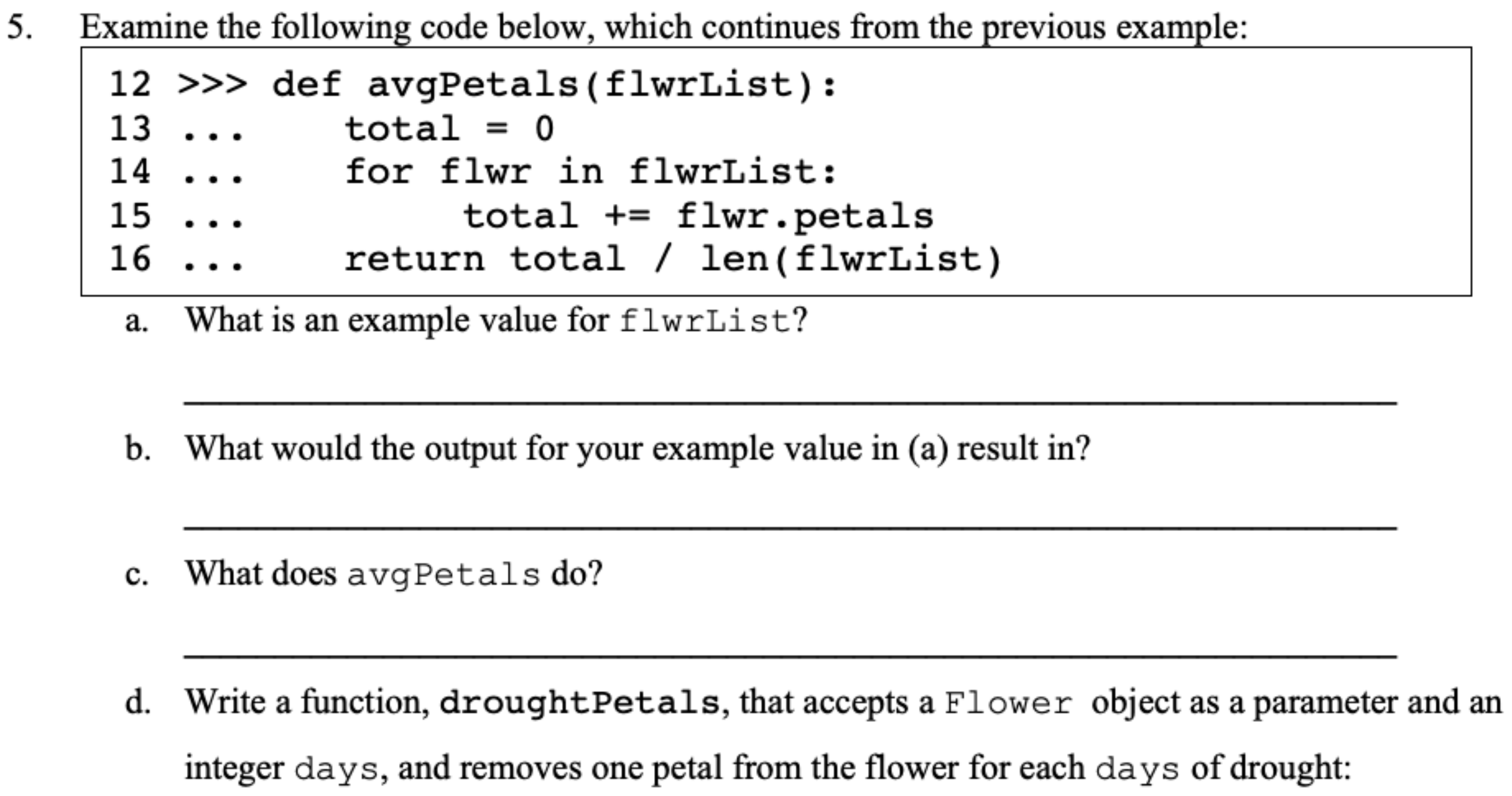
4. Examine the following code below, which continues from the previous example:

```
6   >>> violet = Flower()
7   >>> violet.petals = 5
8   >>> violet.petals
9   5
10  rose.petals + violet.petals
11  10
```

a. What is stored in `violet.petals`?

b. What is happening on line 10?

POGIL – Activity 24b: Question 4



5. Examine the following code below, which continues from the previous example:

```
>>> def avgPetals(flwrList):
...     total = 0
...     for flwr in flwrList:
...         total += flwr.petals
...     return total / len(flwrList)
```

a. What is an example value for `flwrList`?

____

b. What would the output for your example value in (a) result in?

____

c. What does `avgPetals` do?

____

d. Write a function, **`droughtPetals`**, that accepts a `Flower` object as a parameter and an integer `days`, and removes one petal from the flower for each `days` of drought:

Class Syntax

We’re defining a new type of object

```
class Book:
    __slots__ = [‘_title’]
    def __init__(self):
        self._title = ‘’
    def addTitle(self, txt):
        self._title += txt
>>> b = Book()
>>> b._title
‘’
>>>b.addTitle(“Harry Potter”)
>>> b._title
‘Harry Potter’
```

The name of the new type

Only attribute for Book is ‘_title’

Initializer is implicitly called when we create a new Book

Methods must always be passed self as parameter

Object attributes are always accessed through self.

Makes a new book, implicitly calls __init__()

If init() weren’t called, this would throw an error!

Even though method definition has self, method call does not!

_title starts with underscore, so we shouldn’t use it!

There’s something else we should use instead…