Classes are blueprints or designs while an *object* in an instantiation of a class.

1. Compare instance vs. static attributes:

attribute that exists for each object

like a global variable for a class

	Instance	Static
Where do you declare it?	in the constructor! ↓ def __init__(): self.attributes	NOT in constructor

How do you call/ reference it?	self.attributeName	class Person(object):
		NUM_PEOPLE = 0
		def __init__():
		Person.NUM_PEOPLE += 1

How many of these attributes exist *per class*?	However many objects have been instantiated	1

2. What is wrong with the following piece of code?

```
class Student(object):

    def __init__(self, name, age):

        self.__name = name

        self.__year = ~~year~~ age

    def getTimeUntilGraduation(self):

        self.__timeUntilGraduation = 4 – self.__year

        return self.__timeUntilGraduation
```

needs to be defined in the constructor!!

instance attribute

all different:
- year → local var
- self.year → instance var
- self.__year → private instance var

3. Which of the following are true about dictionaries? If the statement is false, correct it to make it true.

   i. Any type can act as a key. → False, keys must be immutable

   ii. Keys must be unique. True (Values? Not unique)

   iii. Writing to a key that already exists just appends to its value. ↳ False → the value is overwritten

   iv. There is no need to check if a key exists before reading from it.

   ↳ False, if you try to get a key that doesn't exist you will get an error

4. Which of the following is NOT a class you have used?

   i. List

   ii. String

   iii. Int

   iv. Dictionary

```
def printList(someList):
    for x in someList:
        thingToPrint = x.upper()
        print(thingToPrint)
```

getName()

Imagine someList are strings and you want to print upper. Easy. Imagine someList has Students instead. Same thing!

5. When creating a child class, what is the first thing you must do in the constructor?

Parent Class

```
class Person(object):
    def __init__(self, name):
        self.__name = name
```

Child Class

```
class Student(Person):
    def __init__(self, name, gpa)
        Super().__init__(name)
        self.__gpa = gpa
```

MUST call parent constructor

6. What is the purpose of getters and setters? When do you need them? Write them.

Need them when attributes are private, otherwise you have no way of accessing that data.

Getter: → no parameters

```
def getName(self):
    return self.__name
```

Setter: → never returns

```
def setName(self, newName):
    self.__name = newName
```

Coding Questions

7. There is a dictionary where the key is a student's name (you may assume these are unique), and the value is a list of scores that student has for ITP 115. You also have access to a function called avg(list) that takes in 1 parameter, a list of integers or floats ~~doubles~~ and returns the average of that list. Write a function that takes in the dictionary and the name of a student as its two parameters, and returns the students average score in ITP 115. If the student is not in ITP 115, return -1.

```
def getScore(classDict, name):
    if name not in classDict.keys():
        return -1
    scoreList = classDict[name]
    grade = avg(scoreList)
    return grade
```

(if you needed to go though all students)

```
for student in classDict.keys():
    scoreList = classDict[student]
```

key

8. Every student has a name, a GPA, and a tuition they pay. The standard tuition is $50,000. However, in-state students receive a flat decrease of $10,000 in tuition. Out-of-state students do not receive this flat decrease, but their tuition does go down based on their academic performance. Specifically, they receive $1000 off their tuition for every GPA point they have (for example, an out-of-state student with a 3.14 GPA would receive a decrease of $3,140). Write a base Student class, and two classes, InStateStudent and OutOfStateStudent, that inherit from Student and calculate how much each student will have to pay in tuition.

Base

```
class Student(object):
    def __init__(self, name, gpa)
        self.__name = name
        self.__gpa = gpa
        self.__tuition = 50000

    def calcTuition(self):
        return self.__tuition

    def getGPA(self):
        return self.__gpa
```

private attributes not accessible to child classes

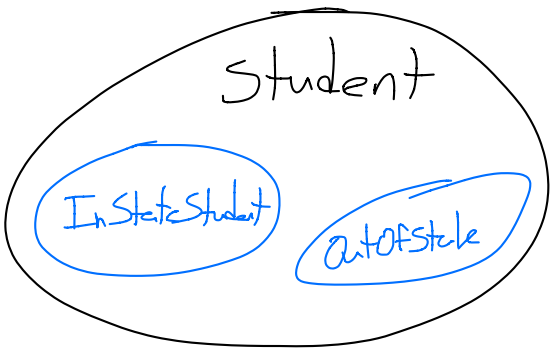
Child

```
class InStateStudent(Student):
    def __init__(self, name, gpa):
        super().__init__(name, gpa)

    def calcTuition(self):
        tuition = super().calcTuition()
                  - 10000
        return tuition
```

need parenthesis

```
class OutOfStateStudent(Student):
    def __init__(self, name, gpa)
        Super().__init__(name, gpa)

    def calcTuition(self):
        tuition = super().calcTuition()
                  -(1000 * self.getGPA())
        return tuition
```

need parenthesis



Multi-files

From [filename] import [class name]