

Classes and OOP

adapted from material by Mike Scott and Bill
Young at the University of Texas at Austin


Think back to averaging student grades

```
[ "student1", 25, 50 ]
```

We did this by accessing `entry[1]` and `entry[2]`.

Now suddenly, our format changes!

```
[ "student1", "cs119" 25, 50 ]
```



New! The class the scores are from.

```
In [1]: x = [ "a1", "cs119", 25, 50 ]
```

```
In [2]: mean = x[1] + x[2] / 2
```

```
-----  
TypeError                                Traceback (most recent call last)
```

```
Input In [2], in <cell line: 1>()
```

```
----> 1 mean = x[1] + x[2] / 2
```

```
TypeError: can only concatenate str (not "float") to str
```

What went wrong?

We were using `entry[1]` as shorthand for "student's midterm grade", but there's no reason why it has to be at entry 1!

Let's bundle related data together in a meaningful manner!

```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
```

```
1 entry1 = [ "Student1", 99, 98 ]
2 entry2 = [ "Student2", "cs119", 98, 99 ]
3
4 grade1 = StudentGrade()
5 grade1.name = entry1[0]
6 grade1.exam1_grade = entry1[1]
7 grade1.exam2_grade = entry1[2]
8
9 grade2 = StudentGrade()
10 grade2.name = entry2[0]
11 grade2.exam1_grade = entry2[2] # Different!
12 grade2.exam2_grade = entry2[3]
```



Object Oriented Programming

Basic idea: turn your program into a *collection of object*. An object has two things:

- Data which characterizes its current state
- A set of actions (called *methods*) that it can perform

In a *pure* object-oriented system, the only way that a programmer interacts with objects is by calling its methods.

These things that belong to a class are often called *class members*.

Let's equip our StudentGrade class with some methods!

Methods are written like normal functions, with two extra rules:

- They have to be indented the same amount as other class members
- Their first argument **has to** be the special word "self".

First method: calculate grade average

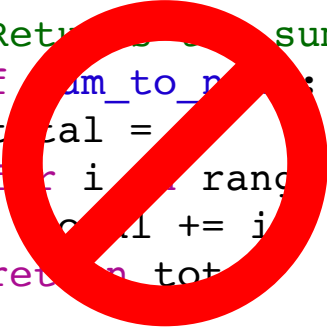
```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
5
6     def calculate_avg(self):
7         return self.exam1_grade + self.exam2_grade
8
9     def from_values(self, name, e1g, e2g):
10        self.name = name
11        self.exam1_grade = e1g
12        self.exam2_grade = e2g
```



Why Objects?

Hide Details!

```
1 # Returns the sum of values from 1 to n.  
2 def sum_to_n(n):  
3     total = 0  
4     for i in range(1, n+1):  
5         total += i  
6     return total
```



```
1 # Returns the sum of values from 1 to n.  
2 def sum_to_n(n):  
3     return (n + 1) * n // 2
```

```
1 def main():  
2     max_num = 1000000  
3     max_sum = sum_to_n(max_num)  
4     print(f"The sum of the first {max_num} numbers is {max_sum}")
```

Enforce Safety!

```
1 num_students = 30
2 for x in range(num_drops):
3     num_students -= 1
```

```
1 mycourse = Course(30)
2 for x in range(num_drops):
3     mycourse.drop_student()
```

**A natural way to approach
some problems**

Bank Account

Design a bank account class.

In the USA, the FDIC insures all accounts up to \$250,000.



Special Methods

Python gives us a few special methods that we can implement. These are used *implicitly* in other contexts.

__init__

The `__init__` method is called when the object is initialized. It replaces the default initializer.

```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
5
6     def calculate_avg(self):
7         return self.exam1_grade + self.exam2_grade
8
9     def from_values(self, name, e1g, e2g):
10        self.name = name
11        self.exam1_grade = e1g
12        self.exam2_grade = e2g
```

```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
5
6     def calculate_avg(self):
7         return self.exam1_grade + self.exam2_grade
8
9     def __init__(self, name, e1g, e2g):
10        self.name = name
11        self.exam1_grade = e1g
12        self.exam2_grade = e2g
```




```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
5
6     def calculate_avg(self):
7         return self.exam1_grade + self.exam2_grade
8
9     def __init__(self, name, e1g, e2g):
10        self.name = name
11        self.exam1_grade = e1g
12        self.exam2_grade = e2g
```

```
1 ## Before:
```

```
2
```

```
3 g = StudentGrade()
```

```
4 g.from_values(lst[0], lst[1], lst[2])
```

```
5
```

```
6 ## After
```

```
7 g = StudentGrade(lst[0], lst[1], lst[2])
```

```
1 class StudentGrade(object):
2     name = ""
3     exam1_grade = 0
4     exam2_grade = 0
5
6     def calculate_avg(self):
7         return self.exam1_grade + self.exam2_grade
8
9     def __init__(self, name, e1g, e2g):
10        self.name = name
11        self.exam1_grade = e1g
12        self.exam2_grade = e2g
13
14    def __str__(self):
15        ??
```