

Classes

Announcements

Class Statements

Class Statements Create User-Defined Types

A class statement creates a new type

Idea: All students have a `name` and a midterm score `mt2_points`; the `Student` class should add those attributes to each newly created instance

The `@dataclass` decorator simplifies class statements

```
from dataclasses import dataclass
```

```
@dataclass
class Student:
    name: str
    mt2_points: int
```

```
>>> eve = Student("Eve", 26)
```

A class can describe the behavior of its instances

Idea: All students share a `total_points` method and a `assign_mt2_grade` method

```
>>> bao = Student('Bao', in_mega=False)
>>> bao.total_points()
0
>>> bao.assign_mt2_grade(26)
>>> bao.total_points()
20
```

The Student Class

```
class Student:
```

`__init__` is a special method name for the function that constructs a Student instance

```
def __init__(self, name, in_mega):  
    self.name = name  
    self.discussion_points = 0  
    if in_mega:  
        self.discussion_points = 10  
    self.mt2_points = 0  
    self.a_plus_points = 0
```

`self` is the instance of the Student class on which `total_points` was invoked

```
def total_points(self):  
    return self.discussion_points + self.mt2_points
```

```
def assign_mt2_grade(self, points):  
    self.mt2_points = min(20, points)  
    if points == 26:  
        self.a_plus_points += 5
```

Methods are functions defined in a class statement

`bao` is an *instance* of the Student class

```
>>> bao = Student('Bao', False)  
>>> eve = Student('Eve', False)
```

`mt2_points` is an *attribute*

```
>>> bao.mt2_points  
0
```

`total_points()` is a *method*

```
>>> bao.total_points()  
0  
>>> bao.assign_mt2_grade(26)  
>>> bao.mt2_points  
20  
>>> eve.mt2_points  
0
```

Discussion Question: Midterm Retakes

```
class Student:
    def __init__(self, name, in_mega):
        self.name = name
        self.discussion_points = 0
        if in_mega:
            self.discussion_points = 10
        self.mt2_points = 0
        self.a_plus_points = 0

    def retake_mt2(self, points):
        """ Replace MT2 grade with points, if points is higher than the existing MT2 grade.

        Cap mt2_points at 20, even if points is higher.
        """
        if _____:
            _____
```

Discussion Question: Midterm Retakes

```
class Student:
    def __init__(self, name, in_mega):
        self.name = name
        self.discussion_points = 0
        if in_mega:
            self.discussion_points = 10
        self.mt2_points = 0
        self.a_plus_points = 0

    def retake_mt2(self, points):
        """ Replace MT2 grade with points, if points is higher than the existing MT2 grade.

        Cap mt2_points at 20, even if points is higher.
        """
        if points > self.mt2_points:
            self.mt2_points = min(points, 20)
```

Discussion Question: Create Many Accounts

Write a function **create_students** that takes a list of strings called **regular** that has all of the students in regular discussion, and **mega** that has all of the students in mega discussion. It returns a dictionary in which each name is a key, and its value is a new **Student** with that name.

```
def create_students(regular: list[str], mega: list[str]) -> dict[str, Student]:
```

dictionary

A Student instance

```
>>> students = create_students(["Daniel", "Darren", "Peyton"], ["Terry", "Tejas"])
```

```
>>> students["Darren"].discussion_points
```

```
0
```

```
>>> students["Terry"].discussion_points
```

```
10
```

```
"""
```

```
result = {_____}
```

```
for name in _____:
```

```
    _____
```

```
return result
```

Discussion Question: Create Many Accounts

Write a function **create_students** that takes a list of strings called **regular** that has all of the students in regular discussion, and **mega** that has all of the students in mega discussion. It returns a dictionary in which each name is a key, and its value is a new **Student** with that name.

```
def create_students(regular: list[str], mega: list[str]) -> dict[str, Student]:
```

dictionary

A Student instance

```
>>> students = create_students(["Daniel", "Darren", "Peyton"], ["Terry", "Tejas"])
```

```
>>> students["Darren"].discussion_points
```

```
0
```

```
>>> students["Terry"].discussion_points
```

```
10
```

```
"""
```

```
result = {name: Student(name, False) for name in regular}
```

```
for name in mega:
```

```
    result[name] = Student(name, True)
```

```
return result
```

Animation Demos by Hany Farid