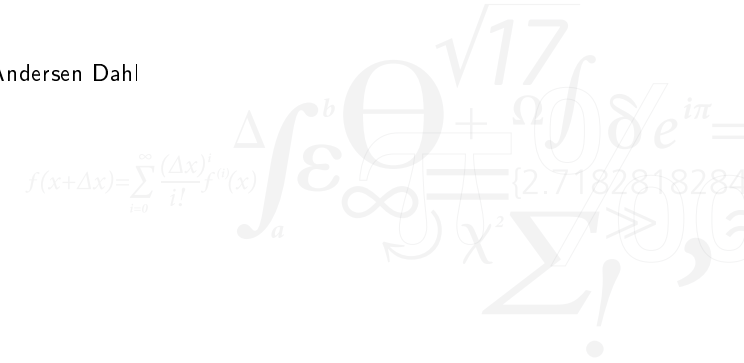
Lecture 10: Classes, part 2

Morten Rieger Hannemose, Vedrana Andersen Dahl

Fall 2023



Today's lecture

1. Continuation about OOP (5 min)
2. A coding demo (40 min)

Continuing where we left off

MyTime

```
class MyTime:

    def __init__(self, hours, minutes):
        self.hours = hours
        self.minutes = minutes

    def print_time(self):
        print(f'{self.hours:02}:{self.minutes:02}')

    def increment_hours(self):
        self.hours += 1
        if self.hours == 24:
            self.hours = 0

    def increment_minutes(self):
        self.minutes += 1
        if self.minutes == 60:
            self.minutes = 0
            self.increment_hours()

my_time = MyTime(23, 55)
for i in range(10):
    my_time.increment_minutes()
    my_time.print_time()
```

Recall from last time

- Attributes and methods bundled together. This is referred to as *encapsulation* and sometimes the access to attributes is restricted.
- Class definition is concerned with the template for objects. Actual objects are made through instantiation.
- Methods have special syntax, in particular the `self` attribute.

Today we continue working with OOP

- The `__str__` method
- Inheritance: defining a child class for `MyTime`
- Operator overloading shown on the `__add__` method

Code shown live during lecture

MyTimeSeconds

```
class MyTimeSeconds(MyTime):
    def __init__(self, hours, minutes, seconds):
        super().__init__(hours, minutes)
        self.seconds = seconds
    def __str__(self):
        return super().__str__() + f":{self.seconds
    :02}"
    def __add__(self, other):
        hours = self.hours + other.hours
        minutes = self.minutes + other.minutes
        seconds = self.seconds + other.seconds
        if seconds >= 60:
            seconds -= 60
            minutes += 1
        if minutes >= 60:
            minutes -= 60
            hours += 1
        if hours >= 24:
            hours -= 24
        return MyTimeSeconds(hours, minutes, seconds
    )

my_time1 = MyTimeSeconds(2, 30, 59)
my_time2 = MyTimeSeconds(2, 30, 59)
print("The time is", my_time1 + my_time2)
```

MyTimeUS

```
class MyTimeUS(MyTimeSeconds):
    def __str__(self):
        if self.hours >= 12:
            AMPM = "PM"
        else:
            AMPM = "AM"
        new_hours = (self.hours - 1) % 12 + 1
        return f"{new_hours:02}:{self.minutes:02}" +
      " " + AMPM

my_time = MyTimeUS(11,00, 00)
print(my_time)
my_time.increment_hours()
print(my_time)
my_time.increment_hours()
print(my_time)
```