03-2 Classes and Functions

CSI 500

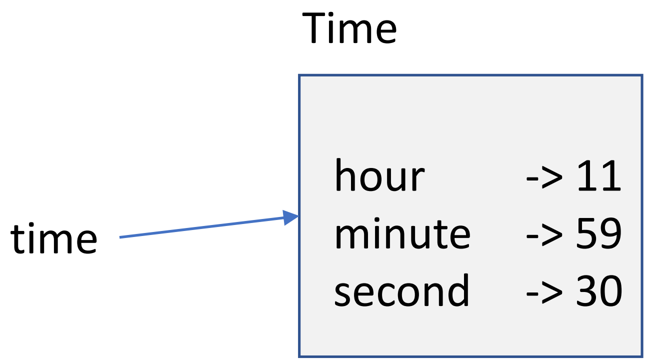
Spring 2018

Course material derived from:
Downey, Allen B. 2012. "Think Python, 2nd Edition". O'Reilly Media Inc., Sebastopol CA.

"How to Think Like a Computer Scientist" by Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers. Oct 2012
http://openbookproject.net/thinkcs/python/english3e/index.html

Time

- Let's build on our experience with user-defined types and build a class to handle "time"
  - attributes for hour, minute, second
- Note:
  - Time is a class
  - time is an object



```
class Time:
    """ Represents time of day
    attibutes: hour, minute, second
    """


time = Time()
time.hour = 11
time.minute = 59
time.second = 30

def print_time( t ):
    print( '%02d : %02d : %02d' % \
    (t.hour, t.minute, t.second)

print_time( time )
11 : 59 : 30
```

Pure functions

- A pure function does not modify objects passed as parameters
  - has no "side effects"
  - may return a value
- Let's build a (buggy) function to add two time values
  - add appropriate attributes
  - return a Time object
- Oops! doesn't handle minute > 60

```
# continue time example
 def add_time( t1, t2 ):
     sum = Time()
     sum.hour = t1.hour + t2.hour
     sum.minute = t1.minute + t2.minute
     sum.second = t1.second + t2.second
     return sum

# test using a start time and a movie run time
# for Monty Python and the Holy Grail
 start = Time()
 start.hour = 9
 start.minute = 45
 start.second = 0

 duration = Time()
 duration.hour = 1
 duration .minute = 35
 duration.second = 0

 done = add_time( start, duration )
 print_time( done )
10 : 80 : 00
```

Pure functions (cont)

- Here's a better implementation
  - compensate for second rollover to min
  - compensate for minute rollover to hrs

```
# continue with time examples
 def add_time( t1, t2):
     sum = Time()
     sum.hour = t1.hour + t2.hour
     sum.minute = t1.minute + t2.minute
     sum.second = t1.second + t2.second

     if sum.second > 60:
             sum.second -= 60
             sum.minute += 1

     if sum.minute > 60:
             sum.minute -= 60
             sum.hour += 1

     return sum

 done = add_time( start, duration)
 print_time( done )
11 : 20 : 00
```

Modifier functions

- Functions can also alter the objects passed as parameters
  - called "modifiers"
- Let's implement a modifier for Time objects
  - it adds a number of seconds to a Time
  - rolls over seconds and minutes
  - Warning: breaks if seconds passed in is > 60

```
# continue our time example…
 def increment( time, seconds ):
    time.second += seconds

    if time.second >= 60:
            time.second -= 60
            time.minute += 1

    if time.minute >= 60:
            time.minute -= 60
            time.hour += 1

 t = Time()
 t.hour = 1
 t.minute = 30
 t.second = 35
 print_time( t )
01 : 30 : 35

 increment( t, 47 )
01 : 32 : 22
```

Summary

- pure function does not modify objects passed as parameters
  - has no "side effects"
  - may return a value
- Modifier functions can also alter the objects passed as parameters
  - has "side effects"
  - may return a value