* Course Overview
  + Callables and lambdas
  + Extended forms of pythons syntax
  + Closures and decorators
  + Advanced use of comprehensions
* Review of Functions
  + Free functions
    - Functions defined at module scope
  + Methods
    - Functions defined within a class definition
  + Positional arguments are matched with formal arguments by position, in order
  + Keyword arguments are matched with formal arguments by name
  + Choice between the two is made at the call site
  + Arguments may have a default value
  + The default value for an argument is only evaluated once
  + Be careful when using mutable data type for default values
    - They may retain changes between calls
  + Functions are objects and can be passed around just like any other object
  + Python 3 system
  + Naming special functions
    - \_\_feature\_\_
    - “dunder” feature
    - Portmanteau of “double underscores”
* Callable Interface
  + \_\_call\_\_()
    - Allows instances of classes to be callable objects
    - \_\_call()\_\_ is invoked on objects when they are called like functions
  + Ex)
    - import socket
    - class Resolver:
    - def \_\_init\_\_(self):
    - self.\_cache = {}
    - def \_\_call\_\_(self, host):
    - if host not in self.\_cache:
    - self.\_cache[host] = socket.gethostbyname(host)
    - return self.\_cache[host]
    - resolve = Resolver()
    - resolve(‘sixty-north.com’) # return 93.93.131.30, syntax sugar for call below
    - resolve.\_\_call\_\_(‘sixty-north.com’) # return 93.93.131.30
  + since callable instances are just normal class instances, their classes can be define any other methods you want