* 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
* Classes Are Callable
  + Class objects and instances of classes are very different things
  + Class binds a class object to a named reference
  + Arguments passed to the class object are forwarded to the class’s \_\_init\_\_()
  + Classes are object factories
  + Classes produce new instances when they are invoked
  + cls
    - shortened version of “class”
    - very common in the python ecosystem
  + klass
    - deliberate misspelling of “class”
    - a bit more explicit
  + conditional expressions
    - evaluates to one of two expression depending on a Boolean
    - result = true\_value if condition else false\_value
* Lambdas
  + in many cases anonymous callable objects will suffice
  + lambda allows you to create such anonymous callable objects
  + use lambda with care to avoid creating inscrutable code
  + use greek letter lambda due to alonzo church’s work on the foundations of computer science
    - his lambda calculus forms the basis for many modern functional languages
  + sorted(iterable, key)
    - iterable like a list of names
    - key: must be a callable like a lambda
  + ex)
    - scientists = [‘Marie Curie’, ‘Albert Einstein’, ‘Rosalind Franklin’ …]
    - sorted(scientists, key=lambda name: name.split()[-1])
  + functions
    - def name(args): body
    - def is a statement which defines a function and binds it to a name
    - must have a name
    - arguments delimited by parentheses, separated by commas
    - zero or more arguments supported - zero arguments == empty parentheses
    - body is an indented block of statements
    - a return statement is required to return anything other than None
    - regular functions can have docstrings
  + lambda
    - lambda args: expr
    - expression which evaluate to a function
    - anonymous
    - argument list terminated by a colon, separated by commas
    - zero or more arguments supported - zero arguments == lambda
    - body is a single expression
    - the return value is given by the body expression, no return statement is permitted
    - lambdas cannot have docstrings
  + detecting callable objects
    - use callable function
  + ex)
    - def is\_even(x):
    - return x % 2 == 0
    - callable(is\_even) #True
    - is\_odd = lambda x: x % 2 == 1
    - callable(is\_odd) #True
* Summary
  + reviewed basics of python functions
  + use \_\_call\_\_() to make callable instances
  + associate state with callable objects
  + classes are callable objects
  + calling a class object creates an instance of the class
  + Lambdas are unnamed callable objects
  + when to use lambdas vs functions and other callables
  + use callable() t o determine if an object is callable
  + conditional expressions are a concise form of conditionals
  + classes are objects
* Extended Formal Argument Syntax