Lecture 38: Python

CS 51G

Spring 2018

Kim Bruce

Announcements

- Test program 2
    - Academic Honesty Guidelines!
- Quiz Friday (Strings & Streams)
- Lecture Friday will be in lab
    - Write searches and sorts in Python

Correction

- Examples from last lecture were run in Python 2 rather than Python 3.
- I've gone back and corrected, but only change was require parentheses around arguments to print.
- Be aware Python 3 now has two division operators 6/4 gives 1.5, while 6 // 4 gives 1

List comprehensions

- Recall lab: filterEvensNSquare
  - collect even elements and square them
- filterEvensNSquare (list[1,2,3,4,5,6]) = [4,16,36]

In Grace

```
// returns a list of numbers that consist of the squares of the
// even number in aList

method filterEvensNSquare (aList: List⟦Number⟧) → List ⟦Number⟧ {
    def answer: List⟦Number⟧ = emptyList⟦Number⟧
    for (aList) do {val: Number →
        if ((val % 2) == 0) then {
            answer.add (val * val)
        }
    }
    answer
}
```

In Python

// returns a list of numbers that consist of the squares of the
// even number in aList

```
def filterEvensNSquare(aList):
    answer = []  # type: List[int]
    for val in aList:
        print val
        if val % 2 == 0:
            answer.append(val*val)
    return (answer)
```

List Comprehensions

```
def squares(aList):
    return [x**2 for x in aList]


def filterEvensNSquare(aList):
    return [x**2 for x in aList if x % 2 == 0]
```

Grace can do similar things with map method on lists

Imports

- Like Grace, can import functions from external files
  - import math
    - must write math.sqrt
  - from math import sqrt #specific function
    - can use it without writing math.sqrt, just sqrt(16).
  - from math import * # import everything
    - again, sqrt, cos, sin, all available without prefix

Example

```
from math import *

x = float( input( "Enter a real value:" ) )
y = sqrt( x )
print (“The square root of", x, "is", y)
print (int(3.7))
```

- input prompts for input, returns response as a string

Exceptions

- try-except rather than try-catch

```
try:
        cost = totalcost / days
except ZeroDivisionError:
        print ("Division by zero error")
```

Object-Oriented Programming in Python

- Python has classes, but no object expressions
- Classes have
  - separate constructors (named __init__)
  - instance variables
  - methods

Class Definitions

```
class myClass(superClass):
    """myClass comment."""
    def __init__(self, otherParams):
        """Initialize object."""
        self.var = exp
        ...

    def someMeth(self, otherParams):
        """comment for method."""
        doStuff ....
```

Class Example

```
from math import sqrt

class Point():
    """Class representing point on screen"""
    def __init__(self, x, y):
        """weird syntax for constructor"""
        self.x = x
        self.y = y

    def translate(self,dx,dy):
        self.x = self.x + dx
        self.y = self.y + dy

    def distance(self,other):
        return sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

    def __str__(self):
        return "<" + str(self.x) + "," + str(self.y) + ">"
```

special methods

Example Using Class

```
p1 = Point(3,4)

print (“p1 = “,p1)  # __str__ automagically called

origin = Point(0,0)

print (“distance =“,p1.distance(origin))

origin.translate(6,8)

print (“new origin = ", origin)
print (“p1 = ", p1)

print (“translated distance =“,p1.distance(origin))
```

Subclass

```
class ColorPoint(Point):
    """Class representing colored point on screen"""

    def __init__(self, x, y, color):
        super().__init__(x,y)
        self.color = color

    def setColor(self,newColor):
        self.color = newColor

    def __str__(self):
        return (super().__str__() + " with color "
                     + self.color)


cp = ColorPoint(2,3,"red")
print (cp)
```

OO in Python

- Faked
- Ugly when writing methods
- OK when calling from libraries
- Be careful: Python 2 syntax for inheritance very different from Python 3

Dictionaries in Python

- Dictionaries are collections that pair a key with a value.
- Example: Phone book pairs name with phone number
- Properties of colors are r, g, and b components
- In Python, pairs designated by ":" to join
  - Keys are unordered.
  - Keys must be immutable!!

More Dictionaries

```
city_population = {"New York City":8550405,
    "Los Angeles":3971883, "Toronto":2731571,
    “Chicago":2720546, "Houston":2296224,
    "Montreal":1704694, "Calgary":1239220,
    "Vancouver":631486, "Boston":667137}

print (city_population["New York City"])  # gives 8550405
print(city_population)       # comes out in different order

city_population["Claremont"] = 35000   # add new city
newDictionary = {}           # create new empty dictionary
```

Why Dictionaries

- Like unordered list where look up items by key rather than index.
- Useful in lots of applications
  - Grace has them as well...

Assignment for class Friday

- Meet in lab
- Learn to use PyCharm
  - Write linear and binary search and time them.

Questions?