TYPES AND LISTS

CSSE 120 – Rose-Hulman Institute of Technology

Outline

- Built-in Help
- Data Types and the **type** function
- Numeric Data Types
- Long Integers vs. Floats
- Type Conversion
- List Operations
- Lab Time

Program Grade Components

Percent	Feature
≥ 70	Correctness: The program accomplishes what the assignment specifies
≤15	Documentation: Comments at beginning of the program. Your name, what the program does How the program is to be run (interactive or reads a file; if the latter, what is its format? Doc comments for classes and functions Internal comments for any parts of the program that may not be obvious to a human reader
≤15	Style/maintainability: Sensible variable and function names No magic numbers Reasonable decomposition into functions, classes, methods Sensible SVN commit messages

Seeing Your Grades in ANGEL

- In the CSSE 120 ANGEL course, choose the REPORTS tab
- Under CATEGORY, choose GRADES
- Click RUN
- You will have to scroll down to see some of your grades

Built-in Help

- dir()
- dir(<identifier>)
- help(<identifier>)
- To see which functions are built-in:
  - `dir(__builtins__)`
  - `help(__builtins__)`
  - `help(abs)`
- Help on imported functions
- `import math`
- `help(math)`
- `help(math.atan2)`

Q1

Data types

- *Data*
  - Information stored and manipulated on a computer
  - Different kinds of data will be stored and manipulated in different ways
- *Data type*
  - A particular way of interpreting bits
  - Determines the possible values an item can have
  - Determines the operations supported on items
  - Python types include: int, float, str, list, function

Numeric data types

```
print "Please enter the count of each kind of coin."
quarters = input("Quarters: ")
dimes = input("Dimes: ")
nickels = input("Nickels: ")
pennies = input("Pennies: ")
total = quarters * 0.25 + dimes * 0.10 +
          nickels * 0.05 + pennies * 0.01
print "The total value of your change is", total
```

Q2

Finding the type of a data item

- Built-in function *type(<expr>)* returns the data type of any value
- Find the types of:
  - 3 3.0 -32 4/5 64.0/5 "Shrubbery" [2, 3]
- Why do we need different numerical types?
  - Operations on int are more efficient and precise
  - Counting requires int
  - floats provide approximate values, used when we need real numbers

Q3

Numeric Types - Summary

- int : integer type
  - Exact values – limited range
  - An operation on two ints **always** yields an int
- float : real number type
  - Approximate values – much larger range
  - An operation on float and int yields a float

```
>>> 5/3
1
>>> 5.0/3
1.6666666666666667
>>> 5/2
2
>>> 5/2.0
2.5
>>> 5%3
2
>>> 5%2
1
>>> 5.0//2.0
2.0
```

Q4

Integer Representations

- An **int** is represented by a fixed-length sequence of bits
  - A **bit** is a **bi**nary digi**t**: its value is either 0 or 1.
- On typical 2009 architectures, that length is 32
- How many different values can be represented by **n** bits?
- Thus there is a largest **int** value
- How to deal with larger integer values?
  - Use floats? What could be wrong with that?
  - Do what other languages do? (overflow)

Q5

Python's long integer type

- Allows arbitrarily large integers
- Automatically created when needed:
```
>>> 10**10
10000000000L
```
- You can specify a long literal
```
>>> 4L/2
2L
>>> type(4L)
<type 'long'>
```
- Since **long** covers all integers (up to the memory limits of the computer) why have an **int** type at all?
  - Why not use **long** for all integer calculations?

Q6

Type Conversions

- Sometimes we have a value of one type, but we need the corresponding value of another type
- In some cases, conversion is automatic:
  - ```
    x = 3
    y = x/7.5
    ```
- Python provides functions that allow you to explicitly convert data to another type
  - int()
  - float()
  - str()

Q3
    ```

Practice with numeric types

- Please download from ANGEL:
  - Lessons > Modules to Download in Class > Session 4 > session04.py
  - Do the **practiceNumberTypes** section.

Sequences in Python

- A sequence is an ordered collection of data items. There are two kinds:
  - List: mutable [3, 4, 6]
  - Tuple: immutable (3, 4, 6)
- Simple examples of generating lists and tuples:
  - `>>> range(4, 11, 2)`
    `[4, 6, 8, 10]`
  - `>>> 3*4, 3-4, 3+4, 3/4`
    `(12, -1, 7, 0)`

Slices of a List

- **`list[m:n]`** returns a new list consisting of `[list[m], list[m+1], list[m+2], … list[n-1]]`
- **`list[:n]`** returns a new list consisting of `[list[0], list[1], … list[n-1]]`
- **`list[m:]`** returns a new list consisting of all elements of `list` beginning with `list[m]`.
- **`list[m:n:k]`**, similar to `range(m, n, k)`, returns a new list consisting of **every k^th element** of `list`, starting with `list[m]`.

Q8

Sequence Operations

- **len(**<sequence>**)**
  - Returns length of the sequence
- <sequence>**.index(**<expr>**)**
  - Returns the index of the first occurrence of the expression in the sequence
- **+** does concatenation
  - `[1, 2] + [7, 5]` is `[1, 2, 7, 5]`
  - `(4,1) + (65, 2)` is `(4, 1, 65, 2)`

List-specific Operations

- <list>**.append (**<expr>**)**
  - Modifies the list by adding the value of the expression to the end of the list
- <list>**.reverse( )**
  - Modifies the list by reversing the order of its elements
- <list>**.sort( )**
  - Modifies the list by sorting the elements into increasing order
- Why don't these operations work with tuples?
- Do **practiceWithLists** from `session04.py` .
  We will do the rest of the exercises next session.

Not all expressions return values

- ```
  >>> numList = [2, 5, 7, 2, 8, 4, 2, 6]
  ```
- ```
  >>> c = numList.count(2)
  >>> c
  3
  ```
- ```
  >>> r = numList.reverse()
  >>> numList
  [6, 2, 4, 8, 2, 7, 5, 2]
  ```
- ```
  >>> r
  ```
- ```
  >>> [r]
  [None]
  ```

Q9
  ```

Optional: A Loop to Make a List

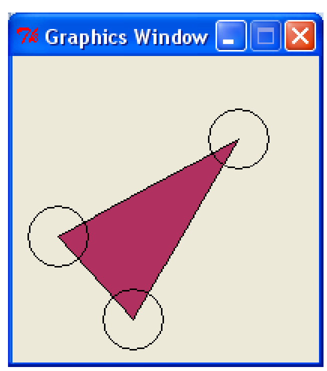
- Python's fancy term for this: **list comprehension**
- `>>> [i*i for i in range(6)]`
  `[0, 1, 4, 9, 16, 25]`
- `>>> [[i, i*i] for i in range(5)]`
  `[[0, 0], [1, 1], [2, 4], [3, 9], [4, 16]]`
- Can you write a list comprehension for the value of the cosine function every 45 degrees around a circle?

A List of Points

```
from zellegraphics import *

win = GraphWin()
pointList = [Point(30, 120), Point(150,55), Point(80, 175)]
poly = Polygon(pointList)
poly.setFill('maroon')
poly.draw(win)

for point in pointList:
    circ = Circle(point, 20)
    circ.draw(win)
```



Homework 4

- See instructions linked from Course Schedule
- Upload solutions to dropboxes on ANGEL
- Once you "get the hang" of problems 3 and 4, you should probably start on *Pizza* and *Polygon* while we're here to help
- It includes a bonus problem 10 pts if you do before Session 5):
  - Make sure that Eclipse, PyDev, and Subclipse are properly installed on your computer (if not, install!)
  - Do some necessary configurations for Eclipse ans Subversion
  - Details in HW4 instructions

Q10, turn in quiz