More on Arithmetic and intro to Objects
Admin

• Any questions
The Project

• Lets talk about the project.
  – What do you need
  – A 'accumulator' variable.
  – Start outside of the loop
    • Lets look at your book's interest example
Data Types

• Review from last time
  – Data represented very differently in computer depending on its type
  – Integer, floating point, character, objects
  – Python hides some of this from you, but the distinction still exists.
    • Time cost of this hiding.
  – Can find the type of anything in python by using the type functions
    • demo
arithmetic

• + - and * all work as expected
  - results are 'closed' on the integers
  - English translation: integer + integer always yields an integer.

• division is not closed on the integers
  - There is a major difference between python 2 and python 3 here. Discuss.
  - but in programming it is
    - print( 8/2) shows?
    - print(7/2) shows?
Integer division in python3

• In python3 can force integer division
  – Using `//`
  – `7//2` is what?
  – Why would I want this?
    • Grid drawing example
    • Searching a list example
  – Modulus/remainder : `%`
    • Explain
    • Rational, integer division & random examples
Rest of arithmetic

- **exponentiation**
  - $4^{**3} = 4*4*4$

- `abs()`
  - Absolute value function
Libraries

- Thus far all of our functions are from the builtin module
- Now we want to use math functions from the math module
  - In order to use stuff from another module need to import it.
  - Let's use math
  - import math
    - Put at top of your file
- Module/library terms in code.
Math library functions

- See 3-2 page 63 in the text for some functions
- Or
  - https://docs.python.org/3.4/library/math.html
- For the whole thing
Using the Math Library

• To access the sqrt library routine, we need to access it as `math.sqrt(x)`.
• Using this dot notation tells Python to use the sqrt function found in the math library module.
• To calculate a root, you can do `discRoot = math.sqrt(b*b – 4*a*c)`
• How would the above be written in mathematical notation?
• If time have students work through factorial example for me
The Limits of Int

• What is 100!?

```python
>>> main()
Please enter a whole number: 100
The factorial of 100 is
933262154439441526816992338856266700490715968264381621468
59296389521759999322991560894146397615651828625369792082
7223758251185210916864000000000000000000000000

• Wow! That’s a pretty big number!
The Limits of Int

- Newer versions of Python can handle it, but...

```python
>>> import fact
>>> fact.main()
Please enter a whole number: 13
13
12
11
10
9
8
7
6
5
4
Traceback (innermost last):
  File "<pyshell#1>", line 1, in ?
    fact.main()
  File "C:\PROGRA~1\PYTHON~1.2\fact.py", line 5, in main
    fact=fact*factor
OverflowError: integer multiplication
```
The Limits of Int

• What’s going on?
  – While there are an infinite number of integers, there is a finite range of ints that can be represented.
  – This range depends on the number of bits a particular CPU uses to represent an integer value. Typical PCs use 32 bits.
The Limits of Int

- Typical PCs use 32 bits.
- That means there are $2^{32}$ possible values, centered at 0.
- This range then is $-2^{31}$ to $2^{31}-1$. We need to subtract one from the top end to account for 0.
- But our 100! is much larger than this. How does it work?
Handling Large Numbers

- Does switching to *float* data types get us around the limitations of *ints*?
- If we initialize the accumulator to 1.0, we get

```python
>>> main()
Please enter a whole number: 15
The factorial of 15 is 1.307674368e+012
```

- *We no longer get an exact answer!*
Handling Large Numbers: Long Int

• Very large and very small numbers are expressed in scientific or exponential notation.
• $1.307674368e+012$ means $1.307674368 \times 10^{12}$
• Here the decimal needs to be moved right 12 decimal places to get the original number, but there are only 9 digits, so 3 digits of precision have been lost.
Handling Large Numbers

• Floats are approximations
• Floats allow us to represent a larger range of values, but with lower precision.
• Python has a solution, expanding ints!
• Python Ints are not a fixed size and expand to handle whatever value it holds.
Type Conversions

• We know that combining an int with an int produces an int, and combining a float with a float produces a float.

• What happens when you mix an int and float in an expression?
  \[ x = 5.0 + 2 \]

• What do you think should happen?
Type Conversions

- For Python to evaluate this expression, it must either convert 5.0 to 5 and do an integer addition, or convert 2 to 2.0 and do a floating point addition.
- Converting a float to an int will lose information.
- Ints can be converted to floats by adding "".0""
Type Conversion

- In *mixed-typed expressions* Python will convert ints to floats.
- Sometimes we want to control the type conversion. This is called *explicit typing.*
Type Conversions

```python
>>> float(22//5)
4.0
>>> int(4.5)
4
>>> int(3.9)
3
>>> round(3.9)
4
>>> round(3)
3
```
Assignment

• Read chapter 3
• New project next time