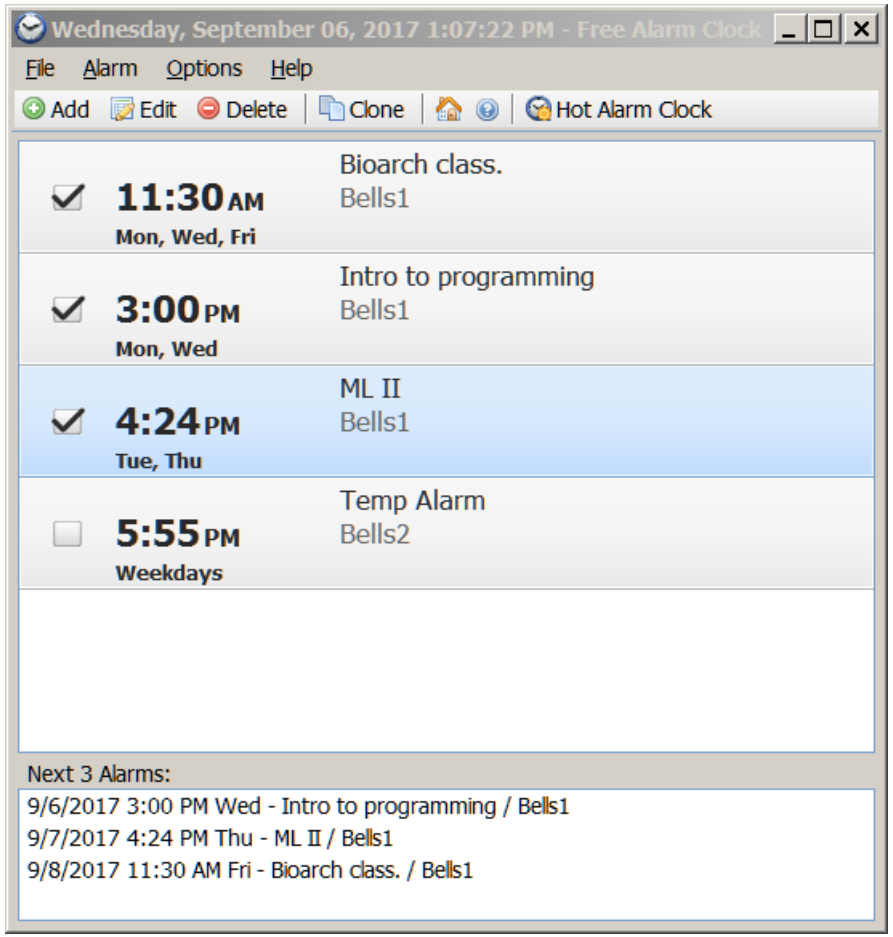
Introduction to Functions

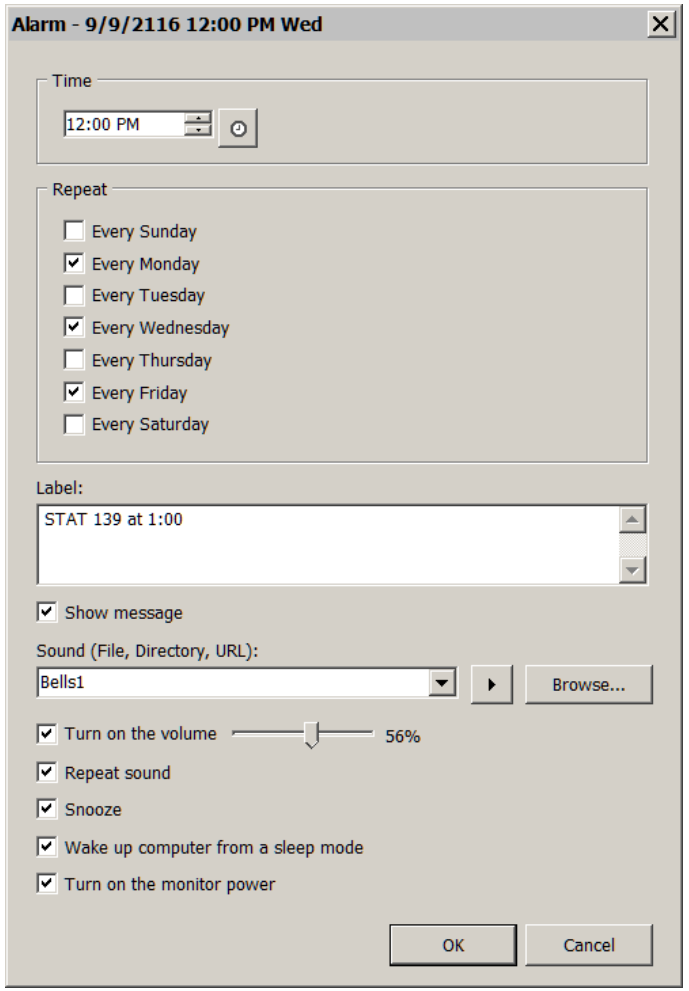
Start thinking about projects
(Start with something you are interested in, like "art"
add that to "Python" in a google search)

Functions are the basis for most applications

A simple but important application for me: Free Alarm Clock

http://freealarmclocksoftware.com/alarmclock.html





Functions are the basis for most programs

Normal format for calling a function:

<function name>**(**<argument(s)>**)**

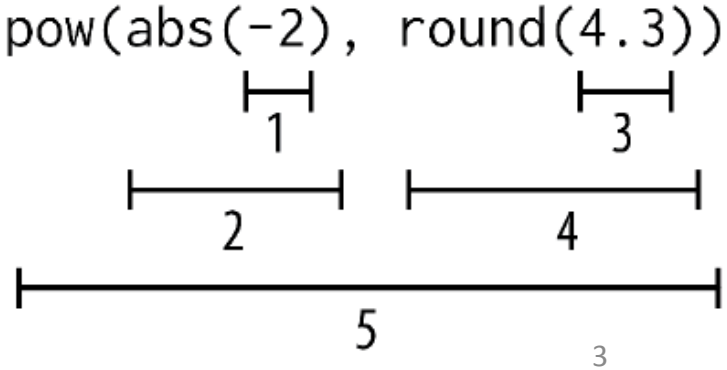
functions generally return something

```
abs(-8) # absolute value
8
pow (3,2) # calculate 3 squared, 3²
9
# round a number - you can input a comment
round(4.343565475476, 2) # round a number
4.34
```

```
# round a number
round(4.343565475476, 2)
4.34
```

nested functions processed from left to right

```
pow(abs(-2), round(4.3)) # 2⁴
16
```



Useful Numeric Functions

```
int(34.65) # return integer, truncated
34
int(-4.3)
-4
float(21) # convert a number (integer or float) into a float
21.0
float(3e+15)  # scientific notation returns number
3000000000000000.0
int(3e+15)
3000000000000000
round (2.5435345) # rounds a number, default is integer
3
round (2.5435345,0) # returns ...?

round (2.5435345,1) # rounds to one decimal place
2.5
```

Useful Numeric Functions

```
# round is a little odd: usually we round up if decimal = .5
round (2.5000) # rounds a number, default is integer
2
round(2.5)  # odd to me, usually round up in measurement
2
round(3.5) # why?
4
round(-2.5) # odd
-2
round(-3.5) # higher magnitude, as in book
-4
round (2.5000000000000000001) # too many digits
2
round (2.5000000000000001) # too many digits (17 total)
2
round (2.500000000000001) # recognized (16 total)
3
round (2438536538534444444444444444444444.500000000000001) # wacky!!!
2438536538534444349510435660103 68
```

Help with Functions

```
# get help
help(pow)
Help on built-in function pow in module builtins:

pow(x, y, z=None, /)
    Equivalent to x**y (with two arguments) or x**y % z (with three arguments)

    Some types, such as ints, are able to use a more efficient algorithm when
    invoked using the three argument form.
pow(2,4) # raise 2 to the fourth power
16
pow(2,4,3) # raise 2 to the fourth power, get modulo (remainder) for 3
1
```

Help with Functions

```
pow(2.2,4.3,3.3) # raise 2.2 to the 4.3 power, get modulo (remainder) for 3.3
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    pow(2.2,4.3,3.3) # raise 2 to the fourth power, get modulo (remainder) for 3
TypeError: pow() 3rd argument not allowed unless all arguments are integers

# so we need two steps (lines):
pow(2.2,4.3) # raise 2.2 to the 4.3 power
29.676831273173452
pow(2.2,4.3) % 3.3
3.2768312731734532
```

The id() Function: Memory addresses

```
help(id)
Help on built-in function id in module builtins:

id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing objects.
    (CPython uses the object's memory address.)
id(-9)
48976752
id(23.4)
48976192
dc = 23.4
id(dc)
48976512
dc = 534543
id(dc)
48976736
id(abs)
1144872
```

The id() Function

Python stores common values 1..255 in the same memory address (cached)

```
i = 3
j = 3
k = 4 - 1
id(i)
4296861792
id(j)
4296861792
id(k)
4296861792
```

each refers to the same object (aliasing)

IDLE and running Python programs

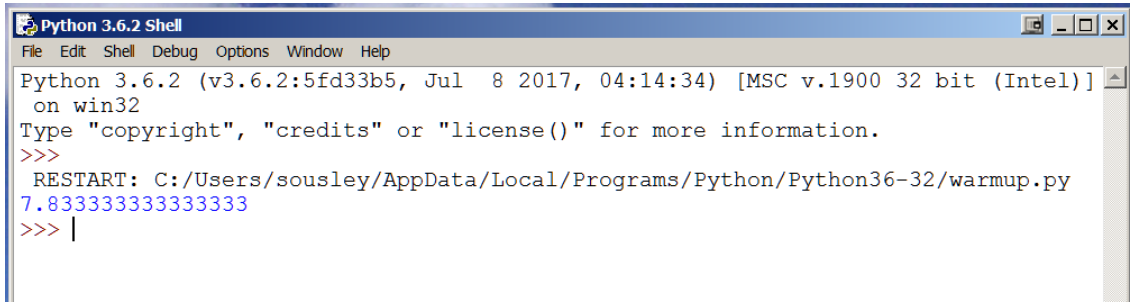
We can only enter one line at a time into IDLE.

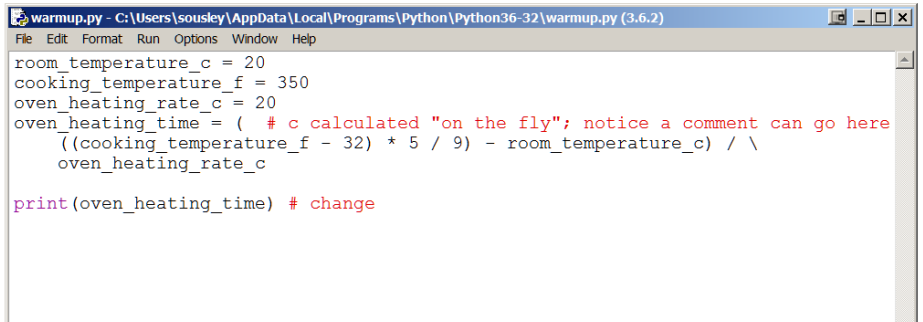
We will start running multiple-line functions and programs, so to save time:

Start IDLE, choose File | New; paste in code, note indents (four spaces)

```
room_temperature_c = 20
cooking_temperature_f = 350
oven_heating_rate_c = 20 # temp rise per minute
oven_heating_time = (  # calculated "on the fly"; notice a comment can go here;
    ((cooking_temperature_f - 32) * 5 / 9) - room_temperature_c) / \
    oven_heating_rate_c

print(oven_heating_time)
```





IDLE and running Python programs

Press F5 and you can run your code.

You will be prompted to save your file.

Give it a name. It will run.

What happened?

Python 3.6.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/sousley/AppData/Local/Programs/Python/Python36-32/warmup.py
7.833333333333333
>>> |
```

warmup.py - C:\Users\sousley\AppData\Local\Programs\Python\Python36-32\warmup.py (3.6.2)

File Edit Format Run Options Window Help

```
room_temperature_c = 20
cooking_temperature_f = 350
oven_heating_rate_c = 20
oven_heating_time = (  # c calculated "on the fly"; notice a comment can go here
    ((cooking_temperature_f - 32) * 5 / 9) - room_temperature_c) / \
    oven_heating_rate_c

print(oven_heating_time) # change
```

Special keys and configuring the editor

What is the most important special key in the history of computing?
`UNDO! (Ctrl-Z)`

What is the SECOND most important special key in the history of computing?
`REDO! (Shift-Ctrl-Z or Ctrl-Y (MS) )`

When you change the file, an asterisk reminds you
Python must save the file before running it (F5)
You can change an option to autosave the file before running
- no worries, you can still undo

Configuring the editor

Set Autosave

Choose Options | Configure IDLE

Lots of options

Choose the general tab, set Autosave preferences to "No prompt"

- file will be saved every time and then run

Defining functions

```
# We want to convert to Celsius
# anatomy of a function: def, then indented code, then what to return
def convert_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5 / 9
#  ^^ notice the four-space indent!!!!
```

fahrenheit in this example is a **parameter** (inside the function)

There are many many metric/English conversion functions

Press F5 - what happens this time?

Defining functions: local variables

```
# Calculate quadratic formula, ax²+ bx + c given a,b,c,x
# return can involve a calculation
def quadratic(a, b, c, x):
    first = a * x ** 2
    second = b * x
    third = c
    return first + second + third
```

a,b,c, and x are parameters: only valid within the function

(**arguments** are provided **to** the function)

first, second, and third are local variables: only valid within the function

Defining functions

We can't use keywords for ANY variables:

```
False assert del for in or while
None break elif from is pass with
True class else global lambda raise yield
and continue except if nonlocal return
as def finally import not try
```

```
try:
pass = 5

pass6 = 6
```

Defining functions more formally (the function design recipe)

```
def convert_to_celsius(fahrenheit):
      """ (number) -> float

    Return the number of Celsius degrees equivalent to fahrenheit degrees.

    >>> convert_to_celsius(75)
    23.88888888888889
    """
    return (fahrenheit - 32.0) * 5.0 / 9.0
```

fahrenheit in this example is a parameter

Defining functions more formally (the function design recipe)

function header

```
def convert_to_celsius(fahrenheit):
    """ (number) -> float
```

number = integer or float

docstring starts with three double quotes;
provides parameter and return format (type contract)

description

```
    Return the number of Celsius degrees equivalent to fahrenheit degrees.
```

example(s)

```
    >>> convert_to_celsius(75)
    23.88888888888889
    """
```

body (the code that does something!)

```
    return (fahrenheit - 32.0) * 5.0 / 9.0
```

fahrenheit in this example is a parameter

Homework 2 due before class next Wednesday

Exercises (3.11): 1, 2, 4, 5, 6, 7

Follow the function design recipe (section 3.6)! (PEP 8 may help too)

(PEP: variable names to avoid: `l`, `O`, `I` / `l`, `O`, `I`)

And:

Write a function called SuperFloatPow that will perform the same as the pow() function but will accept three floats as arguments

Homework Guidelines (again)

IMPORTANT:

You will need to put your answers into a Word document with your name at the top with "Homework 2".

Copy and paste Python code and output with each answer.

Name your file <*LastName*>-Homework2.docx. Example: Smith-Homework2.docx

Attach to an email to me with the subject "DATA 520 Homework2"

to sousley@mercyhurst.edu

- I will show you next time why consistency is important