Functions:
The 8 Step Plan

Professor Doug Blank
cs.brynmawr.edu/~dblank
dblank@cs.brynmawr.edu

Functions

- Used to sequence commands
- Used to do a well-defined computation

Functions

- Used to sequence commands
- Used to do a well-defined operation

- You don't ***need*** functions---but they can be a lot of help!

Writing a Function

52	c	523.251	
51	b	493.883	
50	a♯	466.164	
49	a	440.000	
48	g♯	415.305	
47	g	391.995	
46	f♯	369.994	
45	f	349.228	
44	e	329.628	
43	d♯	311.127	
42	d	293.665	
41	c♯	277.183	
40	c	261.626	(middle C)

Writing a Function

```
c       = 523.251
b       = 493.883
aSharp  = 466.164
a       = 440.000
gSharp  = 415.305
g       = 391.995
fSharp  = 369.994
f       = 349.228
e       = 329.628
dSharp  = 311.127
d       = 293.665
cSharp  = 277.183
c       = 261.626    # middle C
```

Writing a Function

```
c2       = 523.251
b        = 493.883
aSharp = 466.164
a        = 440.000
gSharp = 415.305
g        = 391.995
fSharp  = 369.994
f        = 349.228
e        = 329.628
dSharp  = 311.127
d        = 293.665
cSharp = 277.183
c1       = 261.626    # middle C
```

Writing a Function

```
...
c2          = 261.626    # middle C

beep(.5, c2)
beep(.5, a)
beep(.5, fSharp)
beep(.5, aSharp)
```

Writing a Function

```
...
c2          = 261.626    # middle C
```

beep(.5, c2)
beep(.5, a)
beep(.5, fSharp)
beep(.5, aSharp)

1. Indent commands

Writing a Function

```
...
c2          = 261.626    # middle C

def name():
    beep(.5, c2)
    beep(.5, a)
    beep(.5, fSharp)
    beep(.5, aSharp)
```

1. Indent commands
2. add a **def name():**

Writing a Function

```
...
c2          = 261.626    # middle C

def refrain():
    beep(.5, c2)
    beep(.5, a)
    beep(.5, fSharp)
    beep(.5, aSharp)
```

1. Indent commands
2. add a def name():

Writing a Function

```
...
c2          = 261.626    # middle C

def refrain():
    timing = .5
    beep(timing, c2)
    beep(timing, a)
    beep(timing, fSharp)
    beep(timing, aSharp)
```

1. Indent commands
2. add a def name():
3. abstract common parts

Writing a Function

```
...
c2          = 261.626    # middle C

def refrain(timing):
    beep(timing, c2)
    beep(timing, a)
    beep(timing, fSharp)
    beep(timing, aSharp)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables

Writing a Function

```
...
c2          = 261.626    # middle C

def refrain(timing):
    beep(timing, c2)
    beep(timing, a)
    beep(timing, fSharp)
    beep(timing, aSharp)

refrain(.5)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. “call” the function

Writing a Function

...

```
beep(.5, c2)
beep(.5, a)
beep(.5, fSharp)
beep(.5, aSharp)
beep(.25, c2)
beep(.25, a)
beep(.25, fSharp)
beep(.25, aSharp)
```

Writing a Function

...

```
beep(.5, c2)
beep(.5, a)
beep(.5, fSharp)
beep(.5, aSharp)
```

```
beep(.25, c2)
beep(.25, a)
beep(.25, fSharp)
beep(.25, aSharp)
```

Writing a Function

...

refrain(.5)

refrain(.25)

Functions

- Used to sequence commands
- Used to do a well-defined computation

Python Functions

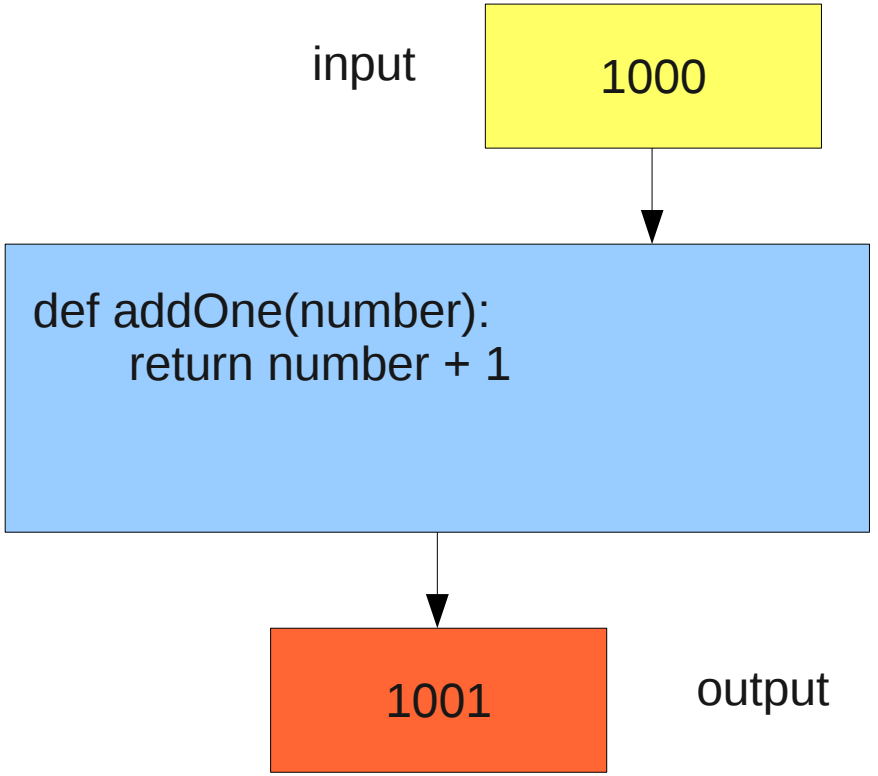
```
def addOne(number):
    return number + 1

def double(num):
    return num * 2
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. **add a return**
6. “call” the function

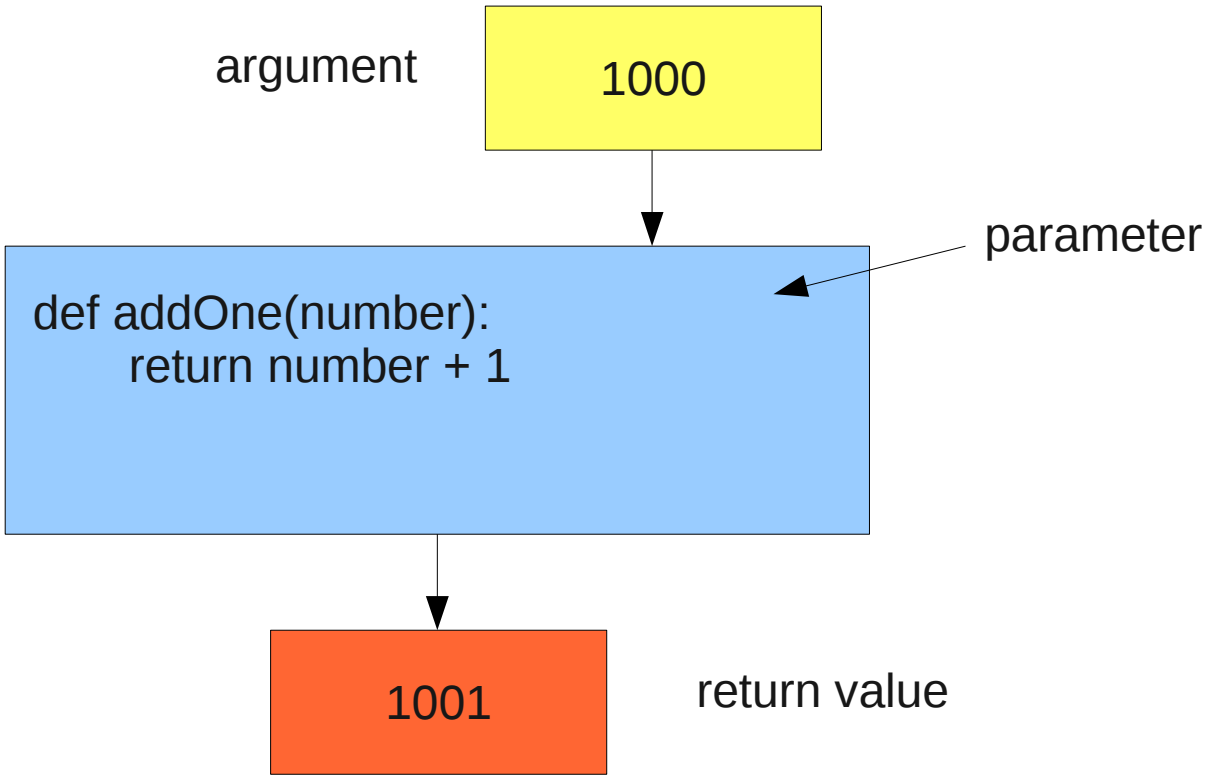
Python Functions

```
def addOne(number):
    return number + 1

def double(num):
    return num * 2

>>> double(5)
10
>>> addOne(1000)
1001
```

Python Functions



Python Functions



Python Functions

```
>>> double(5)
10
>>> addOne(1000)
1001

>>> double( addOne(1000) )
```

Python Functions

```
>>> double(5)
10
>>> addOne(1000)
1001

>>> double( addOne(1000) )
2002
```

Composing functions

Python Functions

```
>>> double(5)
10
>>> addOne(1000)
1001

>>> double( addOne(1000) )
2002

>>> addOne( double(1000) )
```

Composing functions

Python Functions

```
>>> double(5)
10
>>> addOne(1000)
1001

>>> double( addOne(1000) )
2002
```

Composing functions

```
>>> addOne( double(1000) )
2001
```

Python Functions

```
>>> double( addOne(1000) )
2002

>>> addOne( double(1000) )
2001

>>> double( double(1000) )
```

Composing functions

Python Functions

```
>>> double( addOne(1000) )
2002

>>> addOne( double(1000) )
2001

>>> double( double(1000) )
4000
```

Composing functions

Python Functions

What is the temperature, in Celsius?

To convert from Fahrenheit to Celsius

- Begin by subtracting 32 from the Fahrenheit number.
- Divide the answer by 9.
- Then multiply that answer by 5.

The temperature is 72 degrees Fahrenheit

To convert from Fahrenheit to Celsius

- Begin by subtracting 32 from the Fahrenheit number.
- Divide the answer by 9.
- Then multiply that answer by 5.

The temperature is 72 degrees Fahrenheit

F = 72
temp1 = F – 32
temp2 = temp1 / 9
temp3 = temp2 * 5

To convert from Fahrenheit to Celsius

F = 72
temp1 = F – 32
temp2 = temp1 / 9
temp3 = temp2 * 5

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

F = 72
temp1 = F – 32
temp2 = temp1 / 9
temp3 = temp2 * 5

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius():
    F = 72
    temp1 = F – 32
    temp2 = temp1 / 9
    temp3 = temp2 * 5
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    temp1 = F – 32
    temp2 = temp1 / 9
    temp3 = temp2 * 5
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    temp1 = F – 32
    temp2 = temp1 / 9
    temp3 = temp2 * 5
    return temp3
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    temp1 = F – 32
    temp2 = temp1 / 9
    temp3 = temp2 * 5
    return temp3


celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):

    temp2 = (F – 32) / 9
    temp3 = temp2 * 5
    return temp3


celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):

    temp3 = ((F – 32) / 9) * 5
    return temp3

celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):

    return ((F – 32) / 9) * 5

celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    return ((F – 32) / 9) * 5
```

```
celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    return ((F – 32) / 9) * 5
```

```
celsius(72)
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. add **useful** comments
7. “call” the function

To convert from Fahrenheit to Celsius

```
def celsius(F):
    ''' Converts Fahrenheit to Celsius '''
    return ((F – 32) / 9) * 5
```

```
>>> celsius(72)
22.2
```

1. Indent commands
2. add a def name():
3. abstract common parts
4. add variables
5. add a return
6. add **useful** comments
7. “call” the function