Disclaimer

- This is not a class!
- It’s just a CyberEagles python seminar
- I am not a teacher… yet

Functions and List Comprehensions

Vitaly Ford

11/18/15

Function

- A module of code
- Accomplishes a specific task
- Takes input, processes, returns a result
- Good to reuse code
- Easier to comprehend what’s going on
- Good for testing small parts of code

Defining a Function

def *name (parameters)*:

        here_comes_the_code

        return *expression*

Calling a Function

```
def division ( a, b ):
        c = a / b
        return c

result = division ( 64, 16 )
print ( result )
```

What’s the problem with this function?

__main__

```
if __name__ == “__main__”:
    print ( “This program is being run by itself” )
else:
    print ( “I am being imported from another module” )
```

Parameters

- Passed by reference

```
def add_elements ( inputList, headElement, tailElement ):
        inputList.append ( tailElement )
        inputList.insert ( 0, headElement )
        return

if __name__ == “__main__”:
        mylist = [5, 6, 7]
        add_elements (mylist, 4, 8)
```

Parameters (cont.)

- Required
- Keyword
- Default
- Variable-length

Required Parameters

```
def add_elements ( inputList, headElement, tailElement ):
        inputList.append ( tailElement )
        inputList.insert ( 0, headElement )
        return

if __name__ == “__main__”:
        add_elements ()
```

$\leftarrow$ Throws an error!

Keyword Parameters

```
def calc_stuff ( len, width, height, radius, sigma ):
        # do stuff with input params
        result = len – width + height – radius / sigma
        print ( result )
        return


calc_stuff ( 5.6, 7, 8, 33, 0.3 )


calc_stuff ( height = 8, sigma = 0.3, len = 5.6, width = 7, radius = 33 )
```

Default Parameters

```
def calc_weights ( a, b, c, precision = 0.1 ):
        # do stuff
        print ( “Precision: ”, precision )
        return

calc_weights ( 1, 2, 3 )

calc_weights ( 1, 2, 3, 0.5 )
```

Variable-length

```
def sum_them_up ( *allVars ):
        sum = 0
        for v in allVars:
                sum += v
        return sum

s = sum_them_up ( 5, 4 )
s = sum_them_up ( 3, -6, 7, 0, 10 )
```

Global vs Local Variables

```
result = 100

def mult ( a, b ):
        result = a * b
        print ( “Inside result: “, result )

mult ( 8, 6 )
print ( “Outside result: “, result )
```

Want access a global variable inside your function? Do

```
        global result
```

List Comprehensions

```
V = [ 2**i   for  i  in range ( 10 ) ]


E = [ i  for  i  in range (20) if i % 2 == 0]

L = “I love Star Wars”.split()


K = [ [ i, len (i) ]  for  i  in  L ]
```

Lab – Easy

- Ask a user to enter usernames; insert them all into a list. Whenever the user enters a keyword “stop” (capitalized or not – should not matter), print everything that is in the list, line by line.

Lab – Medium

- Find all palindromic primes up to 1000

Lab – Hard

- Write a Fibonacci sequence with the help of list comprehensions