

CIS 122

Fully Functioning

Homework 0 Revisited

- Almost everyone submitted
 - If you didn't submit, let me know
 - I don't get notified when someone drops this class
 - If you don't submit homework, I'll assume the worst
- Very good for the most part
- A few bugs...
 - Better now than later
 - Get them out of our systems

Homework 0 Revisited

- Submit your code, not the shell
 - Python lets you save both (gah!)
 - I can't run your shell session
 - Make sure you can run whatever you submit
- Some code looked right, but couldn't run
 - Minor typos
 - Always run your code right before submitting

Homework 0 Revisited

- Break up comments
 - IDLE lets you type very long strings
 - But your screen is only so long...
This comment is so long
I broke it into two lines
- Write your name at the top of your files
 - I provide the header, just fill it in
CIS 122 Assignment 1
Due July 3, 2011
Name:
Partner: (if applicable)

String Things

- Python numbers its characters
- ord method converts characters to numbers

```
>>> ord('a')
97
```
- chr method converts numbers to characters

```
>>> chr(97)
'a'
```
- Python uses these numbers for comparing strings

String Things

- String Indexing

- `s[i]`
- Return the character in string `s` at position `i`
- Start counting from 0

```
>>> "STRING"[2]
'R'
```

- You can index with negative numbers too

- `s[-i]`
- Return the `i`th character from the right
- Start counting from -1

```
>>> "STRING"[-2]
'N'
```

String Things

- String slicing

- `s[i:j]`
- Return a subset of characters in `s`
- Starting at character `i`,
- Up to (but not including) character `j`

```
>>> "STRING"[1:4]
'TRI'
```

- If you leave off an index, defaults to beginning / end

- `s[i :]` - all characters from character `i` onward
- `s[: i]` - all characters up to (but not including) character `i`
- `s[:]` - all characters

String Things

- String slicing with skips

- `s[i:j:k]`
- Start at character `i`
- Count up by `k`...
- Stop before character `j`

```
>>> "ABCDEFGH"[1:6:2]  
'BDF'
```

- You can skip backwards too!

```
>>> "ABCDEFGH"[6:1:-2]  
'GEC'
```

String Quiz

```
s1 = "STRINGS"
```

```
s2 = "SLICE"
```

```
s3 = "SPLIT"
```

```
s4 = s1 + s2[ :: -1] + s3[ :: -1]
```

```
s5 = s4[2 :: 5]
```

```
s6 = s2[3 : ]
```

```
s7 = s6 + s1[-1]
```

```
message = s7[ :: -1] + s5
```

What's the secret message?

String Quiz

```
s1 = "STRINGS"
```

```
s2 = "SLICE"
```

```
s3 = "SPLIT"
```

```
s4 = s1 + s2[ :: -1] + s3[ :: -1]
```

```
s5 = s4[2 :: 5]
```

```
s6 = s2[3 : ]
```

```
s7 = s6 + s1[-1]
```

```
message = s7[ :: -1] + s5
```

```
s1 → 'STRINGS'
```

```
s2 → 'SLICE'
```

```
s3 → 'SPLIT'
```

What's the secret message?

String Quiz

s1 = "STRINGS"

s2 = "SLICE"

s3 = "SPLIT"

s4 = s1 + s2[:: -1] + s3[:: -1]

s5 = s4[2 :: 5]

s6 = s2[3 :]

s7 = s6 + s1[-1]

message = s7[:: -1] + s5

s1 → 'STRINGS'

s2 → 'SLICE'

s3 → 'SPLIT'

s4 → 'STRINGSECILSTILPS'

What's the secret message?

String Quiz

s1 = "STRINGS"

s2 = "SLICE"

s3 = "SPLIT"

s4 = s1 + s2[:: -1] + s3[:: -1]

s5 = s4[2 :: 5]

s6 = s2[3 :]

s7 = s6 + s1[-1]

message = s7[:: -1] + s5

s1 → 'STRINGS'

s2 → 'SLICE'

s3 → 'SPLIT'

s4 → 'STRINGSECILSTILPS'

s5 → 'RET'

What's the secret message?

String Quiz

s1 = "STRINGS"

s2 = "SLICE"

s3 = "SPLIT"

s4 = s1 + s2[:: -1] + s3[:: -1]

s5 = s4[2 :: 5]

s6 = s2[3 :]

s7 = s6 + s1[-1]

message = s7[:: -1] + s5

s1 → 'STRINGS'

s2 → 'SLICE'

s3 → 'SPLIT'

s4 → 'STRINGSECILSTILPS'

s5 → 'RET'

s6 → 'CE'

What's the secret message?

String Quiz

s1 = "STRINGS"

s2 = "SLICE"

s3 = "SPLIT"

s4 = s1 + s2[:: -1] + s3[:: -1]

s5 = s4[2 :: 5]

s6 = s2[3 :]

s7 = s6 + s1[-1]

message = s7[:: -1] + s5

s1 → 'STRINGS'

s2 → 'SLICE'

s3 → 'SPLIT'

s4 → 'STRINGSECILSTILPS'

s5 → 'RET'

s6 → 'CE'

s7 → 'CES'

What's the secret message?

String Quiz

s1 = "STRINGS"

s2 = "SLICE"

s3 = "SPLIT"

s4 = s1 + s2[:: -1] + s3[:: -1]

s5 = s4[2 :: 5]

s6 = s2[3 :]

s7 = s6 + s1[-1]

message = s7[:: -1] + s5

s1 → 'STRINGS'

s2 → 'SLICE'

s3 → 'SPLIT'

s4 → 'STRINGSECILSTILPS'

s5 → 'RET'

s6 → 'CE'

s7 → 'CES'

message → 'SECRET'

What's the secret message?

Writing functions

- Python has many built-in functions
 - But what if it doesn't have the one you're looking for?
- Write your own!

Anatomy of a Function

```
def plusOne(myNum):  
    """Adds one to myNum"""  
  
    myLargerNum = myNum + 1  
    return myLargerNum
```

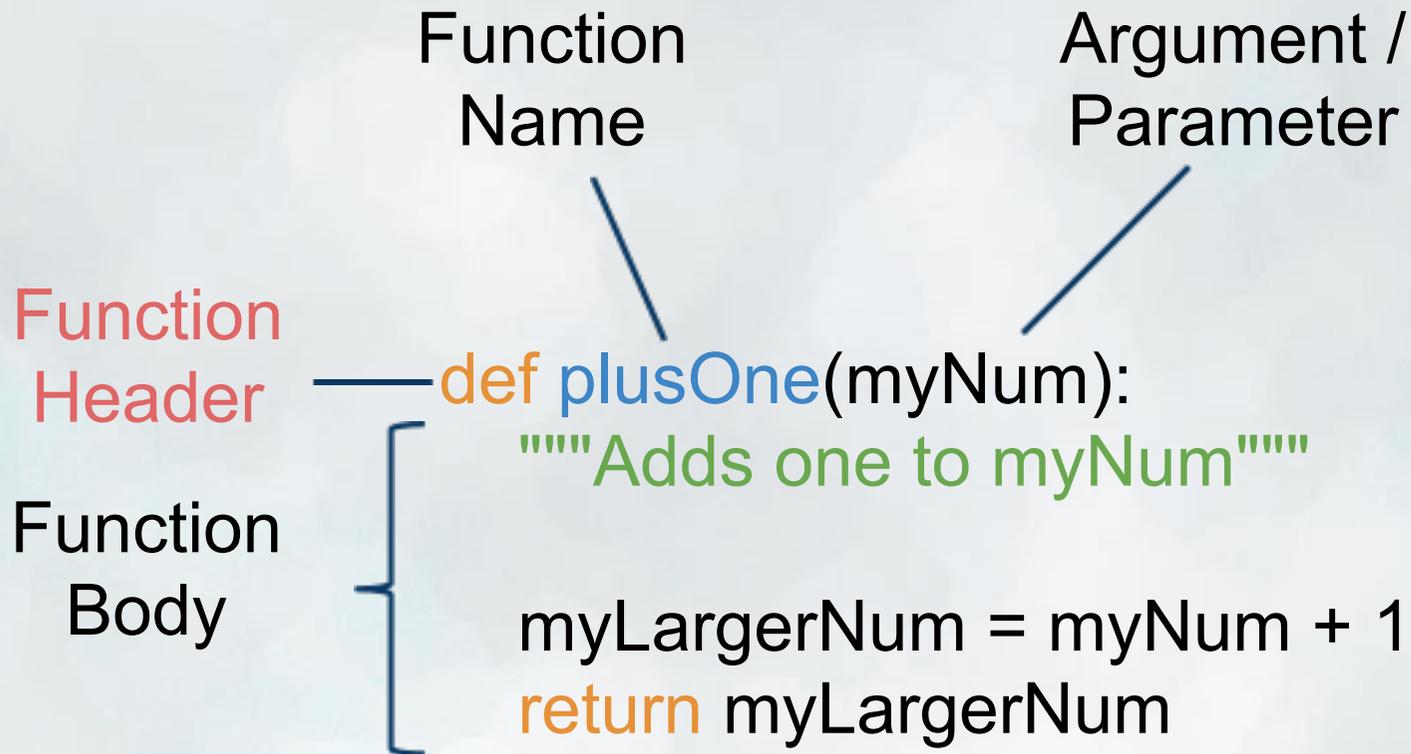
Anatomy of a Function

Function
Header

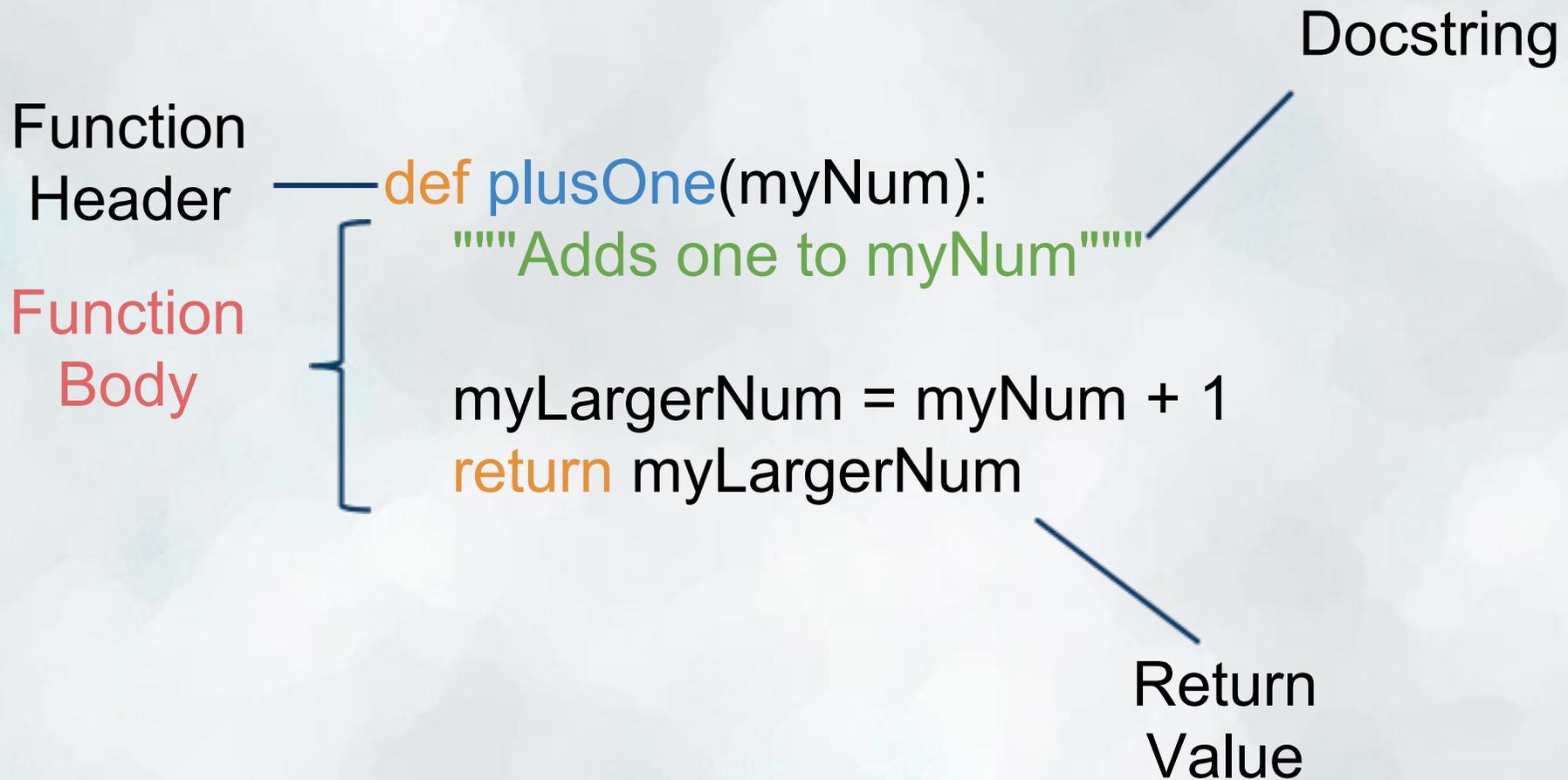
Function
Body

```
def plusOne(myNum):  
    """Adds one to myNum"""  
  
    myLargerNum = myNum + 1  
    return myLargerNum
```

Anatomy of a Function



Anatomy of a Function



Breaking it Down

- Function header
 - def
 - name
 - arguments (formal parameters)
 - colon

```
def plusOne(myNum):  
    """Adds one to myNum"""  
  
    myLargerNum = myNum + 1  
    return myLargerNum
```

Breaking it Down

- Function body
 - Indented
 - Docstring
 - Sequence of commands
 - Return value

```
def plusOne(myNum):  
    """Adds one to myNum"""  
  
    myLargerNum = myNum + 1  
    return myLargerNum
```

Breaking it Down

- So what happens when someone calls my function?
 - Assign actual parameter to formal parameter
 - Run through function code
 - Stop at return value

```
def plusOne(myNum):  
    """Adds one to myNum"""  
  
    myLargerNum = myNum + 1  
    return myLargerNum
```

```
>>> plusOne(3)
```

```
4
```

To be continued...

- Let's try out what we've learned in IDLE

Why Write Functions?

- Why write functions
 - (could just copy function code...)
- Multiple argument functions
- Stack Diagrams
- Conditional logic