Strings
Lists
Functions

Introduction to Computer Science!

https://ucsb-cs8-f18.github.io/

Strings

- A string is a sequence of characters
- Anything within single or double quotes: E.g “UCSB”, ‘73$505abc’
- Special characters may be included by preceding them with ‘\’: E.g. “UC\”SB”

String operations

- Concatenation: +
- Repetition: *
- Extract certain pieces (also called parsing)
  - Indexing: x[0], x[-1]
  - Extract substring: x[0:3]
- We can check if some character is in a string using the ‘in’ or ‘not in’ keywords

What is the value of s after the following code runs?

```
s =  'abc'
s =  'd' * 3 + s
s  = s +  e* 2
```

A. 'abcd3e2'
B. 'abcdddabc'
C. 'dddabcee'
D. 'abcdddabce2'
E. Error

Lists

- Lists - A list is a collection of multiple values (similar to how a str is a collection of characters).
- Note: In python, lists can be of heterogenous (different) types
- Lists can have duplicate values
- The elements of a list can be modified (lists are mutable)

Practice strings

Write code that produces the following output for. the input “Diba”

Run 1:

What is your name? Diba

Hi Dibaaaaaa !!!!

I meant hi Diiiiiba

Sorry I have a cold, Biba

Run 2:

What is your name? Eric

Hi Ericccccc !!!!

I meant hi Errrrric

Sorry I have a cold, Iric

*Function*ing in Python

```
# my own function!
def dbl( x ):
    """ returns double its input, x """
    return 2x
```

This doesn't look quite right…

Functioning in Python

```
# my own function!
def dbl( x ):
    """ returns double its input, x """
    return 2*x
```

Some of Python's *baggage*…

Docstrings

They become part of python's built-in help system!
With each function be sure to include one that

(1) describes overall what the function does, and
(2) explains what the inputs mean/are

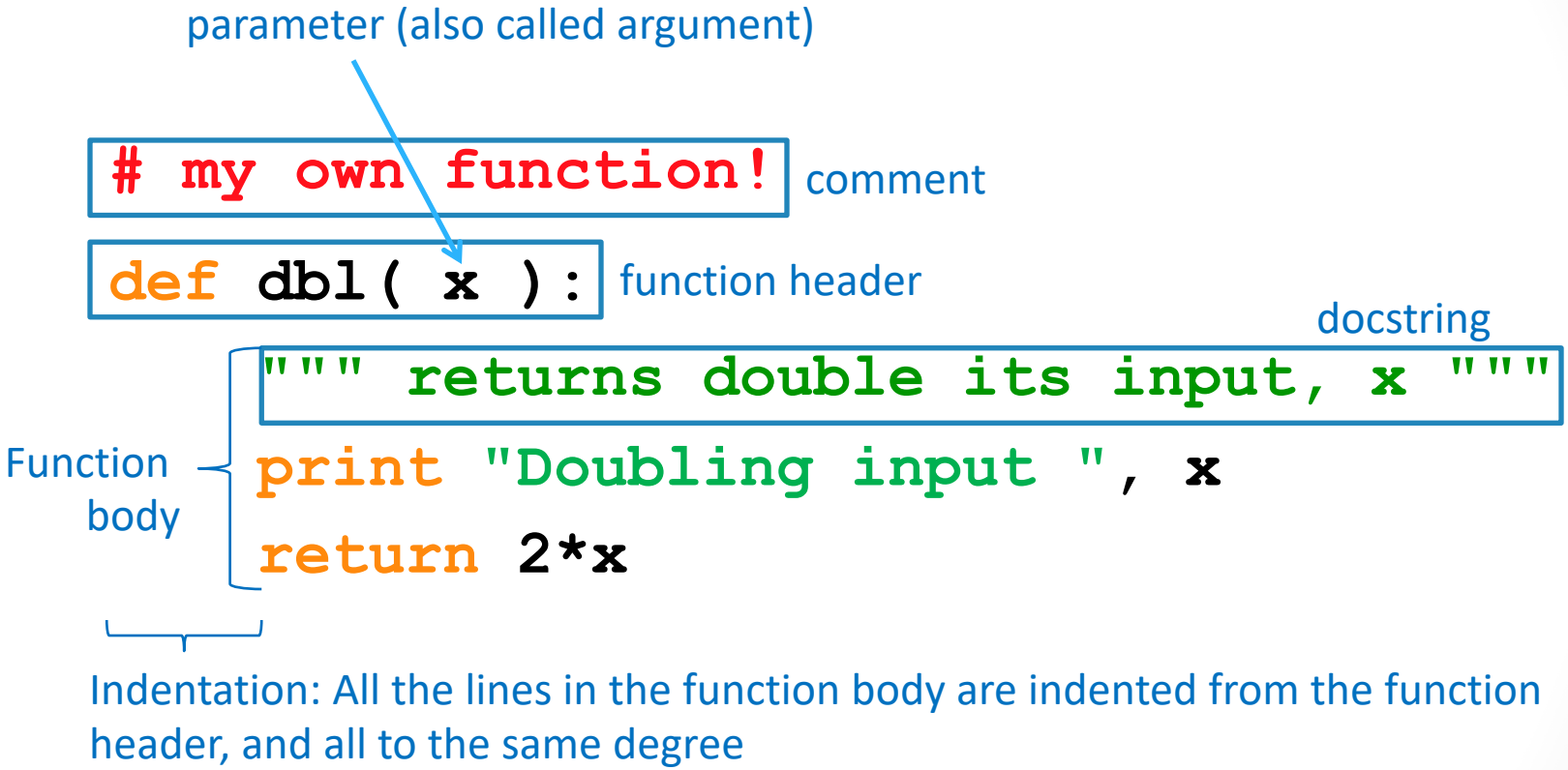
keywords

`def` starts the function
`return` stops it immediately and sends back the return value

Comments

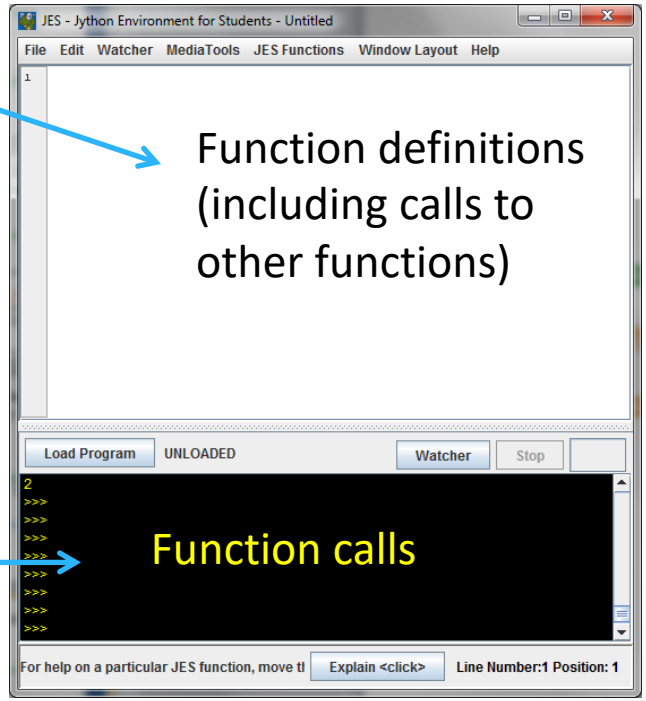
They begin with #

Essential Definitions and Rules *(do memorize)*



Flow of Execution

```
# my own function!
def dbl( x ):
    """ returns double its input, x """
    print "Doubling input ", x
    return 2*x
```




```
>>> dbl( 21 )
```

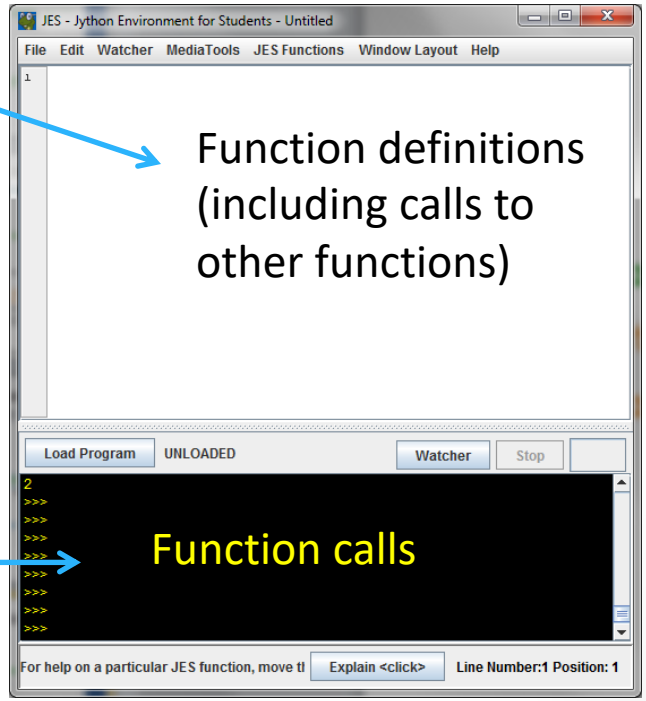
When you call a function, Python executes the function starting at the first line in its body, and carries out each line in order (though some instructions cause the order to change… more soon)

Parameters are special variables

```
# my own function!
def dbl( x ):
    """ returns double its input, x """
    print "Doubling input ", x
    return 2*x
```

x

```
>>> dbl( 21 )
```



When you call a function, the value you put in parenthesis gets put into the "box" labeled with the name of the parameter and is available for use within the function.

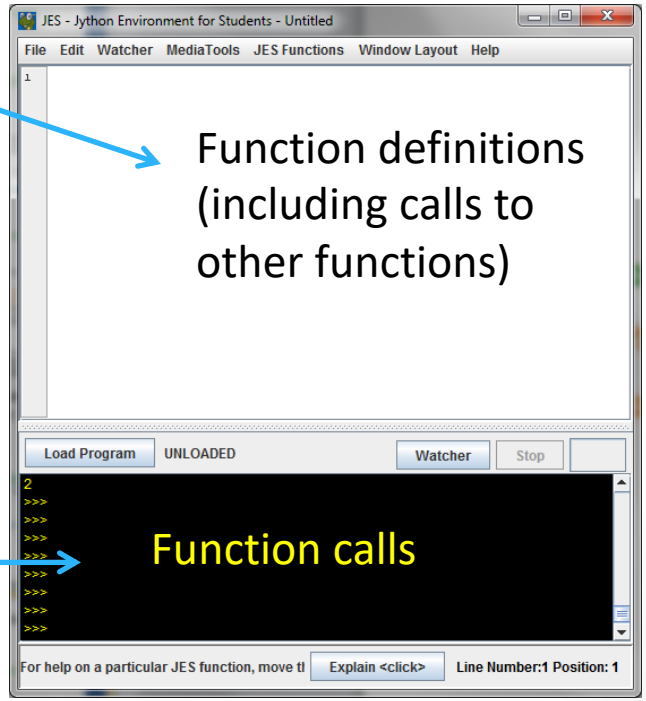
Multiple parameters are allowed

```
# my own function!
def times( x, y ):
    """ returns x times y """
    print "Multiplying ", x, "and", y
    return x*y
```

x

y

```
>>> times( 21, 2 )
```



When you call a function, the values you put in parenthesis gets put into the “boxes” labeled with the names of the parameters (in the order in which they are listed)

No parameters is also allowed

```
# my own function!
def fortyTwo( ):
    """ returns 42 """
    return 42
```

```
>>> fortyTwo
```

As much as I like 42, I don't quite like this…

(But you still need parentheses)

```
# my own function!
def fortyTwo( ):
    """ returns 42 """
    return 42
```

```
>>> fortyTwo()
```

Ahh(), much better

Functions can call Functions!!

When in doubt, draw it out!

```
def halve( x ):
    """ returns half its input, x """
    return div(x, 2)
def div( y, x ):
    """ returns y / x """
    return y / x
```

```
>>> halve( 84 )
```

Functions can call Functions!!

```
def halve( x ):
    """ returns half its input, x """
    return div(x, 2)
def div( y, x ):
    """ returns y / x """
    return y / x
```

```
>>> halve( 85 )
```

What does halve(85) return?

A. 42
B. 42.5
C. 0
D. 0.02352 (i.e., 2 divided by 85)