Clicker Frequency AB

Python Functions

Introduction to Computer Science!

*Function*ing in Python

keyword

name (this can be any name of your choice)

input parameter (s)

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

keyword

like a comment → docstring

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

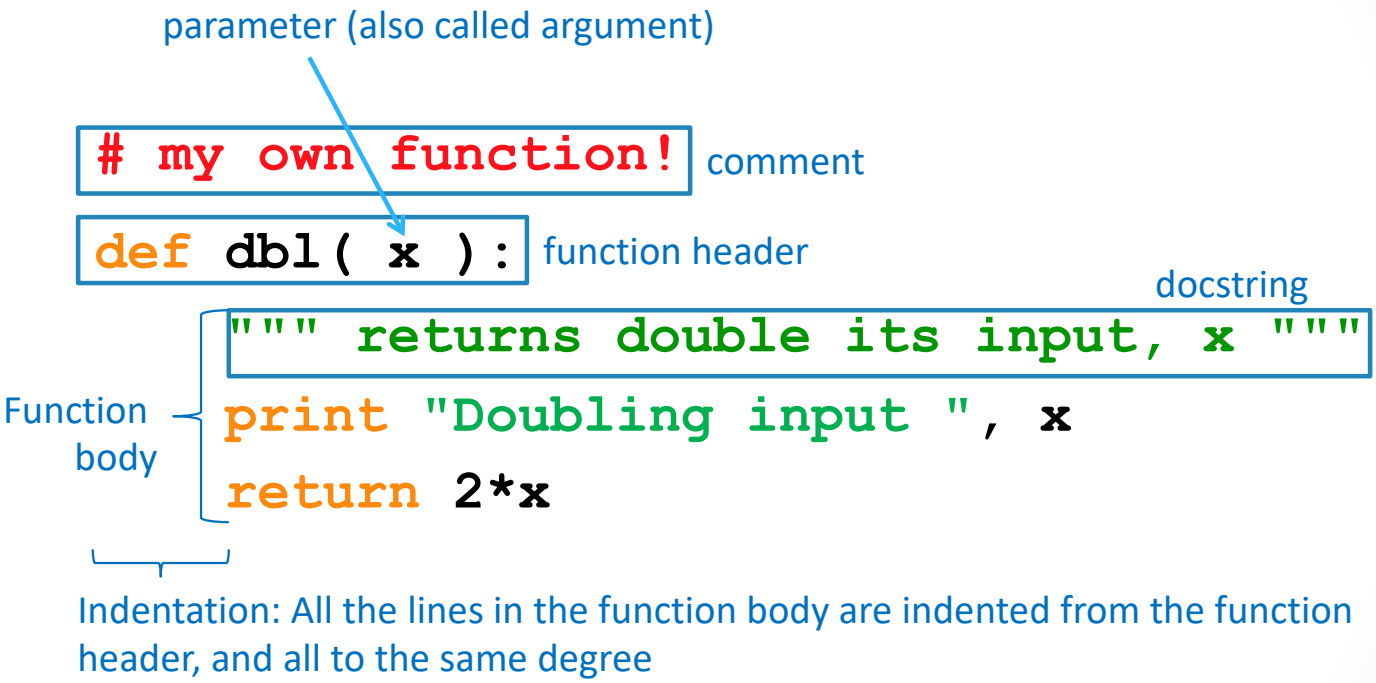
keywords

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

Comments

They begin with #

Essential Definitions and Rules *(do memorize)*



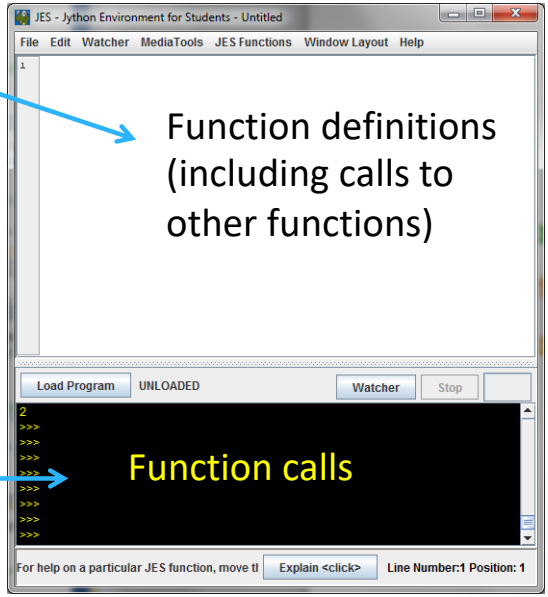
Flow of Execution

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

Define

Call the function

```
>>> 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)

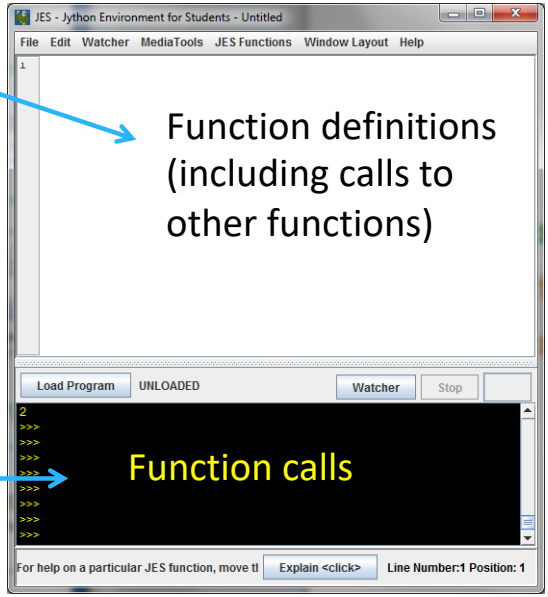
Parameters are special variables

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

x can only be accessed within the function

x | 21

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



Function definitions (including calls to other functions)

Function calls

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.

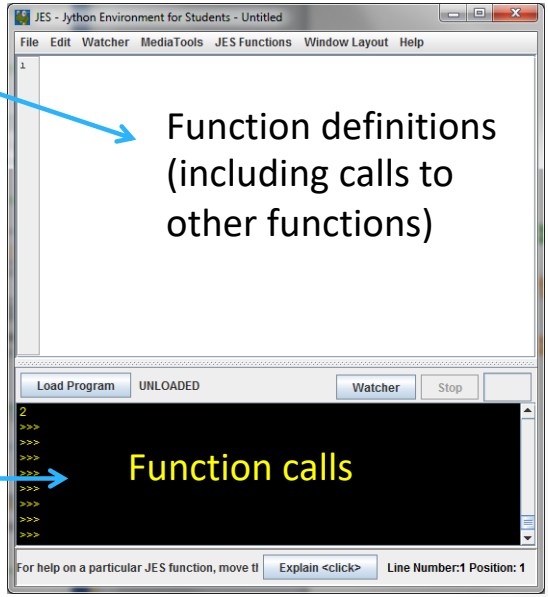
Multiple parameters are allowed

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

x 21

y 2

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



Function definitions (including calls to other functions)

Function calls

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)

Which of the following contains a function call?

```
(1)type(4.5)
(2)def dbl(x):
       return 2*x
(3)area(2, 9)
(4)print("Hello")
```

A. (3) only
B. (2) and (3)
C. (1), (3), and (4)
D. All of (1), (2), (3), and (4) include a function call

No parameters is also allowed

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

doesn't have a parameter & that's okay

```
>>> fortyTwo( )
42
```

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

No return statement is also allowed

```
# my own function!
def printName( ):
    """ prints a message, no return statement"""
    print("My name is Turtle")


>>> printName()
```

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)

Print vs. return

Definition "A"	Definition "B"
`def squared(x):` `  return (x * x)`	`def squared(x):` `  print (x * x)`

Your job: In the following function calls decide which version of squared was used—or whether it is impossible to tell from the output given.

Code	Circle one answer

Python shell
`>>> squared(7)`
`49`
`>>>`
	A B C can't tell

>>> squared(7)
49

Print vs. return

Definition "A"	Definition "B"
`def squared(x):` `  return (x * x)`	`def squared(x):` `  print (x * x)`

returns x*x
prints (x*x)

Your job: In the following function calls decide which version of squared was used—or whether it is impossible to tell from the output given.

```
>>> 2 * squared(3)
18
>>>
```

A B C can't tell

```
>>> 2 * squared(3)
18
```

18

Functions can call Functions!!

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

div does not return a value

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

What does halve(85) return?

A. 42
B. 42.5
C. 0
D. None

Testing

- You must follow good defensive coding strategies, including testing your code extensively
- In class we will explore using the pytest framework, refer to code written in lecture for more information