Functions
Turtle Graphics

Introduction to Computer Science!

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

What is your current status in the class?

A. I am registered for the course
B. I am crashing the course (used to be on the waitlist)

If you are not enrolled, what was your (latest) position on the waitlist?

A. Top 10
B. Between 10 and 15
C. Above 15

Announcements

- If you were among the top 10 students in the waitlist, see me after class
- Homework 00 and 01 are due during lab section tomorrow
- During lab tomorrow, please sit in your assigned seats according to the seating chart posted on the website:

https://ucsb-cs8-f18.github.io/info/seating09am/

https://ucsb-cs8-f18.github.io/info/seating10am/

https://ucsb-cs8-f18.github.io/info/seating11am/

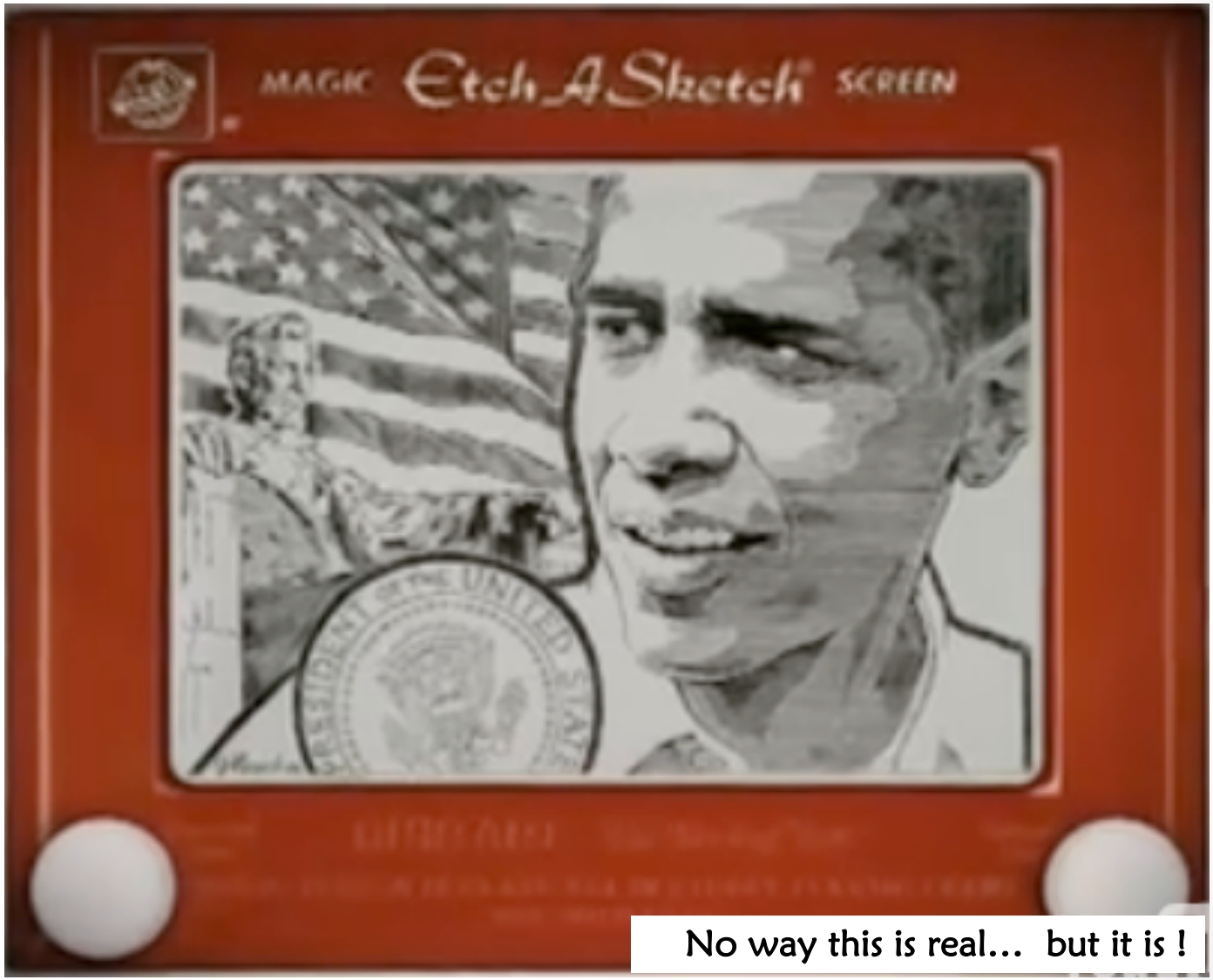
https://ucsb-cs8-f18.github.io/info/seating12pm/

Which of the following contains a function call?

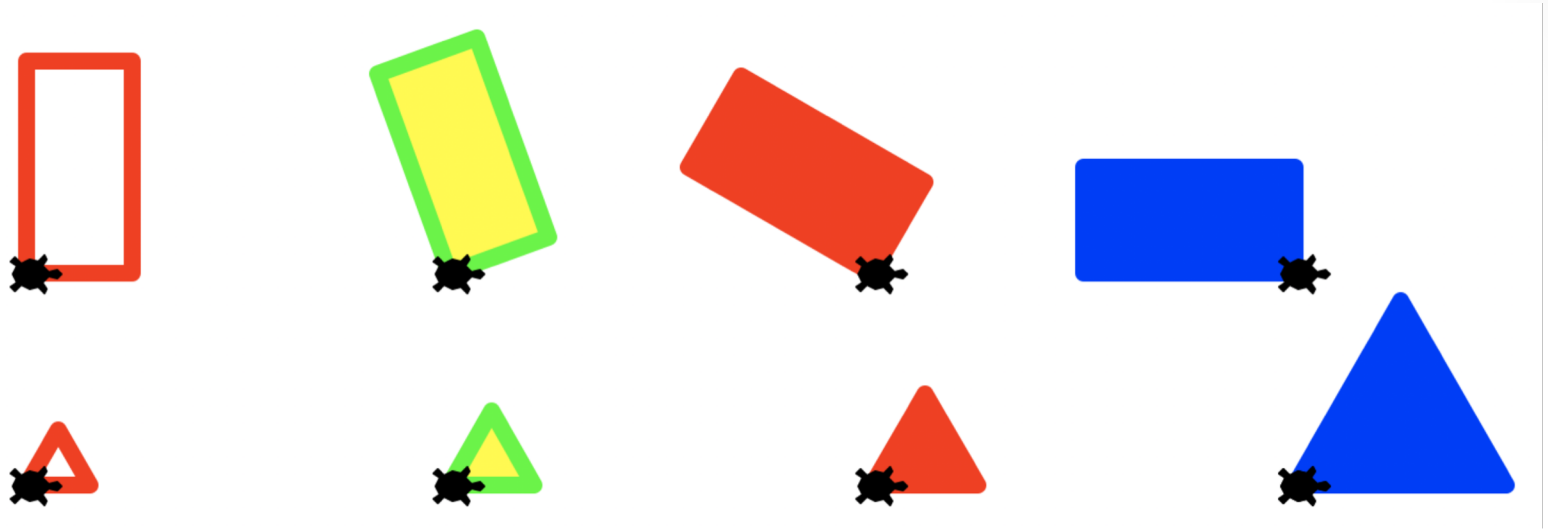
```
(1)type(4.5)
(2)def dbl(x):
       return 2*x
(3)area(2, 9)
(4)print("Hello")
```

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

Etch-a-Sketch ?



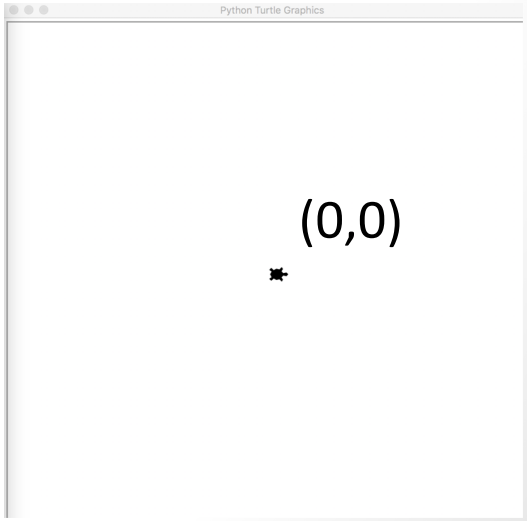
Lab01: Turtle Graphics



```
drawRectangle(t, width, height, tilt, penColor, fillColor)

drawTriangle(t, side, penColor, fillColor)

drawTwoRectangles(t)

drawTwoTriangles(t)
```

Turtle- getting started

```
import turtle
# This statement allows you
to use all the functions in
the turtle package

jane = turtle.Turtle()
# create a new “turtle”
object called jane

jane.shape("turtle")
# change the shape of the
turtle
```



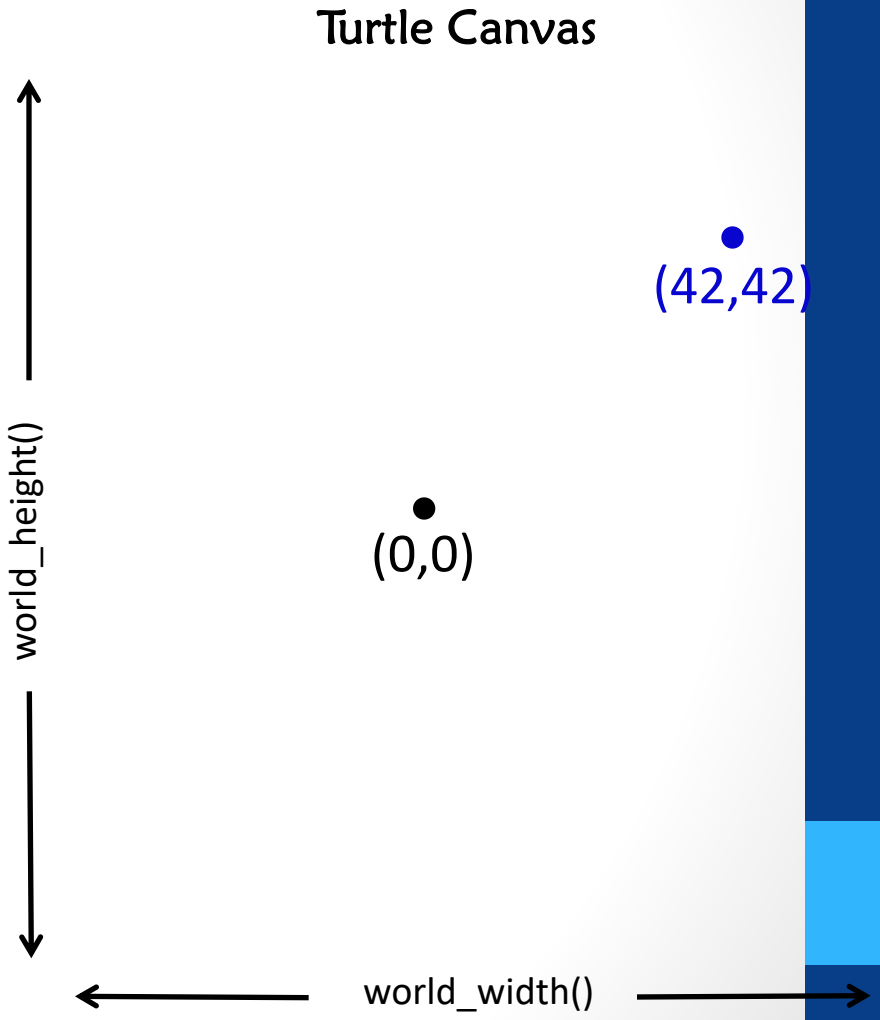
Python's Etch-a-Sketch

```
import turtle
jane = turtle.Turtle()
jane.forward( 100 )
jane.left(60 )
jane.right(90)
jane.width(8)
jane.color("green", "yellow")
jane.up()
jane.forward(50)
jane.down()
```

pixels!

degrees!

90 degrees!



ConcepTest

Which order of instructions produces the following output:

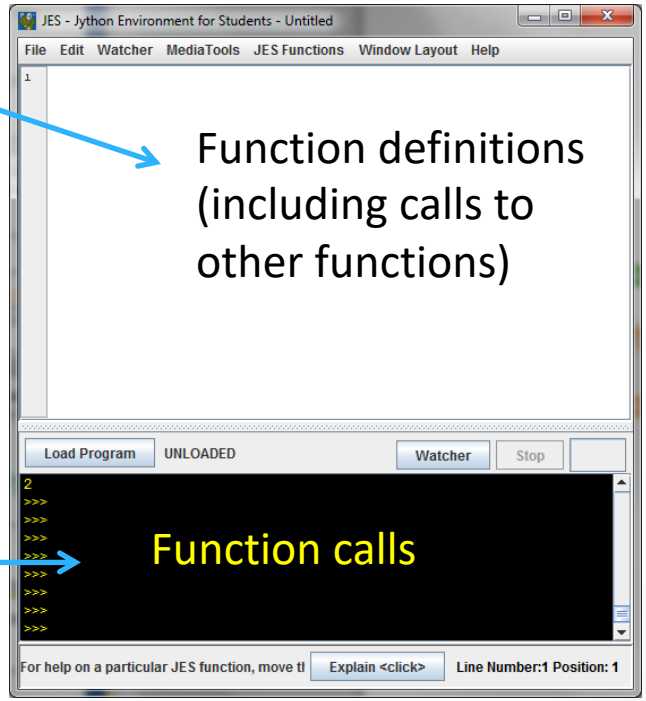
Red: Initial position and orientation
Black: Final position and orientation

```
jane.forward(100)   #(1)
jane.left(90)       #(2)
jane.forward(50)    #(3)
jane.right(90)      #(4)
```

- A. (1), (2), (3), (4)
- B. (4), (3), (2), (1)
- C. (1), (4), (2), (3)
- D. (1), (4), (3), (2)

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)

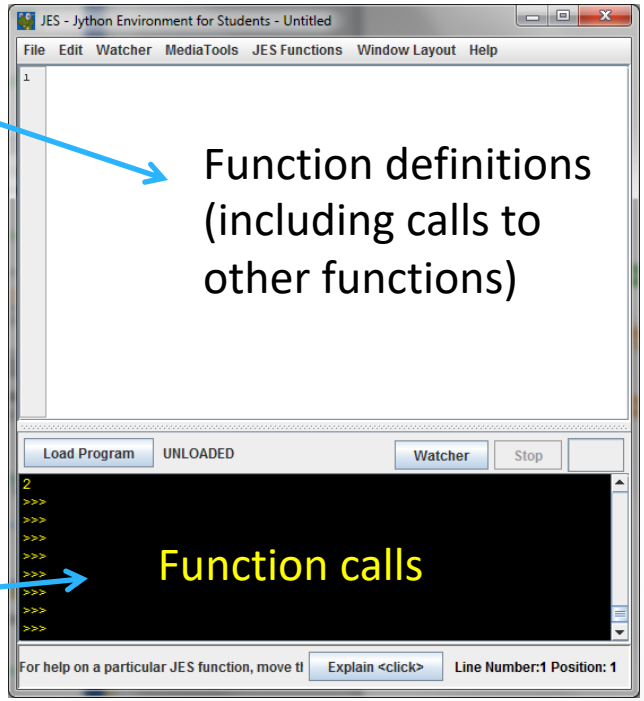
Parameters are special variables

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

x has "local" scope: cannot access it outside of dbl

x

```
>>> y = 21
>>> dbl(y)
```



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.

What is/are the bug(s) in the following code?

```
def dbl(x):
return 2*x
y = 2
x = 5
dbl(y)
print(x, y, dbl(y))
```

A. No bugs. The code is fine

B. The function body is not indented

C. We are referring to x outside the definition of the function

D. Both B and C are bugs

Global vs. Local variables
What is the output of this code?

```
def dbl(x):
    x = 2*x
    return x
y = 2
x = 5
x=dbl(y)
print(x, y, dbl(y))
```

A. 10 4 8
B. 5 2 4
C. 10 2 4
D. None of the above

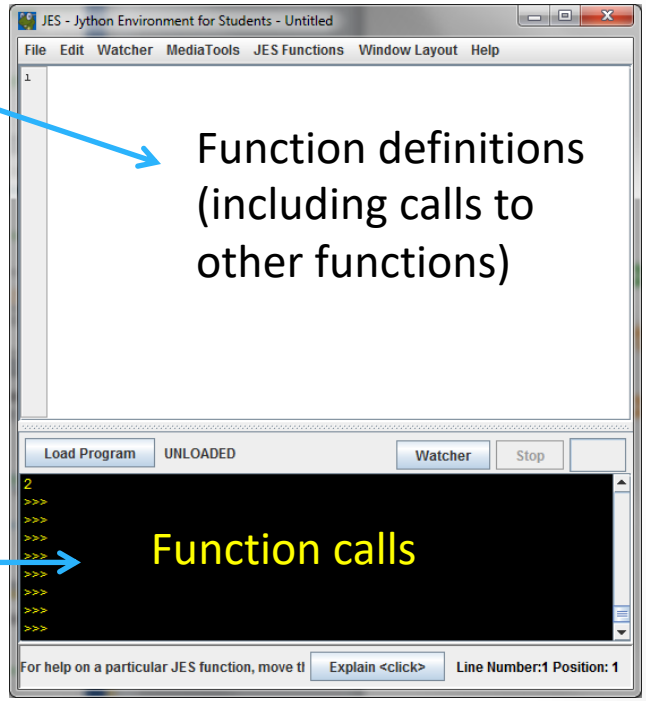
Multiple parameters are allowed

```
# 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)