Passing Parameters

```python
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
```
# Draw a frame of the animation

def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down

with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(\(y\)):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, \(y\), 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
import drawSvg as draw

# Draw a frame of the animation
def draw_frame(y):
    d = draw.Drawing(250, 250, origin='center')
    d.append(draw.Circle(0, y, 25, fill='lime'))
    return d

# bounce the ball up and down
with draw.animate_jupyter(draw_frame, delay=0.05) as anim:
    anim.draw_frame(100)
    anim.draw_frame(90)
    anim.draw_frame(80)
Returning Parameters

# the function sqrt takes as input a number and returns a number
from math import sqrt
x = sqrt(4)
print(sqrt(4))

# the function "+" takes as input two numbers and returns a number
x = 8 + 12
print(8 + 12)

# the function len takes as input a string and returns an integer
print(len("eggplant"))

# the return value of one function can be the input to another
print(int(8.485) + 12)
print(int( sqrt(72) ) + 12)
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)

y = 24 / compute_four()
print(y)

print(compute_four())
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x) # 4

y = 24 / compute_four()
print(y) # 36

print(compute_four()) # 4
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)  # 4

y = 24 / compute_four()
print(y)  # 6

print(compute_four())
def compute_four():
    return 24 / 4 - 2

x = compute_four()
print(x)  # 4

y = 24 / compute_four()
print(y)  # 6

print(compute_four())  # 4
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)

x = add_five(z)
print(x)
print(add_five(z))
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)

x = add_five(z)
print(x)

print(add_five(z))
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z)  # 4

x = add_five(z)
print(x)  # 9
print(add_five(z))
def add_five(x):
    x = x + 5
    return x

z = 4
add_five(z)
print(z) # 4

x = add_five(z)
print(x) # 9

print(add_five(z)) # 9
def return_something(x):
    if (x > 0):
        return 1
    else:
        return -1

def print_something(x):
    if (x > 0):
        print(1)
    else:
        print(-1)

x = return_something(10) + return_something(-8)

y = print_something(12)
print(y)
def return_something(x):
    if ( x > 0 ):
        return 1
    else:
        return -1

def print_something(x):
    if ( x > 0 ):
        print(1)
    else:
        print(-1)

x = return_something(10) + return_something(-8)
y = print_something(12)
print(y)
from random import uniform

def cheap_magic_8_ball():
    r = uniform(0, 4)

    if r > 3:
        print("Most likely")
    elif r > 2:
        print("Ask again later")
    elif r > 1:
        print("Don't count on it")
    else:
        print("No")

cheap_magic_8_ball()
from random import uniform

def cheap_magic_8_ball():
    r = uniform(0, 4)

    if r > 3:
        return "Most likely"
    elif r > 2:
        return "Ask again later"
    elif r > 1:
        return "Don't count on it"
    else:
        return "No"

print(cheap_magic_8_ball())
# --- DRILL ---
# Write a python function, count_evens, that 
# takes as input a list of numbers and 
# returns the number of even numbers.
def count_evens( L ):
    count = 0
    for l in L:
        if( l % 2 == 0 ):
            count = count + 1
    return count

c = count_evens( [1,2,3,4,5,6] )
Passing & Returning Parameters

```python
def return_two_things(x, y):
    s = x + y
    p = x * y
    return (s, p)

(s, p) = return_two_things(2, 5)
print(s, p)
```
def return_two_things(x, y):
    return (x + y, x * y)

(s, p) = return_two_things(2, 5)
print(s, p)
def return_two_things(x, y):
    return (x+y, x*y)
    print(x, y)

(s, p) = return_two_things(2, 5)
print(s, p)
Passing & Returning Parameters

```python
x = 17
c = 2

while( c < x ):
    if( x % c == 0 ):
        print( "not prime" )
        break
    c = c + 1

if( c == x ):
    print( "is prime" )
```
def is_prime(x):
    c = 2
    while (c < x):
        if (x % c == 0):
            return( "not prime" )
        c = c + 1
    return( "is prime" )
Local vs. Global Variables

def some_function():
    x = 4
    print(x)

some_function()
print(x)
Local vs. Global Variables

def some_function():
    x = 4
    print(x)

some_function()  
print(x)  

error: x is not defined
Local vs. Global Variables

def print_x():
    global x
    x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()
print_x()
print_x()
Local vs. Global Variables

def print_x():
    global x
    x = x + 1
    print(x)

x = 5
print(x)  # x is a global variable
print_x()
print_x()
print_x()
Local vs. Global Variables

def print_x():
    global x
    x = x + 1
    print(x)

x = 5
    # x is a global variable
print_x()  # 6
print_x()  # 7
Local vs. Global Variables

```python
def print_x():
    #global x
    x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()
```
Local vs. Global Variables

def print_x():
    #global x
    x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()  
error: local variable 'x' referenced before assignment
Local vs. Global Variables

def print_x():
    #global x
    #x = x + 1
    print(x)

x = 5  # x is a global variable
print_x()
Local vs. Global Variables

def print_x():
    #global x
    #x = x + 1
    print(x)

x = 5
# x is a global variable
print_x()
from math import pi

def compute_circle_area(r):
    global pi
    pi = 3  # changes the value of pi. Not cool!
    return pi * r * r

compute_circle_area(10)
Local vs. Global Variables

```python
x = 1

def test(x):
    global x
    x = x + 1
    print(x)

test(10)  # error: name 'x' is local parameter and global
```