

MORE CODING BASICS WITH SOME TURTLE

ANNOUNCEMENT

- Homework 2 will be posted this Thursday. Due dates will be adjusted.
- Missing quiz policy: no quiz submissions will be accepted late, except in the case of a documented medical emergency.
- Since we drop a quiz grade, there is NO make-up quiz unless there is a medical emergency.
- If you have situations where you suspect you will miss the quiz regularly (e.g. military service), please discuss with the instructor via writing, supporting documents are required.
- If you do miss a quiz, you will have other opportunities to earn points throughout the class

Learn to program

= Learn to solve problems

PYTHON

MORE CODING BASICS

Credit: lecture notes modeled after <http://www.openbookproject.net/thinkcs/python/english2e/index.html>

What is a function?

Syntax

Def FUNCTIONNAME(parameters):
----- STATEMENTS

header

body

↑
Indentation (4 empty spaces)

User-defined Functions

```
def new_line():  
    print  
    print "The weirdest movie at Sundance this year  
is:"  
    new_line()  
    print "Swiss Army Man."
```


A white line-art illustration of a computer monitor with a stand, centered on a black background. The monitor's screen area is defined by a white border and contains white text. A small white dot is visible at the top center of the monitor's bezel.

The weirdest movie at Sundance this year is:

Swiss Army Man.

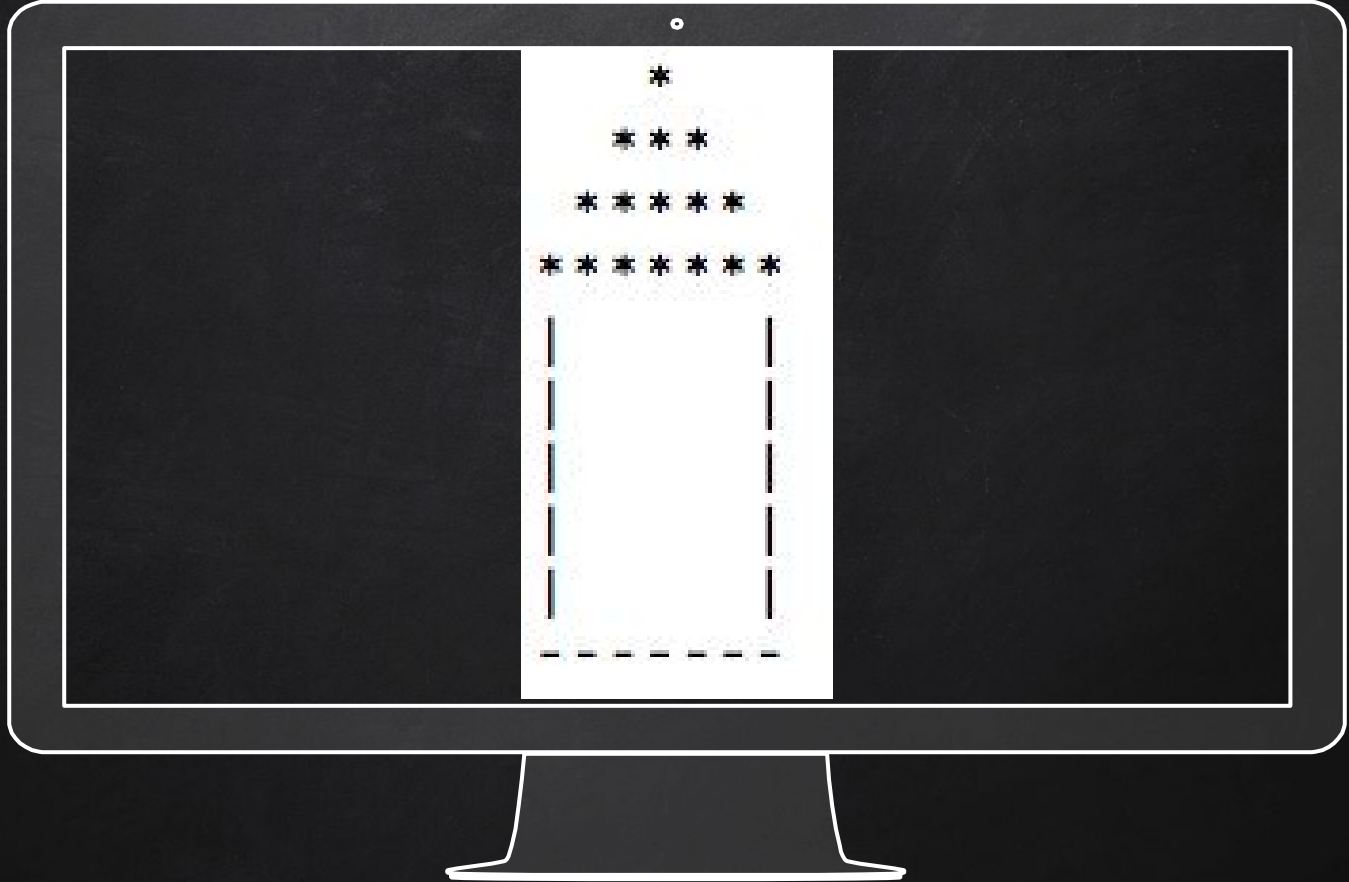
```
def new_line():  
    print  
    print "The weirdest movie at Sundance this year  
is:"  
    new_line()  
    new_line()  
    print "Swiss Army Man."
```

A white line-art illustration of a computer monitor with a stand. The monitor's screen is filled with a dark gray background, and the text is written in white. A small white dot is centered at the top of the monitor's bezel.

The weirdest movie at Sundance this year is:

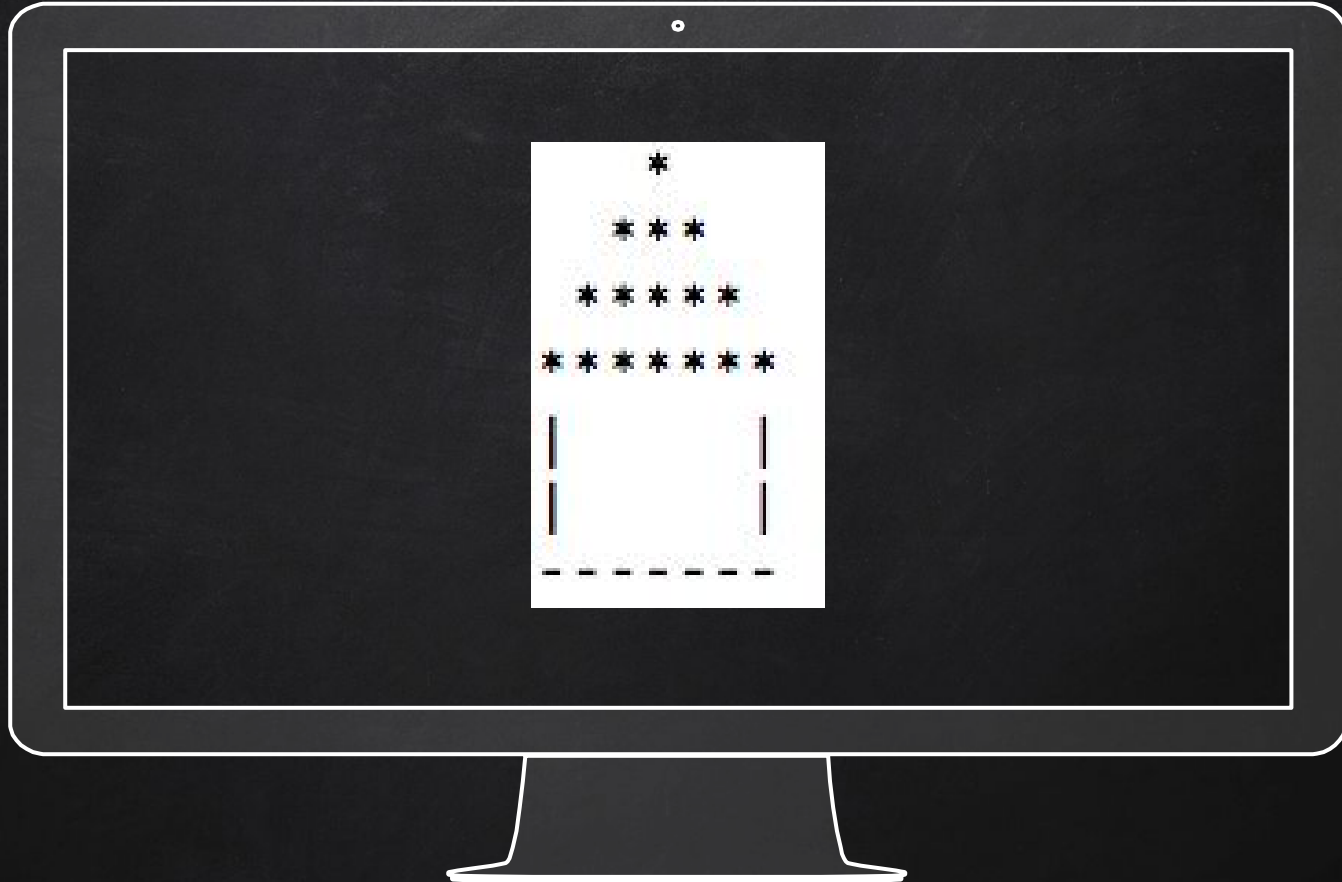
Swiss Army Man.

```
print ' * '  
print ' *** '  
print ' ***** '  
print ' ***** '  
for h in range(0,5):  
    print '| |'  
print '-----'
```



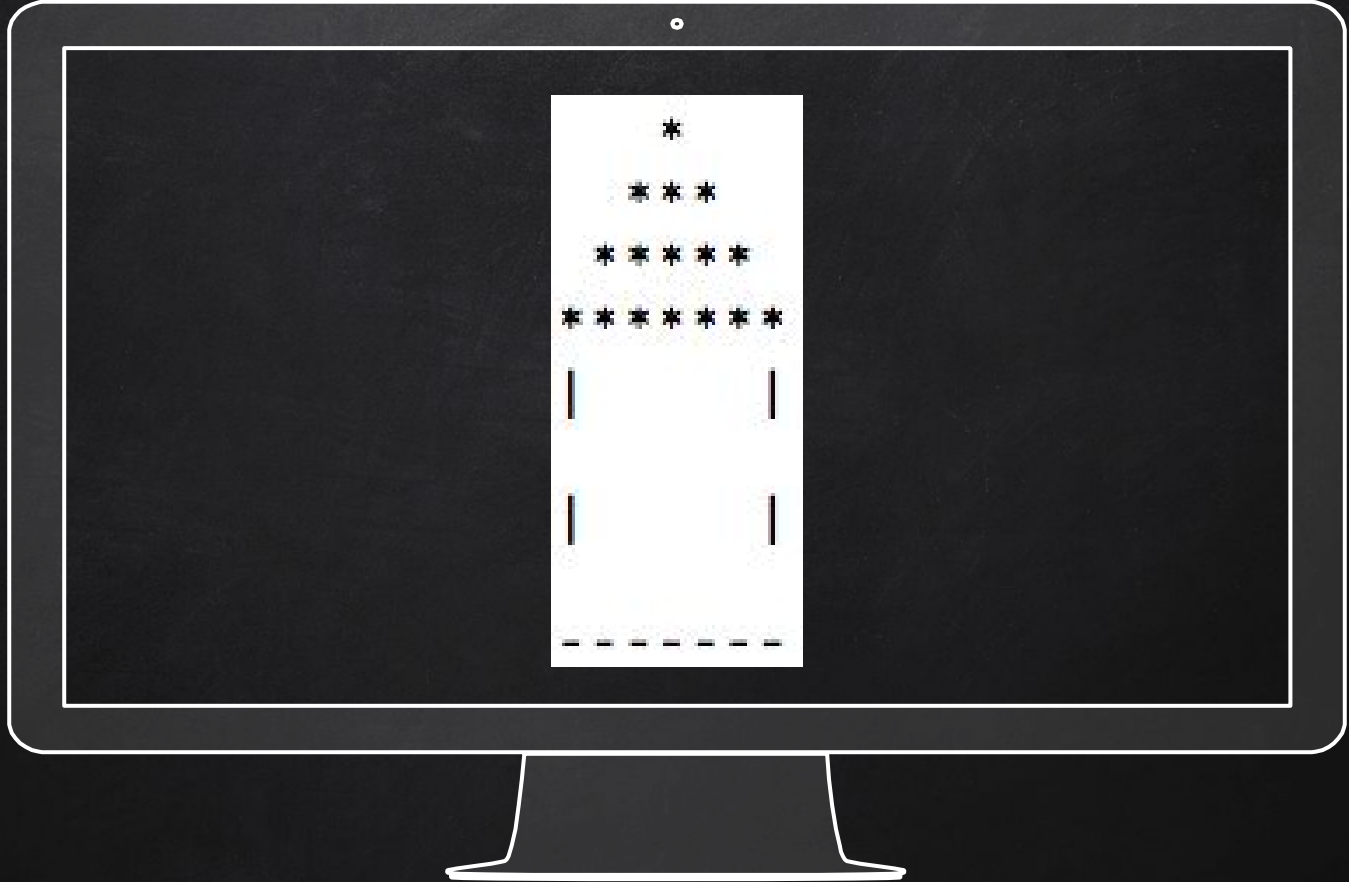
```
def print_house(height):  
    print ' * '  
    print ' *** '  
    print ' ***** '  
    print '*****'  
    for h in range(0,height):  
        print '|  |'  
        print '-----'  
print_house(2)
```

<http://www.pythontutor.com/index.html>



A **Function** can be called repeatedly.
Function can call another function.


```
def new_line():
    print
def print_house(height):
    print ' * '
    print ' *** '
    print ' ***** '
    print '*****'
    for h in range(0,height):
        print '| |'
        new_line()
    print '-----'
print_house(2)
```



Advantages of using a function:

- Group complex computations
- Smaller code for repetitive ops

Flow of Execution

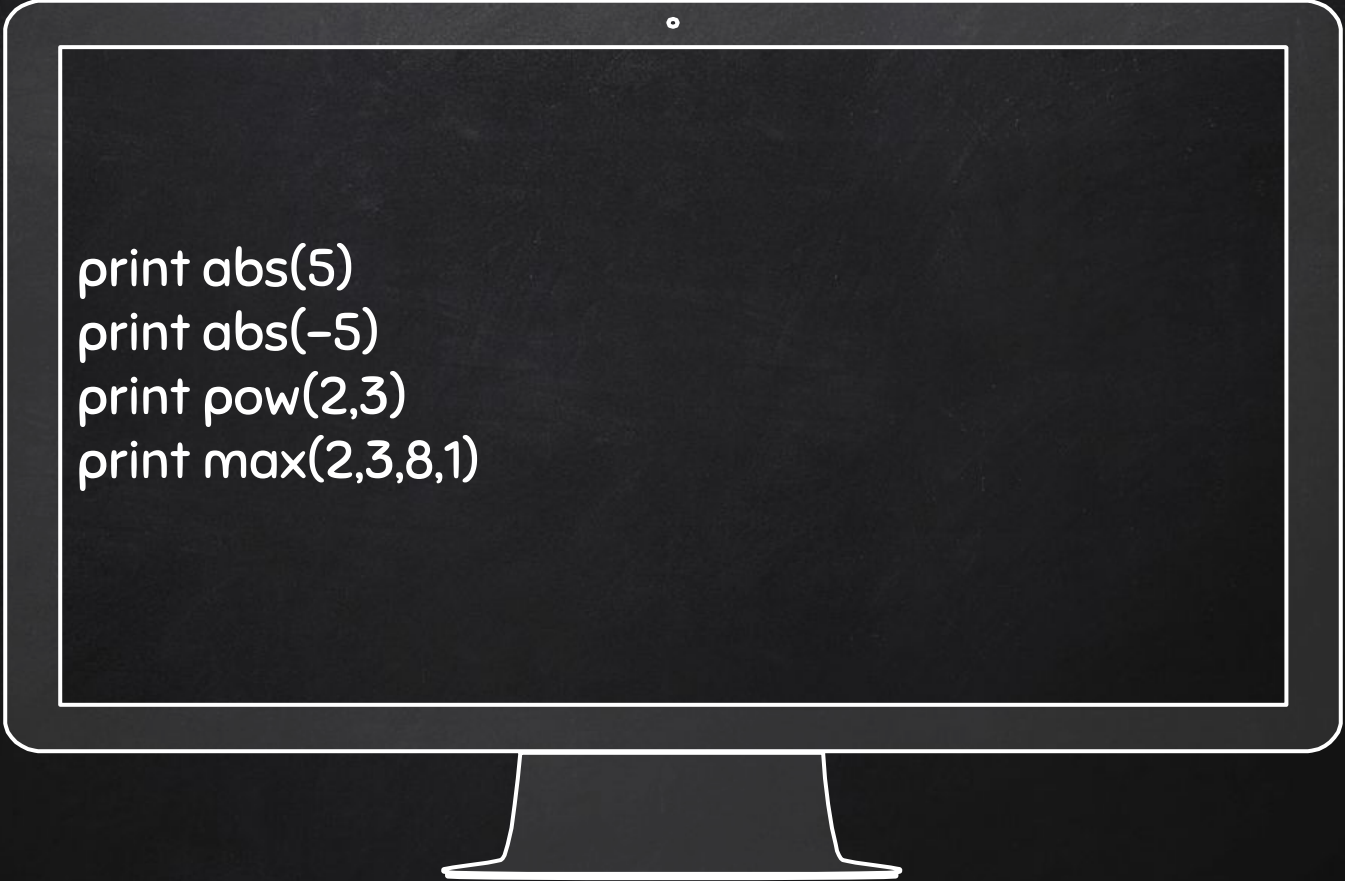
<http://www.pythontutor.com/index.html>

Forward bottom will show step by step
execution of the program

Build-in Functions

Built-in Functions				
<code>abs()</code>	<code>divmod()</code>	<code>input()</code>	<code>open()</code>	<code>staticmethod()</code>
<code>all()</code>	<code>enumerate()</code>	<code>int()</code>	<code>ord()</code>	<code>str()</code>
<code>any()</code>	<code>eval()</code>	<code>isinstance()</code>	<code>pow()</code>	<code>sum()</code>
<code>basestring()</code>	<code>execfile()</code>	<code>issubclass()</code>	<code>print()</code>	<code>super()</code>
<code>bin()</code>	<code>file()</code>	<code>iter()</code>	<code>property()</code>	<code>tuple()</code>
<code>bool()</code>	<code>filter()</code>	<code>len()</code>	<code>range()</code>	<code>type()</code>
<code>bytearray()</code>	<code>float()</code>	<code>list()</code>	<code>raw_input()</code>	<code>unichr()</code>
<code>callable()</code>	<code>format()</code>	<code>locals()</code>	<code>reduce()</code>	<code>unicode()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>long()</code>	<code>reload()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>map()</code>	<code>repr()</code>	<code>xrange()</code>
<code>cmp()</code>	<code>globals()</code>	<code>max()</code>	<code>reversed()</code>	<code>zip()</code>
<code>compile()</code>	<code>hasattr()</code>	<code>memoryview()</code>	<code>round()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hash()</code>	<code>min()</code>	<code>set()</code>	
<code>delattr()</code>	<code>help()</code>	<code>next()</code>	<code>setattr()</code>	
<code>dict()</code>	<code>hex()</code>	<code>object()</code>	<code>slice()</code>	
<code>dir()</code>	<code>id()</code>	<code>oct()</code>	<code>sorted()</code>	

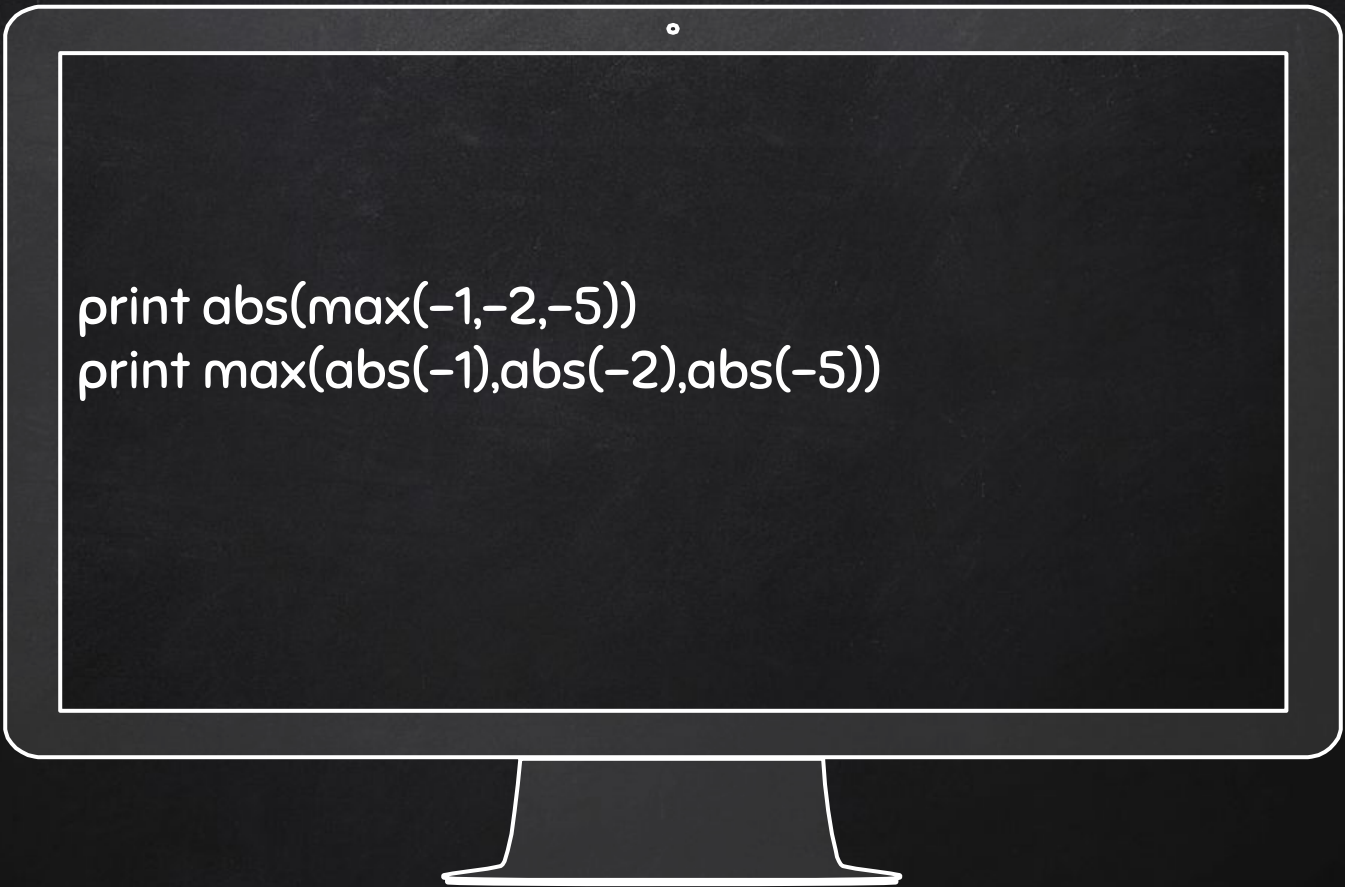
<https://docs.python.org/2/library/functions.html>

A white line-art illustration of a computer monitor with a stand. The monitor's screen is filled with a dark gray background, and the text is written in white. The code is as follows:

```
print abs(5)
print abs(-5)
print pow(2,3)
print max(2,3,8,1)
```



Functions can be composed.

A white outline of a computer monitor is centered on a black background. The monitor's screen area contains two lines of Python code in white text. The first line is `print abs(max(-1,-2,-5))` and the second line is `print max(abs(-1),abs(-2),abs(-5))`.

```
print abs(max(-1,-2,-5))  
print max(abs(-1),abs(-2),abs(-5))
```



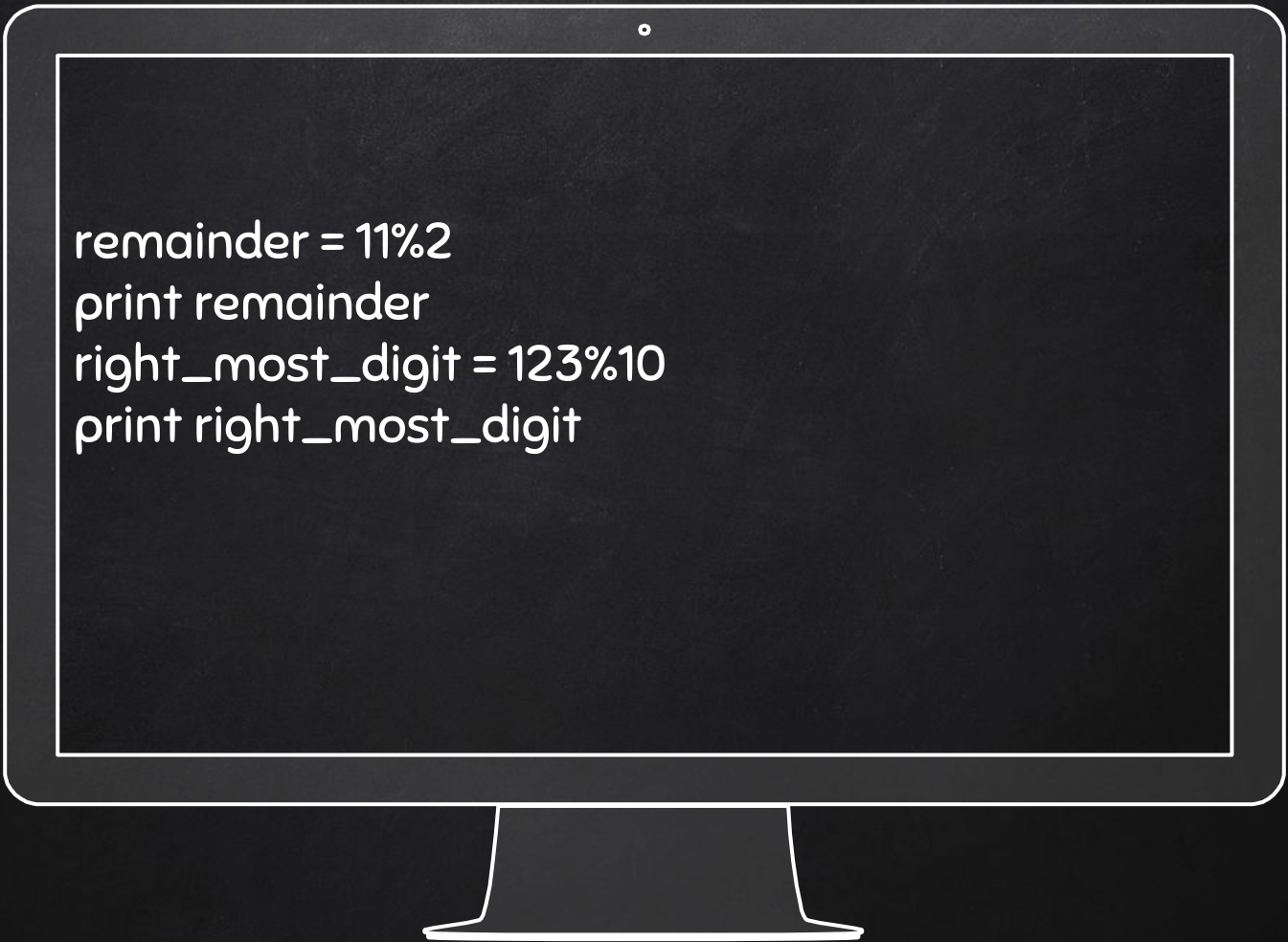
1
5

Local variables are only defined inside the function.

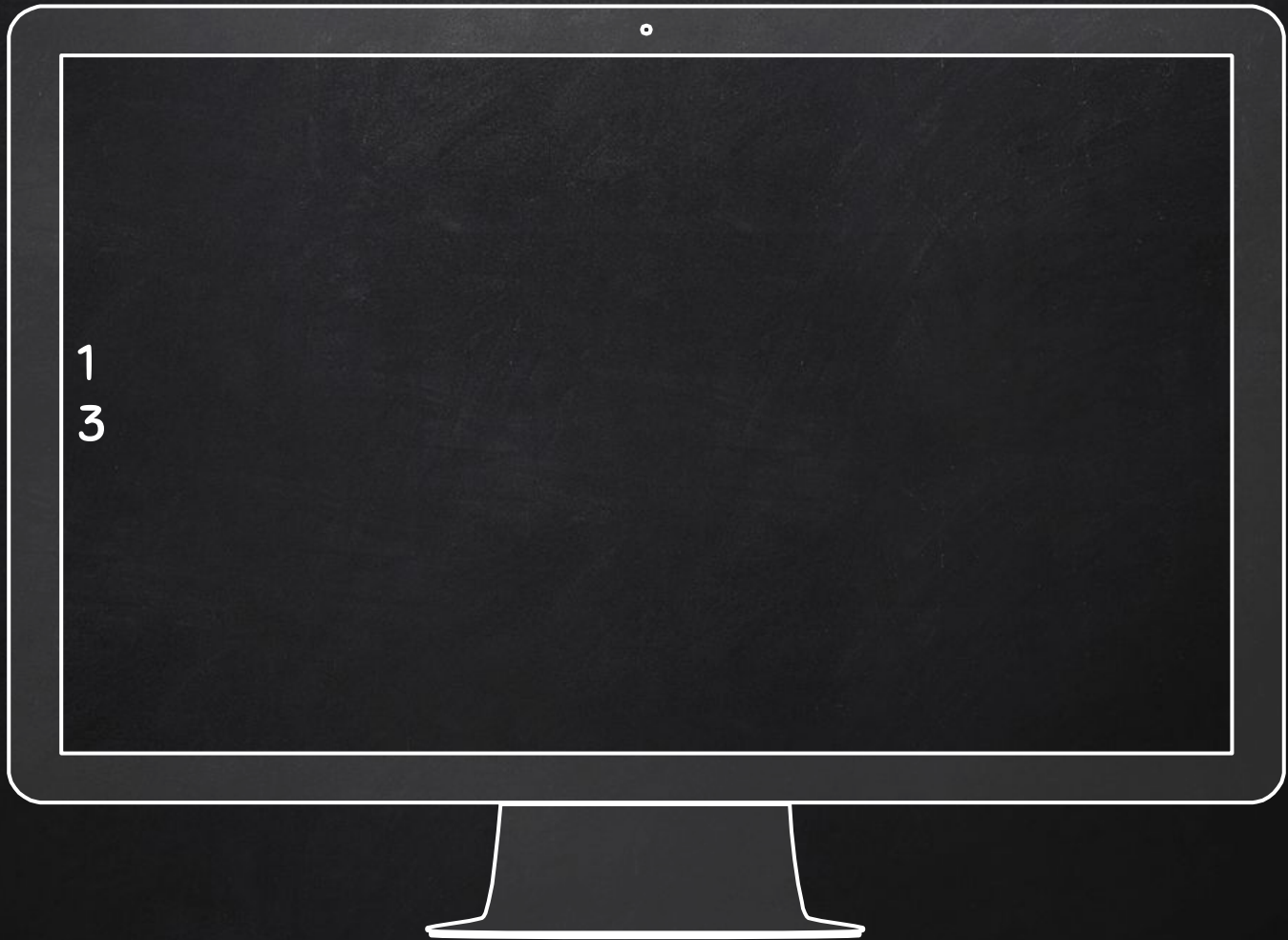
```
def print_house(height):  
    print ' * '  
    print ' *** '  
    print ' ***** '  
    print '*****'  
    for h in range(0,height):  
        print '|  |'  
        print '-----'  
        print type(height)  
print_house(2)
```

```
def print_house(height):  
    print ' * '  
    print ' *** '  
    print ' ***** '  
    print ' ***** '  
    for h in range(0,height):  
        print '|  |'  
        print '-----'  
print_house(2)  
print type(height)
```

Modulus operator: %

A white line-art illustration of a computer monitor with a stand. The screen area is a dark rectangle containing white text. The text consists of four lines of Python code: 'remainder = 11%2', 'print remainder', 'right_most_digit = 123%10', and 'print right_most_digit'.

```
remainder = 11%2  
print remainder  
right_most_digit = 123%10  
print right_most_digit
```

1
3

Conditionals

Boolean Values and Expressions

True, False

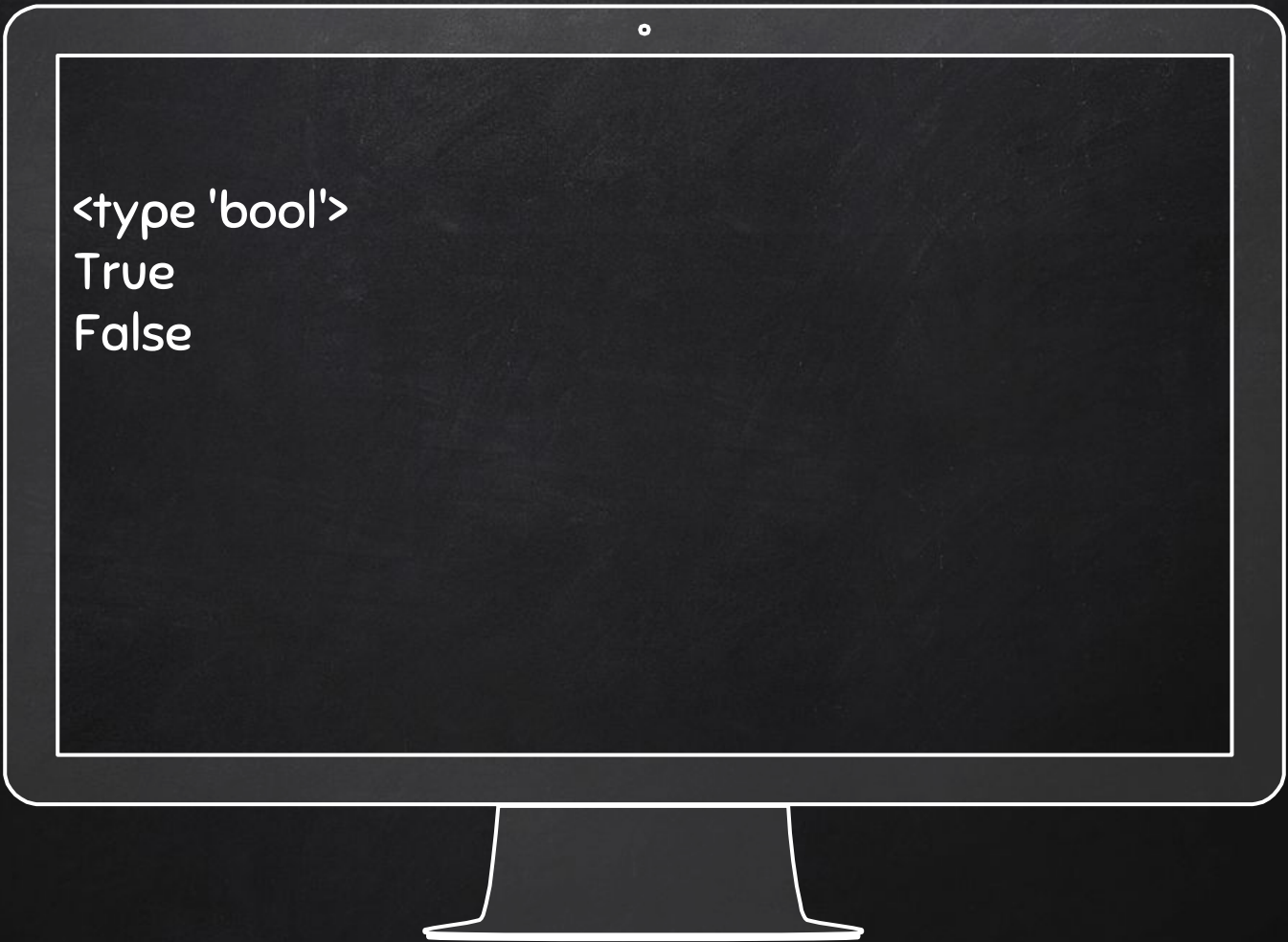
==, !=, >=, <=, >, <

A white line-art illustration of a computer monitor with a stand. The screen is black and contains three lines of white Python code. The code is: print type(True), print 5==5, and print 4==5.

```
print type(True)
```

```
print 5==5
```

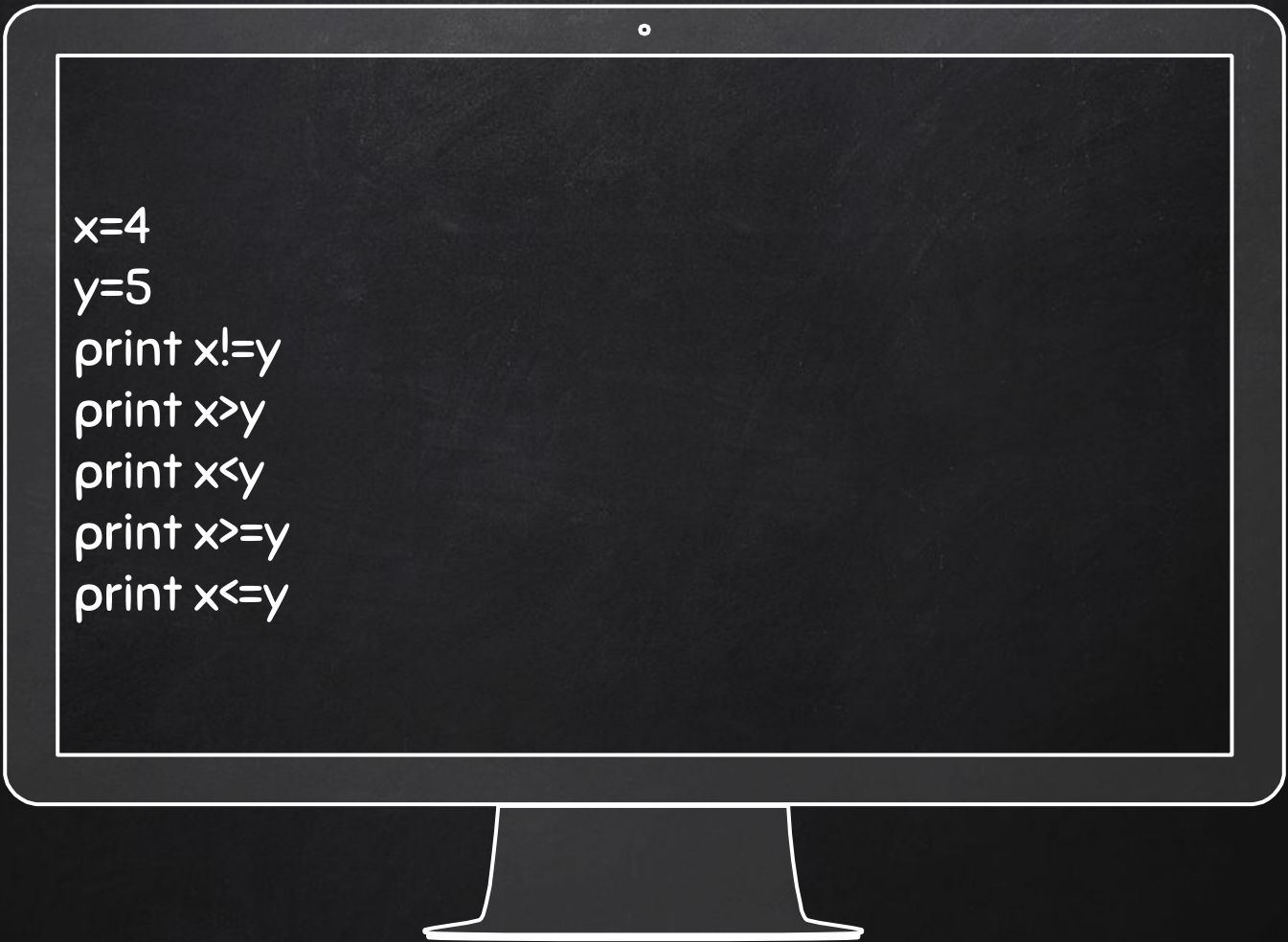
```
print 4==5
```

A white line-art illustration of a computer monitor with a stand. The screen is black and contains white text. At the top center of the bezel is a small white circle. The text on the screen is left-aligned and consists of three lines: a type declaration, the value 'True', and the value 'False'.

```
<type 'bool'>
```

```
True
```

```
False
```

A computer monitor with a white outline and a small circle at the top center. The screen is black and contains white text. The text is a series of Python code lines: x=4, y=5, print x!=y, print x>y, print x<y, print x>=y, and print x<=y.

```
x=4
y=5
print x!=y
print x>y
print x<y
print x>=y
print x<=y
```

```
y=5
```

```
print x!=y
```

```
print x>y
```

```
print x<y
```

```
print x>=y
```

```
print x<=y
```



True
False
True
False
True

Boolean Values and Expressions

A computer monitor with a white outline and a small circle at the top center. The screen is black and contains white text. The text is Python code: x=4, y=5, print (x!=y) and (x>y), print (x!=y) or (x>=y), and print not(x==y). The words 'and', 'or', and 'not' are highlighted in orange.

```
x=4
```

```
y=5
```

```
print (x!=y) and (x>y)
```

```
print (x!=y) or (x>=y)
```

```
print not(x==y)
```



False

True

True

If Statement

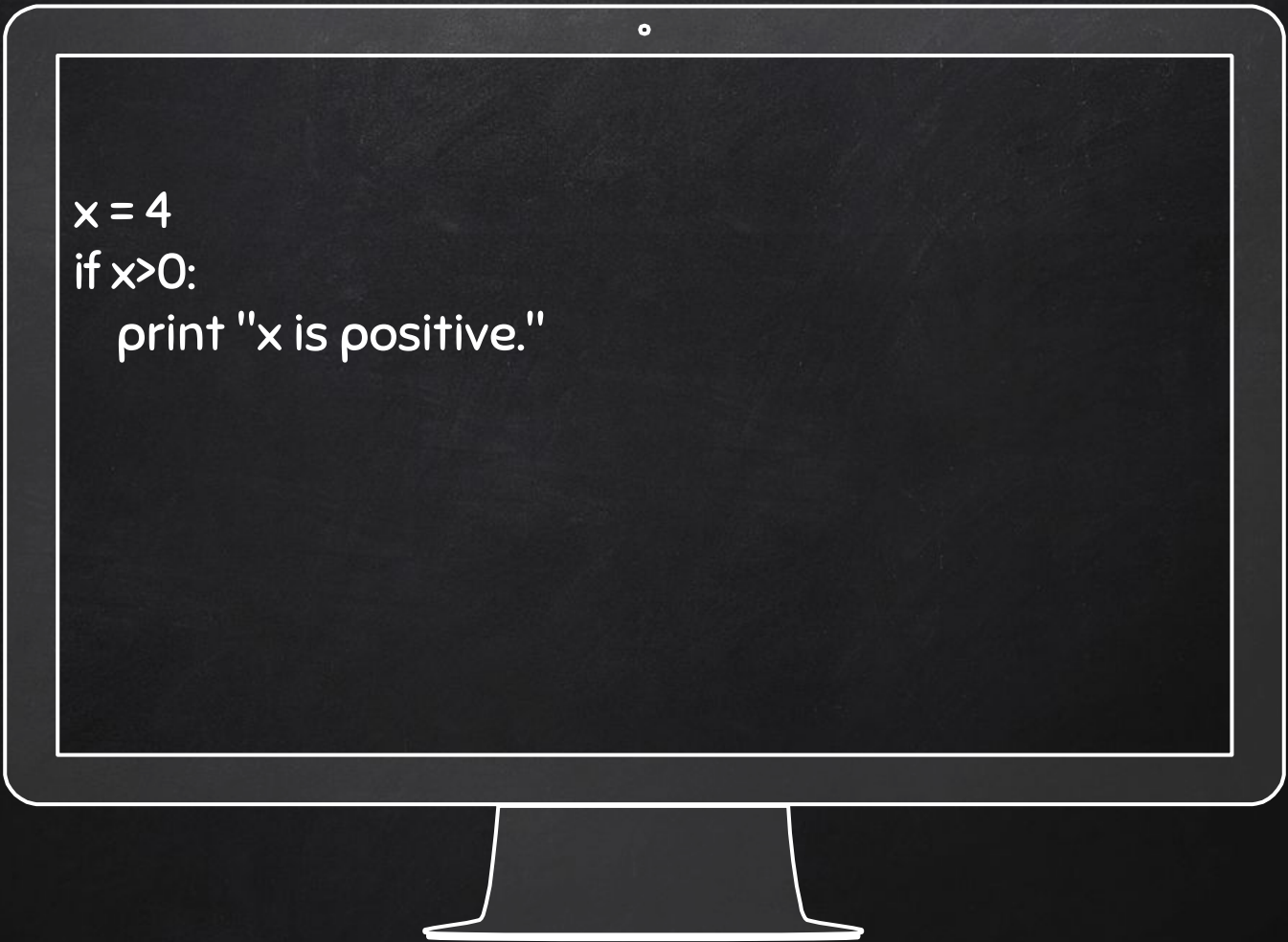
Syntax

IF BOOLEAN EXPRESSION:
----- STATEMENTS

header

body

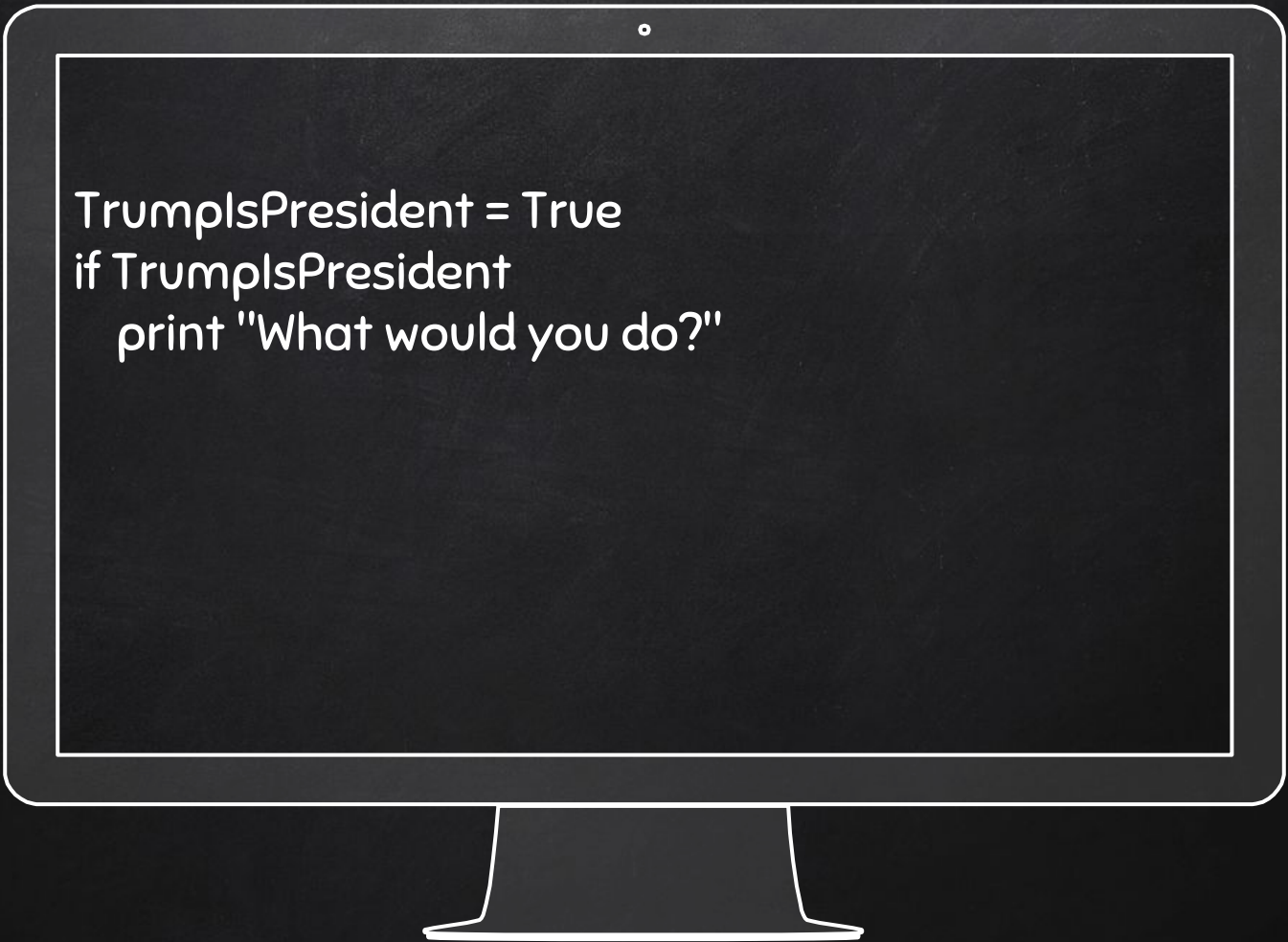
↑
Indentation (4 empty spaces)

A computer monitor with a white outline and a small circle at the top center. The screen is black and contains white text. The text is a Python code snippet. The monitor has a simple stand at the bottom.

```
x = 4
```

```
if x > 0:
```

```
    print "x is positive."
```

A white line-art illustration of a computer monitor with a stand, centered on a black background. The monitor's screen is a white-bordered rectangle containing Python code. At the top center of the monitor's bezel is a small white circle representing a webcam.

```
TrumpsPresident = True
if TrumpsPresident
    print "What would you do?"
```

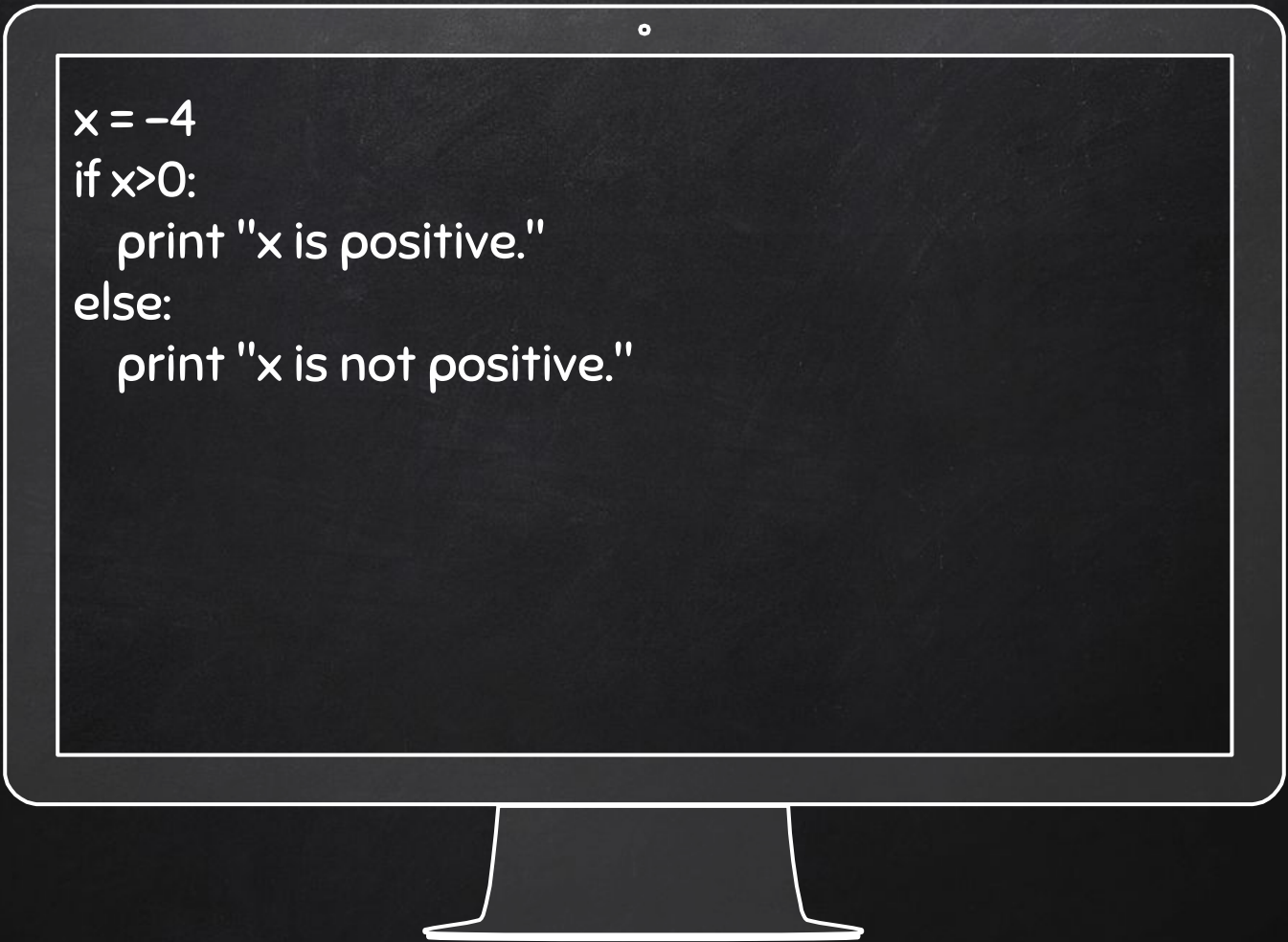
Syntax

if BOOLEAN EXPRESSION:

____STATEMENTS

else:

____STATEMENTS

A white line-art illustration of a computer monitor with a stand. The screen area is a dark rectangle containing white text. The text is a Python code snippet. The monitor has a small circle at the top center of its bezel.

```
x = -4
```

```
if x > 0:
```

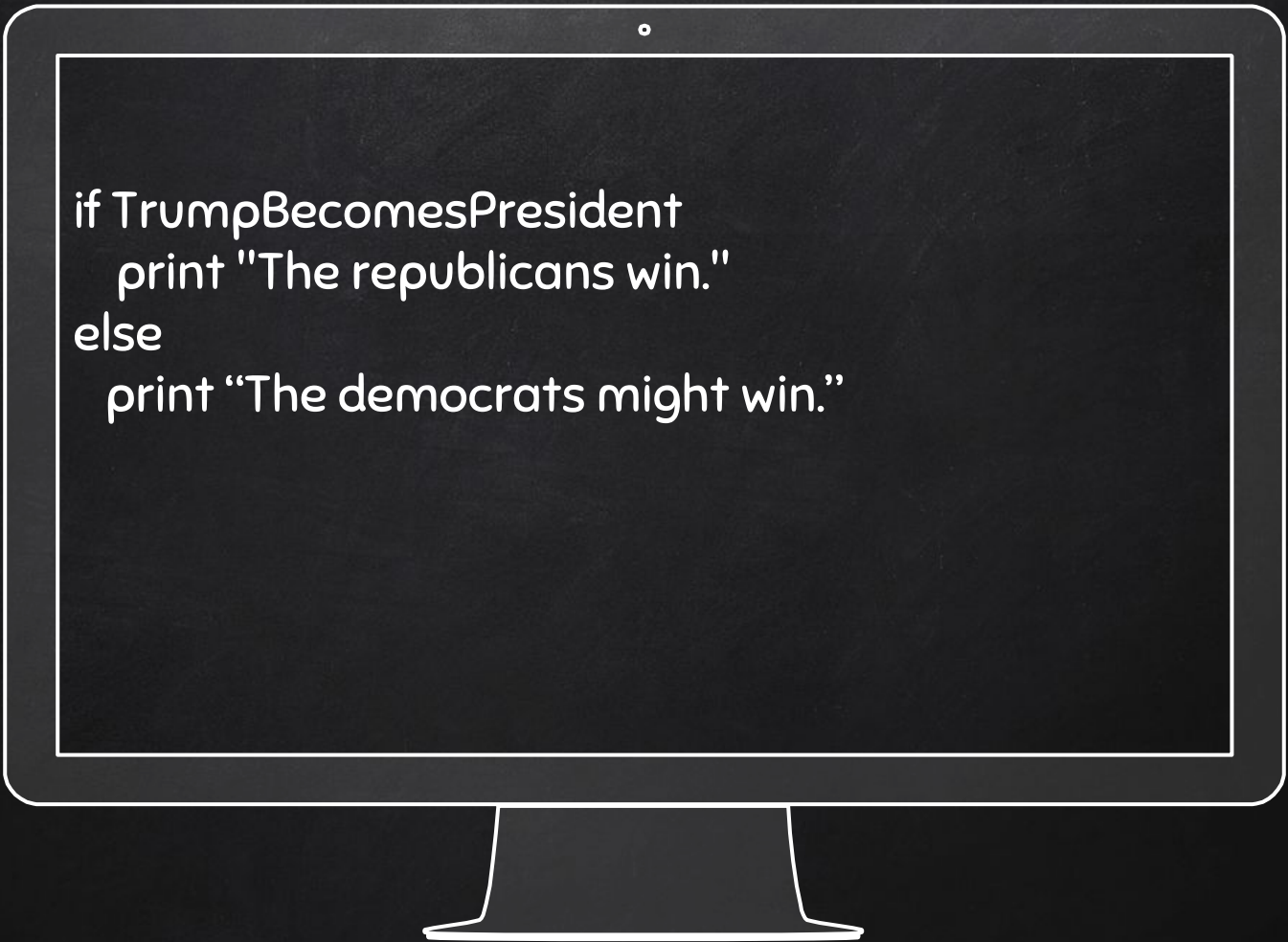
```
    print "x is positive."
```

```
else:
```

```
    print "x is not positive."
```



```
x = 4
if x%2==0:
    print x, "is even."
else:
    print x, "is odd."
```

A white line-art illustration of a computer monitor with a stand. The screen area is a dark rectangle containing white text. The text is a Python conditional statement. At the top center of the monitor's bezel, there is a small white circle.

```
if TrumpBecomesPresident
    print "The republicans win."
else
    print "The democrats might win."
```

Syntax: Chained Conditionals

if BOOLEAN EXPRESSION:

____STATEMENTS

elif BOOLEAN EXPRESSION:

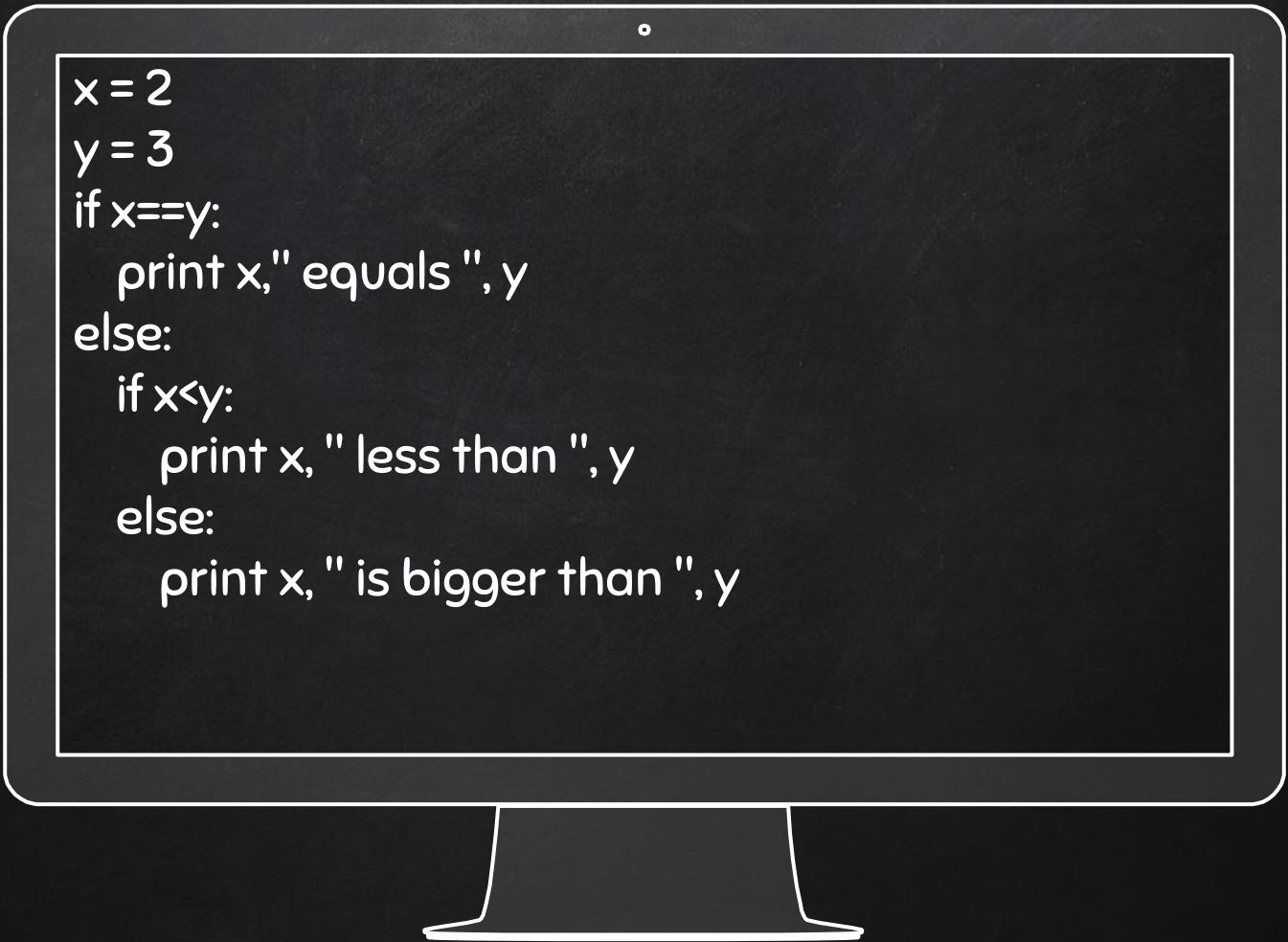
____STATEMENTS

else:

____STATEMENTS

```
x = 0
if x > 0:
    print "x is positive."
elif x < 0:
    print "x is negative."
else:
    print "x is zero."
```

Nested Conditionals

A computer monitor with a white outline and a small circle at the top center. The screen displays Python code in white text on a black background. The code is as follows:

```
x = 2
y = 3
if x==y:
    print x," equals ", y
else:
    if x<y:
        print x, " less than ", y
    else:
        print x, " is bigger than ", y
```

x = 2

y = 3

if x==y:

print x," equals ", y

else:

if x<y:

print x, " less than ", y

else:

print x, " is bigger than ", y

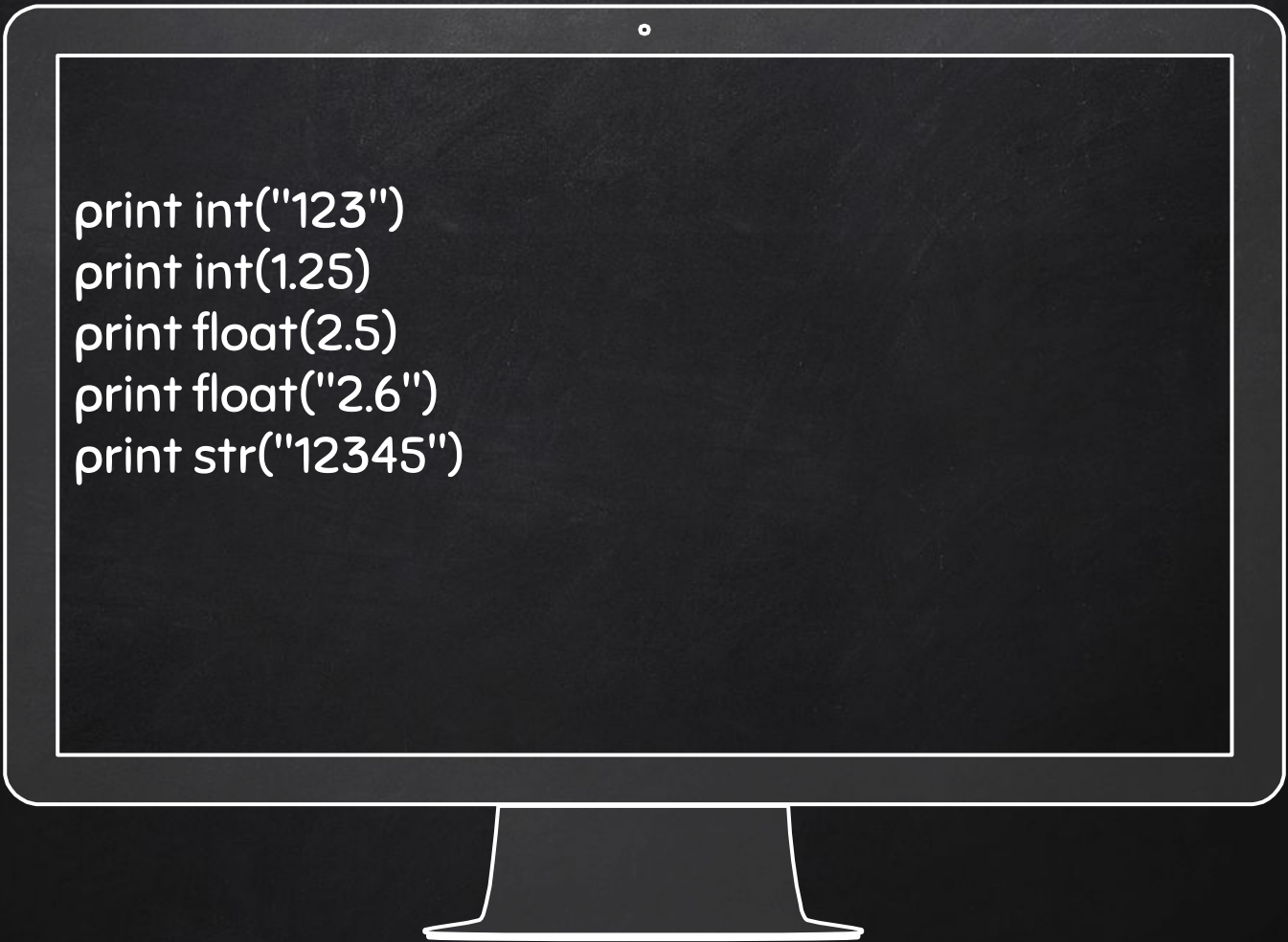
```
TrumpBecomsePresident=False
HillaryBecomesPresident=False
if TrumpBecomsePresident:
    print "The republicans win."
else:
    if HillaryBecomesPresident:
        print "The Democrats win."
    else:
        print "Some independent wins."
```

Return Statement

Terminates the execution of a function
before it reaches the end

```
def print_square_root(x):  
    if x <= 0:  
        print "Invalid input."  
        return  
  
    result = x**0.5  
    print "The square root of", x, "is", result  
print_square_root(-8)
```

Type Conversion

A white line-art illustration of a computer monitor with a stand. The monitor's screen is a black rectangle containing five lines of white Python code. The code demonstrates type conversion: converting a string to an integer, a float to an integer, a float to a float, a string to a float, and a string to a string.

```
print int("123")  
print int(1.25)  
print float(2.5)  
print float("2.6")  
print str("12345")
```

Function that Returns

```
def absolute_value(x):  
    if x < 0:  
        return -x  
    else:  
        return x  
print absolute_value(-5)
```

<http://www.pythontutor.com/index.html>

For Loop

Syntax

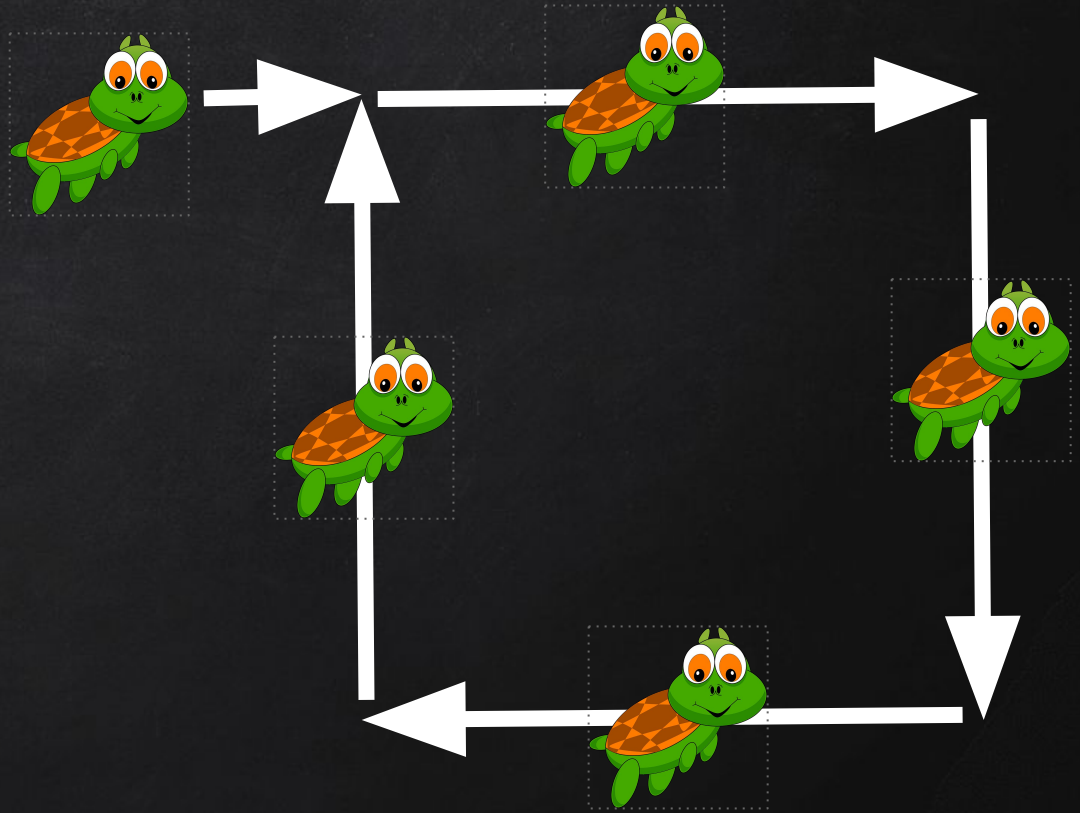
```
for iterating_var in sequence:  
    _____STATEMENTS
```




PYTHON TURTLE

```
import turtle
johnny = turtle.Turtle()
for i in range(0,4):
    johnny.forward(100)
    johnny.right(90)
```

<https://trinket.io/python>



```
import turtle
johnny = turtle.Turtle()
for i in range(0,4):
    johnny.forward(100)
    johnny.right(90)
```

```
import turtle
def run_in_square(x,y):
    johnny = turtle.Turtle()
    johnny.hideturtle()
    johnny.setpos(x, y)
    print johnny.position()
    for i in range(0,4):
        johnny.forward(10)
        johnny.right(90)
run_in_square(10,10)
run_in_square(20,20)
for i in range(1,5):
    run_in_square(i*10,i*10)
```

```
import turtle
def draw_polygon(sides, length):
    johnny = turtle.Turtle()
    for i in range(0,sides):
        johnny.forward(length)
        johnny.right(360/sides)
```

```
draw_polygon(4,20)
```

```
draw_polygon(6,20)
```

```
import turtle
def draw_spiral(angle, length_start, length_increase, sides):
    for i in range(0,sides):
        johnny.forward(length_start+(i*length_increase))
        johnny.right(angle)

johnny = turtle.Turtle()
draw_spiral(30, 10, 2, 20)
```

PLAY WITH PYTHON
LABS ON YOUR OWN!



THANKS!

Any questions?

You can find me at
beiwang@sci.utah.edu

<http://www.sci.utah.edu/~beiwang/teaching/cs1060.html>

CREDITS

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by [SlidesCarnival](#)
- Photographs by [Unsplash](#)