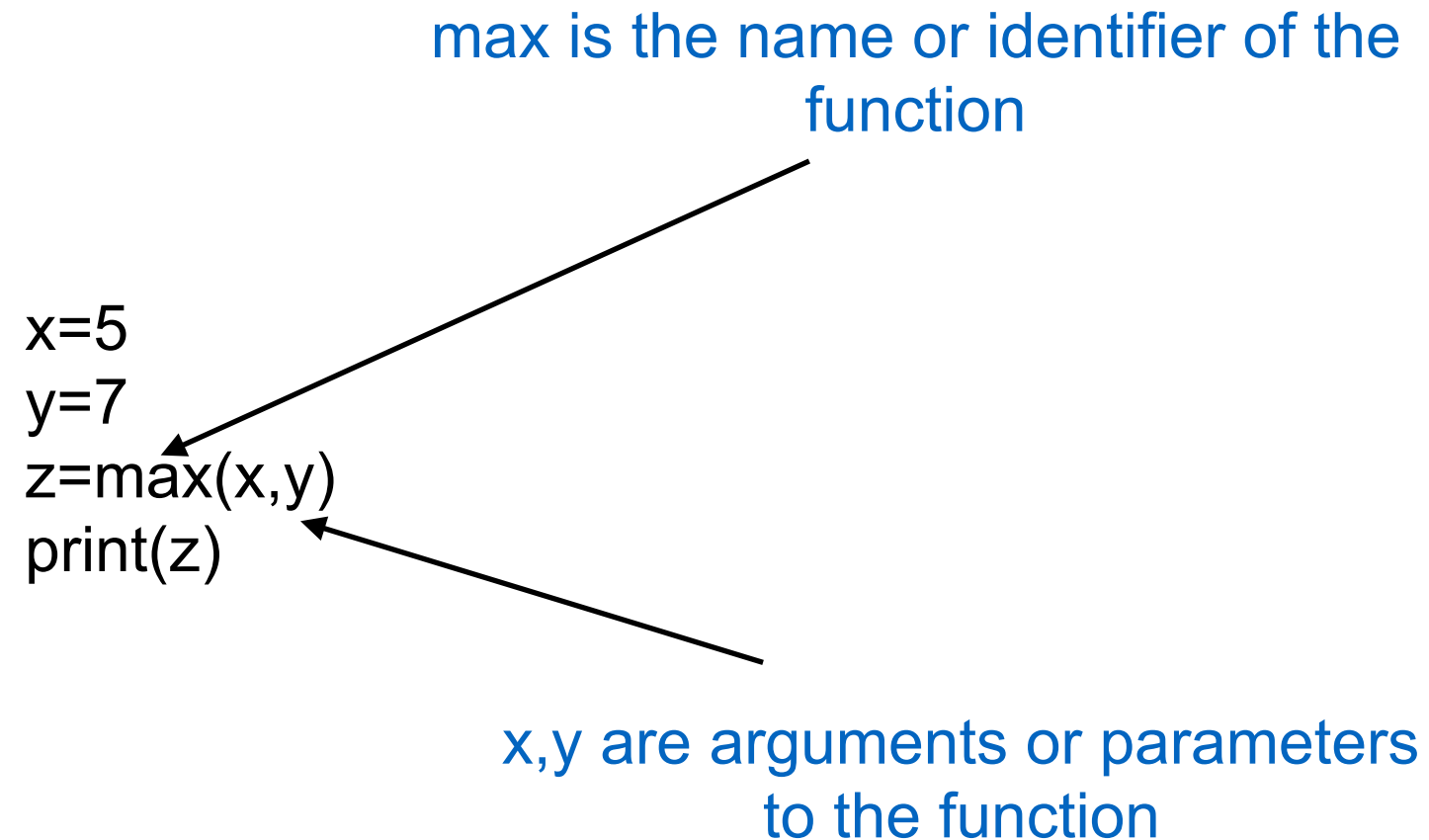


# Functions

# Calling a function



max is a black box. we don't know how python is figuring out which one is the greater of the two (and we don't want to know!)


# Function libraries

Functions can be grouped in libraries

Libraries need to be imported into a program

```
import math  
x=74  
math.sqrt(x)
```

math is a library. the  
program sets up a  
pointer to math

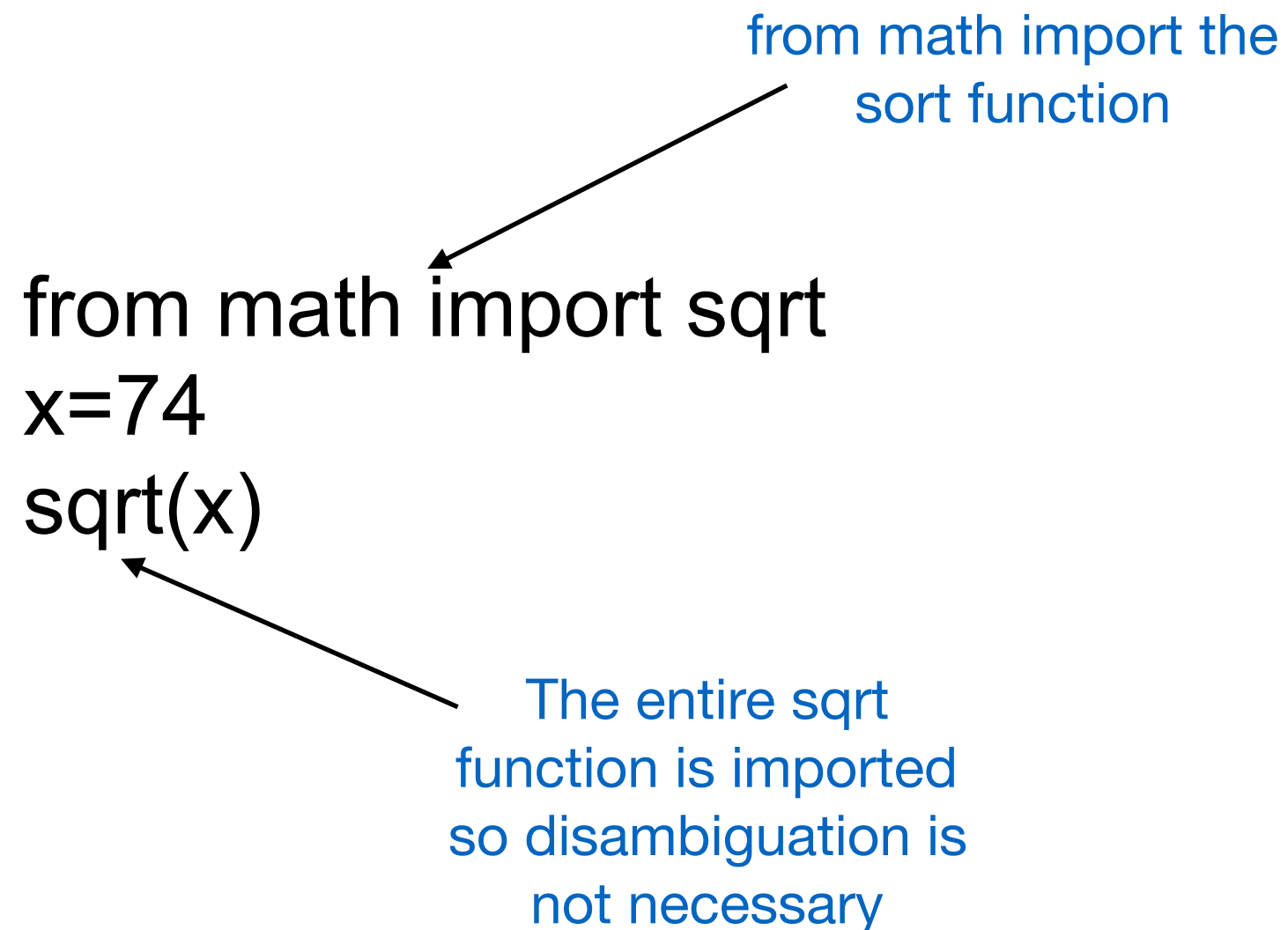


sqrt is a function in  
the math library and  
it needs to be  
disambiguated

# Function libraries

Functions can be grouped in libraries

Libraries need to be imported into a program



The diagram illustrates the process of importing a function from a library. It features three lines of code: `from math import sqrt`, `x=74`, and `sqrt(x)`. An arrow points from the explanatory text "from math import the sort function" to the `import` statement. Another arrow points from the text "The entire sqrt function is imported so disambiguation is not necessary" to the `sqrt(x)` function call.

```
from math import sqrt  
x=74  
sqrt(x)
```

from math import the  
sort function

The entire sqrt  
function is imported  
so disambiguation is  
not necessary

# Function libraries

Python is an open source language

With many libraries

Most need to be explicitly installed on your computer

Authenticated libraries are available at  
<https://pypi.python.org/pypi>

# Install libraries using pip

pip: python installer  
program

easygui: a gui  
development library

```
In [21]: !pip install easygui
```

```
Collecting easygui
```

```
  Downloading easygui-0.97.4-py2.py3-none-any.whl (78kB)
```

```
    100% |████████████████████████████████████████| 81kB 389kB/s
```

```
[?25hInstalling collected packages: easygui
```

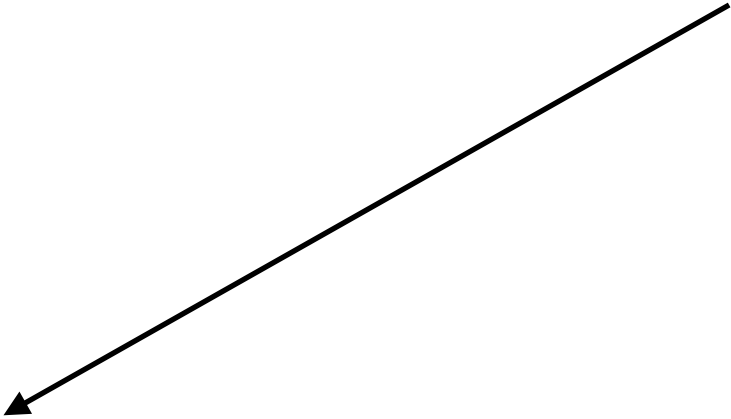
```
Successfully installed easygui-0.97.4
```

```
You are using pip version 7.1.2, however version 8.0.2 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.
```

pip is an independent program and can be run directly from windows powershell or mac's terminal. Anaconda ipython notebook is the hassle free way of installing libraries

## And then import them into your program

easygui is a library.  
the program sets up  
a pointer to easygui  
but will use the name  
eg instead



```
import easygui as eg  
eg.msgbox('To be or not to be', 'What Hamlet elocuted')
```



msgbox is a function in the easygui  
library and  
it needs to be disambiguated using  
whatever name our program gave to  
the library

# Defining your own functions

def is a keyword. it tells python that we're defining a function

```
def compute_return(price_then, price_now):  
    investment_return = (price_now - price_then) / price_then * 100  
    return investment_return
```

return is a keyword. it tells python what the function should return

investment\_return is a variable. you can use any expression here that evaluates to a value



# Returning values from a function

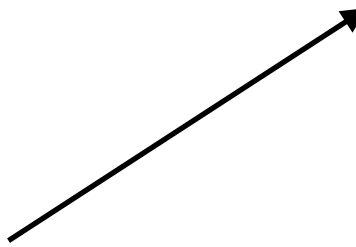
A function returns a value through the return statement. If there is no return statement, python uses **None**

```
def spam(x):  
    x=x+1  
  
print(spam(5)) --> None
```

# Returning multiple values from a function

```
def minmax(x,y):  
    return min(x,y),max(x,y)
```

```
x,y = minmax(7,2)  
print(x,y) --> 2,7
```



multiple assignment. x will take the value of the first item on the RHS and y the second. The RHS items must be separated by commas

# Passing arguments to a function

arguments are assigned values from left to right

```
def div(x,y):  
    return x/y
```

```
a=30  
print(div(a,10)) —> x is 30, y is 10, prints 3
```

```
def div(x,y):  
    return x/y
```

```
x=10  
y=30  
print(div(y,x)) —> x is 30, y is 10, prints 3
```

# Passing arguments to a function

You can give values to arguments directly in a function call

```
def div(x,y):  
    return x/y
```

```
print(div(x=30,y=10)) --> 3
```

```
print(div(y=10,x=30)) --> 3
```

# Functions can have default arguments

```
def compute_return(x,y,z=0):  
    investment_return=(y-x)/x  
    if z and z==100:  
        investment_return * 100  
    return investment_return
```

0 is the default for z



r1 = compute\_return(1.2,91.2)

z is 0



r1 = compute\_return(1.2,91.2,100)

z is 100

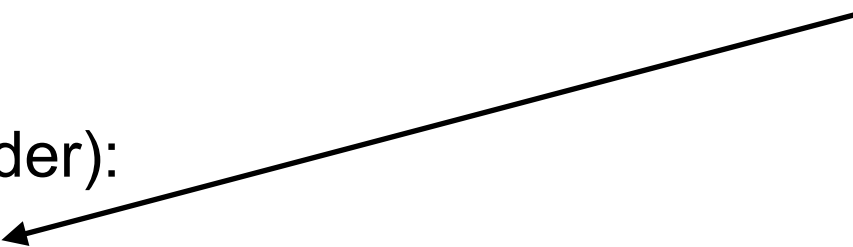


# Functions can have functions as arguments

```
def order_by(a,b,order):  
    return order(a,b)
```

```
order_by(4,7,max)
```

since we're using order  
like a function, it must  
be a function



pass the function max  
to order\_by



# Collections

# Lists: Sequential ordered mutable collections

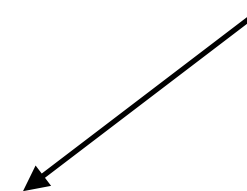
## Key properties

- \* collection of related objects
- \* ordered or sequential collection
- \* mutable. Lists can be modified

## Examples

```
list_of_names = ["John", "Jack", "Jill", "Joan"]  
list_of_tickers = ["AAPL", "IONS", "GE", "DB"]  
list_of_natural_numbers = [1, 2, 3, 4, 5, 6, 7]  
long_list = [1, ['a', ['b', 'c']], 43, "Too many cooks spoil the broth"]
```

objects in a list don't  
have to be of the same  
type





# Operations on lists

```
long_list = [1,['a',['b','c']],43,"Too many cooks spoil the broth"]
long_list.append('Many hands make light work') #adds an item to the back of the list
long_list[3] #Gets the 4th item in the list
long_list[1][1][0] #Accessing nested items
long_list.extend(['e','f']) #appends contents of a list
long_list.remove(1) #Removes the item with the VALUE 1
long_list.pop() #Removes and returns the last item
long_list.pop(1) #Removes and returns the ith item
len(long_list) #Returns the length of the list
```

# Lists are mutable

Contents of a list can be changed

Examples

`x = [1,2,3,4]`

`x[0]=8`  $\longrightarrow$  `[8,2,3,4]`

# Mutable vs immutable

**immutable:** data objects that cannot be changed  
e.g. the number 5 is immutable (we can't make it into an 8!)

**mutable:** data objects that can be changed  
e.g., a list of objects owned by Jack and Jill  
['pail', 'water']  
(it can be changed to ['pail'])

int, str, bool, float are immutable

list objects are mutable

**every python object is either mutable or immutable**

# Mutable vs immutable

Try this?

```
x = [1,2,3]
y = x
x[2] = 4
print(x)
print(y)
```

# Mutable vs immutable

And this

```
y=['a','b']  
x = [1,y,3]  
y[1] = 4  
print(x)  
print(y)
```

# Mutable vs immutable

## What's the difference?

```
def eggs(item,total=0):  
    total+=item  
    return total  
print(eggs(1))  
print(eggs(2))
```

```
def spam(elem,some_list=[]):  
    some_list.append(elem)  
    return some_list  
print(spam(1))  
print(spam(2))
```

## Tuples: sequential, immutable collections

```
price = ("20150904",545.23)
price[0] —> "20140904"
price[1] —-> 545.23
price[1]=26.3 —-> TypeError
price[2] —-> IndexError
```

Tuples are just like lists except they are not mutable  
(cannot be changed)

All list operations, except for the ones that change  
the value of a list, are also valid tuple operations

# Iteration



# iterating using location indices

**range**: a **sequence** of integers from 0 to length of prices

**len**: the number of items in prices

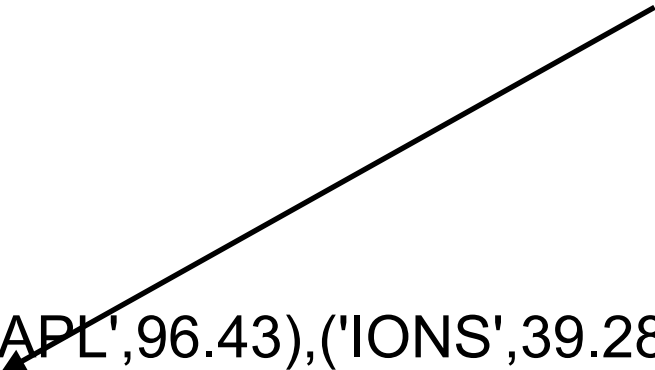
```
for index in range(len(prices)):
    print(prices[index][0], prices[index][1])
```

**index**: a variable name that holds each value of the sequence in turn. One iteration - one value!

# iterating by accessing items sequentially

**stock\_price:** a variable that will map to each element in the list sequentially

```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]  
for stock_price in prices:  
    print(stock_price[0],stock_price[1])
```



# controlling iteration: break and else

```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]  
ticker = input('Please enter a ticker: ')
```

```
for item in prices:  
    if item[0] == ticker:  
        print(ticker,item[1])  
        break  
else:  
    print("Sorry",ticker,"was not found in my database")  
print("Statement after for")
```

the for block

**else:** the program will do this only if the for does not encounter a 'break'

**break:** the loop will end and control will pass outside the for loop

## practice problem

Write a function `search_list` that searches a list of tuple pairs and returns the value associated with the first element of the pair

```
prices = [('AAPL',96.43),('IONS',39.28),('GS',159.53)]
```

```
x=search_list(prices,'AAPL')
```

```
#The value of x should be 96.43
```

```
x=search_list(prices,'GOOG')
```

```
#The value of x should be None
```

```
inventory = [('widgets',100),('spam',30),('eggs',200)]
```

```
y=search_list(inventory,'spam')
```

```
#The value of y should be 30
```

```
y=search_list(prices,'hay')
```

```
#The value of y should be None
```

# Dictionaries: key-value pairs

```
mktcaps = {'AAPL':538.7,'GOOG':68.7,'IONS':4.6}
mktcaps['AAPL'] #key-based retrieval
print(mktcaps['AAPL'])
mktcaps['GE'] #error (no "GE")
'GE' in mktcaps
mktcaps.keys() #returns a list of keys
sorted(mktcaps.keys()) #returns a sorted list of keys
```

# Sets: unordered collections of unique objects

```
tickers={"AAPL","GE","NFLX","IONS"}
regions={"North East","South","West coast","Mid-
West"}
"AAPL" in tickers #membership test
"IBM" not in tickers #non-membership test
pharma_tickers={"IONS","IMCL"}
tickers.isdisjoint(pharma_tickers) #empty intersection
pharma_tickers <= tickers #subset test
pharma_tickers < tickers #proper-subset test
tickers > pharma_tickers #superset
tickers & pharma_tickers #intersection
tickers | pharma_tickers #union
tickers - pharma_tickers #set difference
```

# Home assignment 1

Write a function **word\_distribution** that takes a string as an input and returns a dictionary containing the frequency of each word in the text. For example if the argument to the function is:

```
text_string = "Hello. How are you? Please say hello if you don't love me!"  
print(word_distribution(text_string))
```

 should print

```
{'hello': 2, 'how':1, 'are':1, 'you':2,'please':1, "don't": 1 ...}
```

Make sure that you exclude all punctuation from your count and that your count ignores case. Thus, Hello and hello are the same word in the above example. The . ? and ! are all ignored. But the apostrophe in don't doesn't count and don't is treated as a single word.