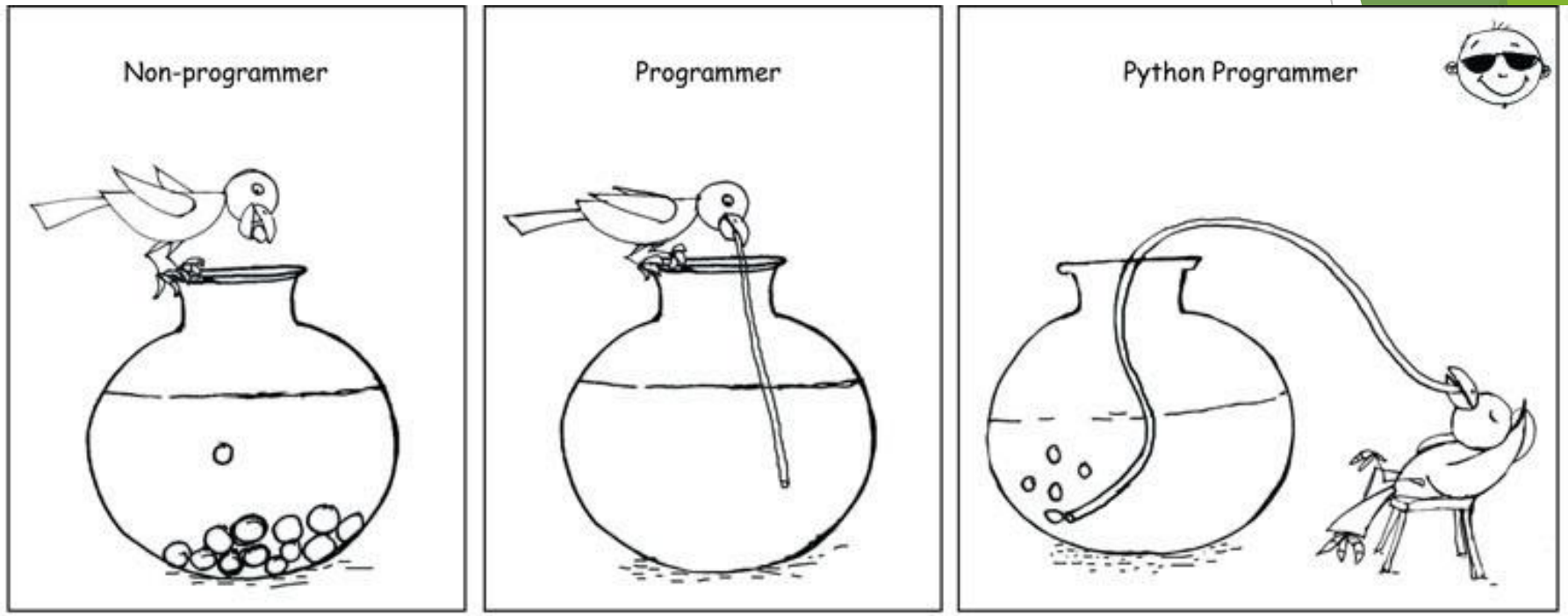
Let's play with Python

Ramesh S

What is Python?

Python is a high level programming language


Non-programmer
Programmer
Python Programmer

What is Python?

Python is a general purpose programming language

What is Python?

Python is
fun
powerful
fast

What is Python?

Python has a huge standard library

What is Python?

Python is interpreted

What developers do during compilation?

What is Python?

Python is dynamically typed

What is Python?

Python
is
object oriented

What is Python?

Python is portable

RaspberryPi

What is Python?

Python is open source

Who created Python?

Guido van Rossum

Benevolent Dictator for Life (BDFL)
- Python Software Foundation.

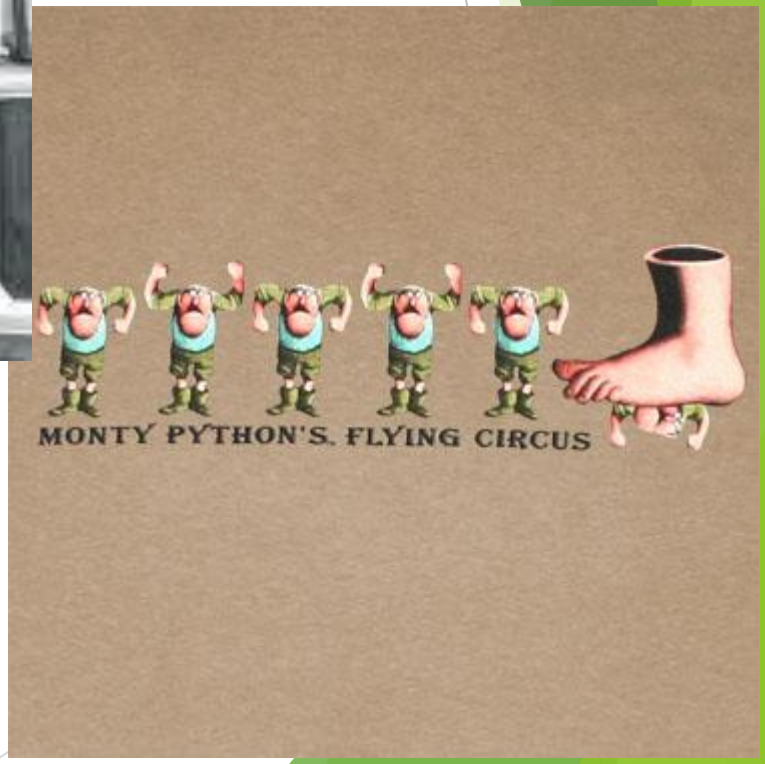
When was python released?

1991

Who is using python?

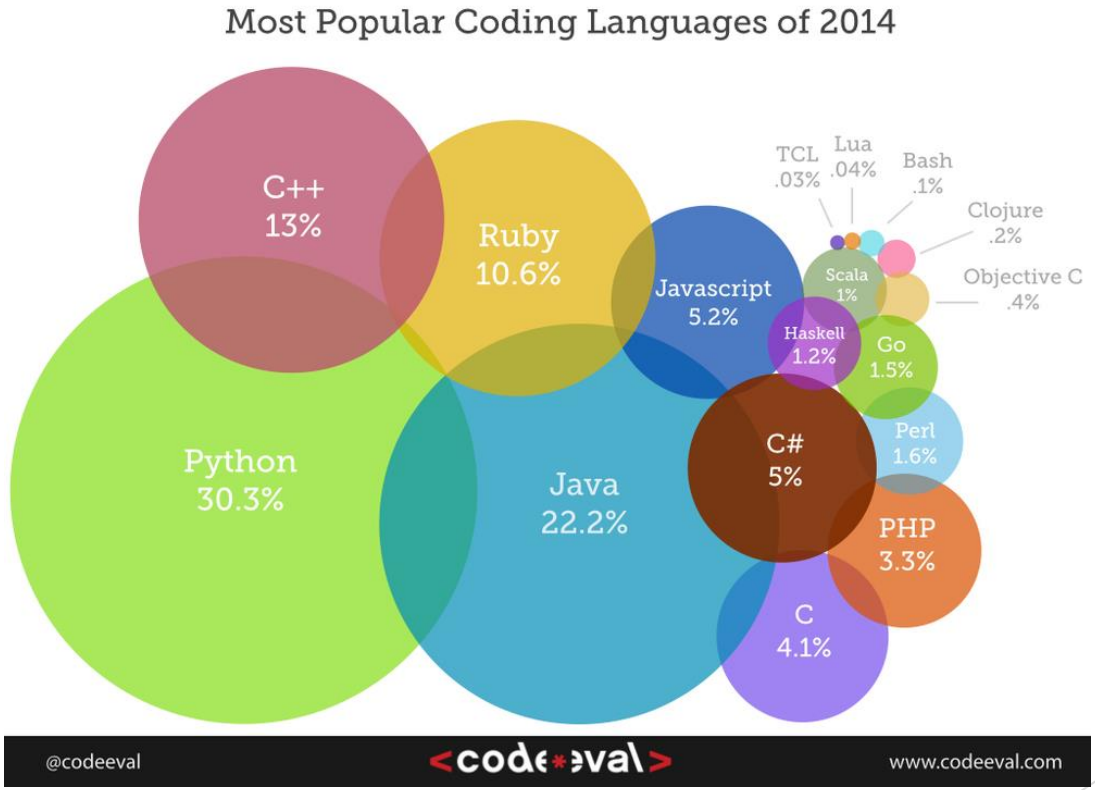
NASA
Google
Microsoft
eBay
PayPal
Dropbox
OpenStack

Why is it called python?

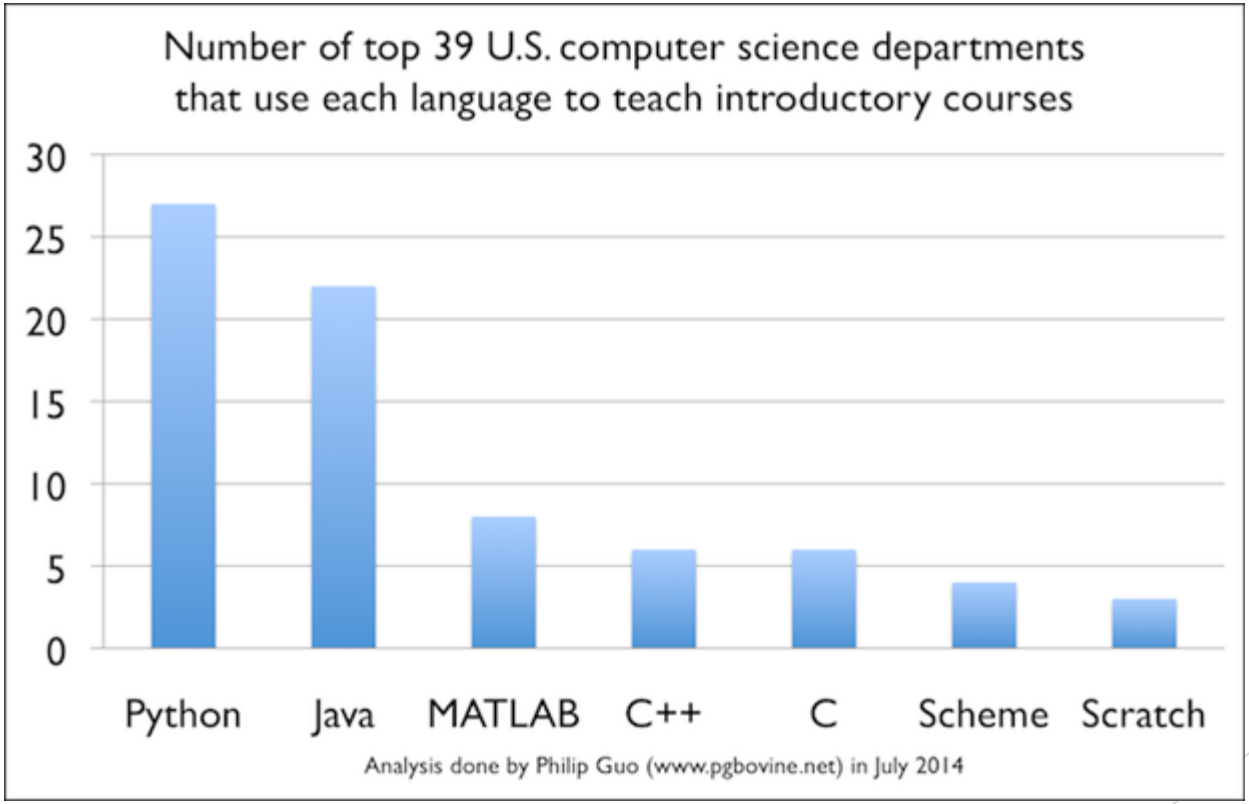


Monty Python
Flying Circus

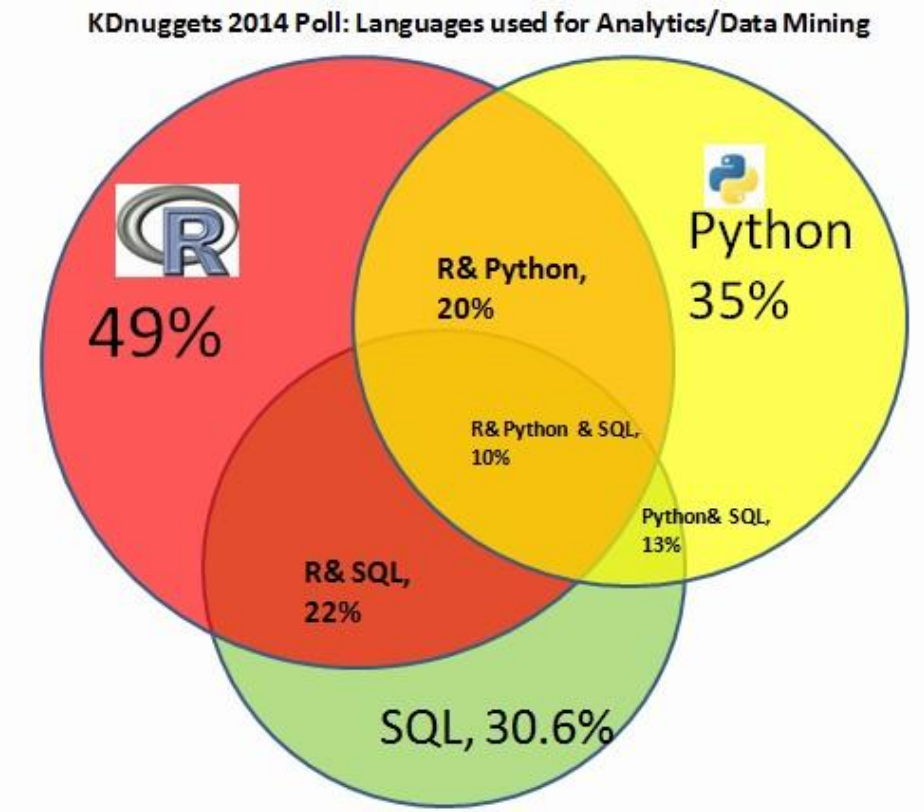
How popular is Python?



How popular is Python?



How popular is Python?



Where is Python used

- Scripting
- Rapid Prototyping
- Text Processing
- Web applications
- GUI programs
- Game Development
- Database Applications
- System Administration Automation
- Scientific Computing
- Machine Learning
- BigData and Data Analysis

Which version????

2.x vs 3.x

Learning Python is Easy - Check these out

What does this program do?

```
print "Hello World"
```

What does this program do?

```
count=0
while count<11:
    print count
    count+=1
```

What does this program do?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
for kid in kids:
    print kid
```

What does this program do?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
for kid in kids:
    if len(kid)==6:
      print kid
```

What does this program do?

```
kids=['Lahari','Lalitha','Jithu','Vishal','Ujwal','Nitish']
for kid in kids:
    if "it" in kid:
      print kid
```

What does this program do?

```
marks=[96,87,67,81,81]

print(len(marks))
print(sum(marks))
print(min(marks))
print(max(marks))

print(sum(marks)/len(marks))
```

What does this program do?

```
marks={'maths':97,'science':'98','history':78}
for subject,marks in marks.items():
    print subject,marks
```

What does this program do?

```
f=open('dictionary.txt','r')
for line in f:
    if line.startswith('s'):
      print line
```

What does this program do?

```
for x in range(5):
    print(x)
```

What does this program do?

```
for x in range(5,10):
    print(x)
```

What does this program do?

```
for x in range(5,100,5):
    print(x)
```

What does this program do?

```
f=open('words.txt','r')
j=open('new-words.txt','w')
j.write(f.read())
f.close()
j.close()
```

What does this program do?

```
def factorial(n):
    fact=1
    for x in range(n,1,-1):
      fact*=x
    return fact

print factorial(4)
```

What does this program do?

```
squares=[]
for x in range(1,100):
    squares.append(x*x)

print squares
```

Playing with strings

```
print( "Hello World")
name=raw_input("Please enter your name:")
print( "Hello", name)
print(len(name))
print(type(name))
print(name[0])
print(name[-1])
print(name[0:3])
print(name[3:])
print(name[:4])
```

Playing with strings

```
print name[0:10:2]
print name[::1]
print name[::-1]
print name[:]
print name.lower()
print name.upper()
print name.capitalize()
print name.swapcase()
```

Playing with strings

```
var='c'
var1="India"
_var="""This is
a multi line
string."""
print type(var)
print type(var1)
print type(_var)
```

Playing with strings

```
a='hello'
print a + a
print a + “hi”

print a * 5
```

Comments

Use # for single line comments

Use """ for multiline comments

```
#this is a single line comment
"""this is a
multiline
comment
"""
```

Playing with numbers

```
a,b,c=10,3,3.0
print a/b
print a/c
print a//c
print a%b
print a**b
print abs(b-a)
```

Playing with booleans

```
a,b=True,False
print a
print type(b)
print a and b
print a or b
print not b
print a and not b
print (4>6) and (7<3)
print (3==3.0) and (7<10)
```

Playing with lists

List is like an array in C, but far more flexible.

```
a=[1,5,8,3,5,9,7]
print type(a)
print a[0]
print a[-1]
print len(a)
print sum(a)
print min(a)
print max(a)
print a[2:5]
print a[::2]
```

Playing with lists

```
a=[1,5,8,3,5,9,7]
a.append(9)
print a
a.insert(1,200)
print a
print a.index(5)
print a.count(5)
a.sort()
print a
a.reverse()
print a
```

Playing with lists

```
print a
a.remove(9)
print a
c=a.pop()
print c
print a
a.extend([7,8,9])
print a
```

List Exercises

- create a list of strings
- sort it
- print the last 4 strings

- create a list of numbers(integers and floats)
- remove the smallest one
- find out if 9 is there in the list

Playing with tuples

A tuple is a read only list.

```
a=(1,5,8,3,5,9,7)
print type(a)
print a[0]
print a[-1]
print len(a)
print sum(a)
print min(a)
print max(a)
print a[2:5]
print a[::2]
```

Playing with tuples

```
b=3,4,5
print type(b)
print b.count(4)
print b.index(3)
print b[2]
```

```
b[2]=45
b.append(3)
b.sort()
```

Playing with dictionaries

```
d={'H':'Hydrogen','O':'Oxygen'}
print type(d)
print d.keys()
print d.values()
print d.items()
print d['H']
print d.get('H')
d['C']='Carbon'
print d
del d['H']
print d
```

Printing

- print("Hello World")
- print("Hello", "India")
- print("{} is a {}".format('Water','Liquid'))
- print "Avg height of Indian men is %f" % 1.62
- print "Avg height of Indian men is %d" % 1.62
- print "Avg height of Indian men is %s" % 1.62

Data Types

- a=None
- b=True
- c=45
- d=56.3
- e="hello"
- f=[]
- g=3,4,5
- h={"a":"apple","b":"Bat"}

type casting

a=3

print(float(a))

type cast a str into list

type cast a dict into list

type cast a tuple into list

type cast a list into set

type cast a list into dict

Advanced assignments

```
>>> a=b=c=2
>>> print a,b,c
>>> a,b,c=2,3,4
>>> print a,b,c
>>> a=[4,5,6]
>>> x,y,z=a
>>> print x,y,z
>>> x,y=a
>>> w,x,y,z=a
```

Swapping

```
>>> a=10
>>> b=5
>>> a,b=b,a
>>> print a
>>> print b
```

Conditional - if

```
aura = 2
if aura < 2.5:
    print "you are not healthy"
```

Conditional - if - else

```
aura = 2
if aura <= 1:
    print( "You're dead!" )
else:
    print( "You're alive!" )
```

Conditional -if-elif-else

```
aura = 2
if aura <= 1:
    print "You're dead!"
elif aura > 3:
    print "You're spiritual!"
else:
    print "You're alive!"
```

Looping-for

```
weapons = ["Arrow", "Mace", "Spear", "Sword"]
for x in weapons:
    print(x)

for weapon in weapons:
    print(weapon)
    print len(weapon)
```

Looping with range

```
for x in range(5):
    print(x)

for x in range(5,15):
    print(x)

for x in range(0,20,3):
    print(x)
```

Looping with while

```
a=0
while a<10:
    print(a)
    a=a+1

# a++ and a-- are not valid
# you may use a+=1 or a-=1
```

Exercise

Create a list of squares of all odd numbers below 50 and print the list.

Exercise

```
a=[]
for x in range(1,50,2):
   a.append(x*x)

print a
```

Simple function

```
def hello():
    print("I am a simple function")
hello()
```

Function with arguments

```
def add(x,y):
    return x+y

c=add(4,5)
print(c)
print(add(5,6))
```

Function assignment

Write a function **diff** which takes two parameters and returns their difference.

ex:

diff(5,2) should return 3

diff(2,5) should return 3

Do not use abs() inside your function

```
def diff(a,b):
    if a>b:
      return a-b
    else:
      return b-a
```

```
def diff(a,b):
      if b > a:
          a,b=b,a
      return a-b
```

Function with arguments with default values

```
def add(x,y=10):
    return x+y

c=add(4,5)
d=add(5)
print(c,d)
print(add(5,6),add(8))
```

Keyword-arguments

```
def wish(name,age):
    print("Hello {} you are {} years old".format(name,age))

wish('India',67)
wish(67,'India')

wish(age=67,name='India')

#When calling a function using keyword arguments the
#order of arguments does not matter.
```

global

```
age=16
def grow():
    print age
    age=age+1
    print age
grow()
print age
```

global

```
age=16
def grow():
    global age
    print age
    age=age+1
    print age
grow()
print age
```

Function that returns multiple values

```
#python functions can return any number of
#values.

def sumdiff(a,b):
    return a+b,abs(a-b)
print type(sumdiff(4,9))
mysum,mydiff=sumdiff(4,9)
print(mysum,mydiff)
```

Handling variable length arguments

```
def average(*num):
    print(type(num))
    print(num)
    print(float(sum(num))/len(num))
average(3,4)
average(3,4,8)
average(3,4,8,90,4.5,5.3,7.8)

#*args will pack all the arguments into a tuple
#called args
```

Handling variable length arguments

```
def average(a,b,*num):
    print(type(num))
    print(num)
    print((a+b+sum(num))/(2+len(num)))
average(3,4)
average(3,4,8)
average(3,4,8,90,4.5,5.3,7.8)
average()
#*args will pack all the arguments into a tuple
#called args
```

Variable length keyword-arguments

```
def polygon(**kwds):
    print type(kwds)
    print kwds
polygon(width=10,length=20)
polygon(width=10,length=20,height=5)
polygon(width=10,length=20,height=5,units='cm')


#**kwds will pack all the arguments into a dict
#called kwds
```

Handling any type of arguments

```
def polygon(a,b,c,*sides,**options):
    print(type(options))
    print(type(sides))
    print(a,b,c)

polygon(8,7,6,4,2,8,units='cm',compute='area')


#**kwds will pack all the arguments into a dict
#called kwds
```

lambda functions

```
add=lambda x,y:x+y

print(add(4,5))

#these are anonymous functions
#they work are inline functions
```

Let's revisit functions

```
def parrot(voltage, state='a stiff', action='voom', type='Norwegian Blue'):
    print "-- This parrot wouldn't", action,
    print "if you put", voltage, "volts through it."
    print "-- Lovely plumage, the", type
    print "-- It's", state, "!"
```

Which call is correct?

- parrot(1000)
- correct
- parrot()
- wrong - required argument missing
- parrot(action = 'VOOOOOM', voltage = 1000000)
- correct
- parrot('a thousand', state = 'pushing up the daisies')
- correct
- parrot(actor='John Cleese')
- wrong - unknown keyword
- parrot('a million', 'bereft of life', 'jump')
- correct
- parrot(110, voltage=220)
- wrong - duplicate value for argument
- parrot(voltage=5.0, 'dead')
- wrong - non-keyword argument following keyword

File I/O - common scenarios

Read file contents at once f=open('input.txt','r') print(f.read()) f.close()	**Read all lines into a list** f=open('input.txt','r') lines= f.readlines() print(lines) f.close()
Read file char by char f=open('input.txt','r') print(f.read(1)) print(f.read(1)) f.close()	**Write into a new file** f=open('input.txt','w') f.write("this is line one") f.close()
Read file line by line f=open('input.txt','r') print(f.readline()) print(f.readline()) f.close()	**Append to an existing file** f=open('input.txt','a') f.write("\nthis is a new line") f.close()

Modules

- Generic import
- Universal import
- Function import
- Function import with rename
- default name space
- dir
- help

Sample Module

mymodule.py

```
name='python'

def add(a,b,c):
    return a+b+c

def sub(a,b):
    return a-b
```

Sample Module

mymodule.py

```
name='python'

def add(a,b,c):
    return a+b+c

def sub(a,b):
    return a-b
```

myprogram.py

```
from mymodule import *

print name

print add(2,3,4)

print sub(7,3)
```

Sample Module

mymodule.py

```
name='python'

def add(a,b,c):
    return a+b+c

def sub(a,b):
    return a-b
print __name__
```

myprogram.py

```
from mymodule import *

print name

print add(2,3,4)

print sub(7,3)
```

Sample Module

mymodule.py

```
name='python'

def add(a,b,c):
    return a+b+c

def sub(a,b):
    return a-b
print __name__
```

myprogram.py

```
from mymodule import *

print name

print add(2,3,4)

print sub(7,3)
print __name__
```

Sample Module

mymodule.py

```
name='python'
def add(a,b,c):
    return a+b+c
def sub(a,b):
    return a-b
```

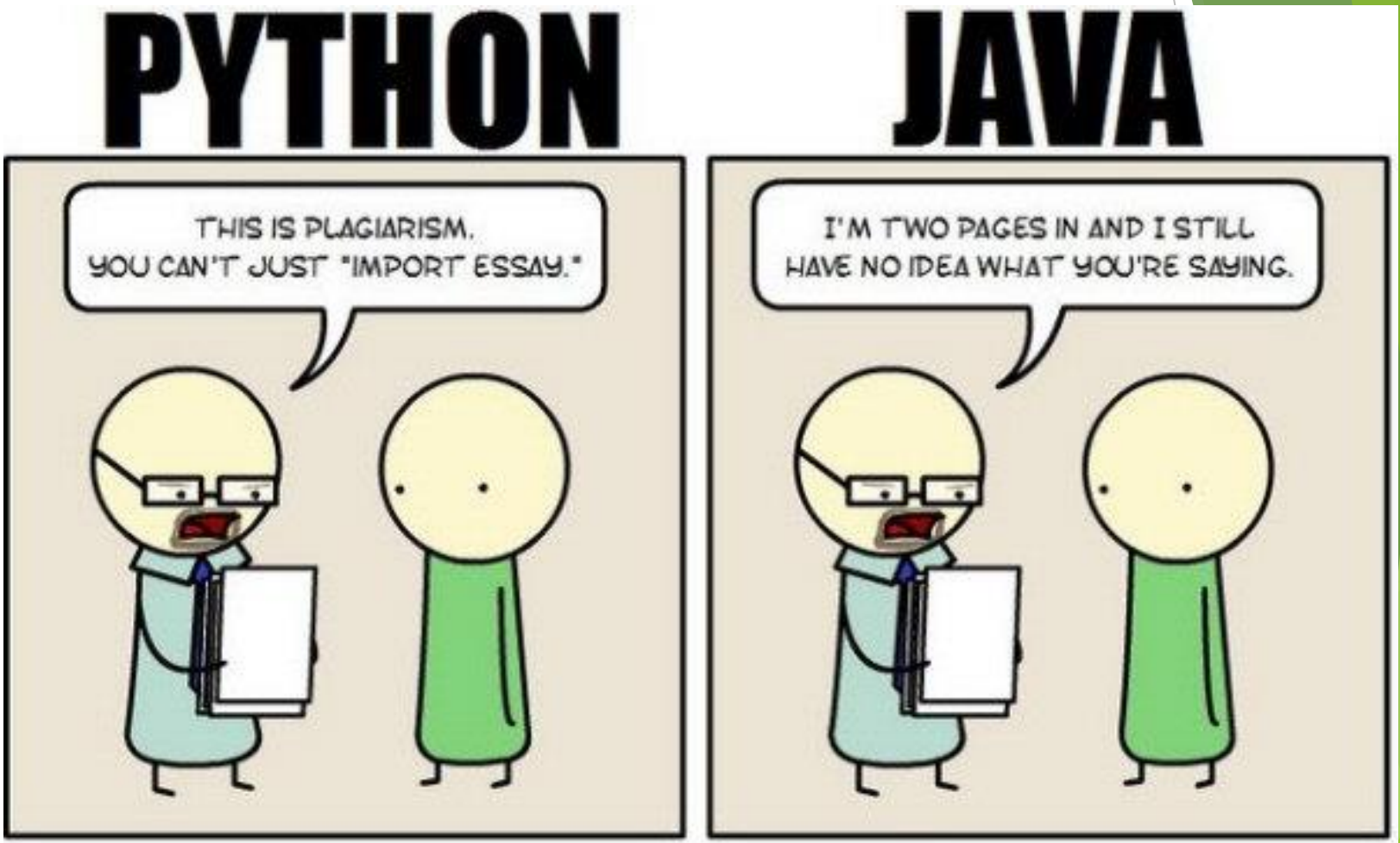
myprogram.py

```
from mymodule import *

def main():
    print name
    print add(2,3,4)
    print sub(7,3)

if __name__=='__main__':
    main()
```

Writing an essay




© Neale Haynes


Name.....
Class.....Div.....
Sub.....
School.....