what is python

interpeted (programs are executed line by line instead of being compiled and then run)

slower as a consequence : compiler can make passes over the code to optimize it

dynamic typing (type-checking done at run-time rather than at compile time)

type-checking : checking for expressions in which types are incorrectly used

procedural + functional + object-oriented

object-oriented : can define classes, instantiate objects

procedural : do this, then this, then this

functional : define functions, express the result of your program as an output of those functions

what's so great about python

easy to use, learn

good support network

good documentation

very readable

more technically:

very powerful built-in functions and methods

very powerful built-in data structures

support for lambda functions

type coercion

e.g. quicksort in python

Python vs C++

not compiled

dynamic typing (with type coercion)

typically slower

very few lines of code required

'batteries loaded' - most important functions/libraries preinstalled

centralized package system (PyPy)

REPL (read-eval-print loop) provided by IDLE

use of whitespace to denote block structure rather than braces

def quicksort(ls):

pivot = ls[0]

less = [x for x in ls if x < pivot]

more = [x for x in ls if x > pivot]

equal = [x for x in ls if x == pivot]

return less + equal + more

installing python

python.org/download

linux : pre-installed

simplest possible program

print "Hello world!"

no need to load any libraries or define any 'main' function!

what a python program looks like

-stored in a .py file

-one statement on one line

-indentation is used to express block structure

how to use IDLE

type in a statement or two, hit Enter

quick and easy way to test out small pieces of code

very useful when you're learning python to begin with!

comments

--------

single-line comments : #

Multi-line comments : '''<comment (which may include newlines)>'''

variables

---------

a = 1

asdf = True

b, c = 'abcdefghijklmnopqrstuvwxyz', -4.5

every variable has its own type

types in Python: bool, int, str, float

when in doubt, use the type() function

>>> type('i am a string!')

<type 'str'>

>>> type(-345)

<type 'int'>

>>> type(5.6)

<type 'float'>

and so on

can convert between types in certain cases!

>>> int('678')

678

>>> float(633)

633.0

representing numbers in python

negative numbers: -2, -3

floating point numbers: 5.7 -44.2

large numbers: python adds an 'L'

>>> 42365376484

42365376484L

can convert between bin, hex, oct, dec

to use a binary number as part of an expression: 0b01101 (precede it with a 0b)

for hex: 0x, for oct: 0o

>>> 0010

8 (python returns the value in dec)

basic operations with numbers

-----------------------------

x, y = 5, 6 #multiple assignment

x + y

x - y

x \* y

x / y # in Python 2, truncates the answer

x \*\* y # x raised to the power of y

conditionals

------------

if 3 > 5:

print "hello"

else:

print "world"

x = ['some\_string']

if len(x) == 0:

print "x is empty"

else:

print "x has %d %s" %(len(x), 'elements' if len(x) > 1 else 'element')

else statement not compulsory

if not []: # the empty list evaluates to false

print 3

functions

---------

def function\_name(function\_arguments):

<some code>

return <some expression>

no need to mention the 'types' of formal arguments or return type of the function

returning a value is optional e.g.

def say\_hello(name):

print "hello %s" %(name)

we can also add a conditional to the return statement

makes our code looks neater

def even\_or\_odd(number):

return "even" if number%2 == 0 else "odd"

lists

-----

a = []

a.append(3)

a.append(True)

a += ['hello', -4, 1.66, [3,44]]

len(my\_list)

a[0]

a[3]

a[-1]

a[-2]

a[-1][1]

slicing

a[:2]

a[2:]

a[:-1]

a[:-3]

a[2:5]

a[2:-2]

a[:40]

a[:]

a.insert(0, 'good')

a.insert(-1, 'bye')

list comprehensions

a = range(7)

b = [x\*\*2 for x in a]

b = [x-3 for x in a[:-3]]

b = [a[i] for i in range(len(a))]

reversing a list

list(reversed(a)) # does not modify a

a.reverse() # modifies a

sorting a list

from random import shuffle

a = range(10)

shuffle(a)

sorted(a) # returns a new list, does not modify a

a.sort() # changes a

we can multiply lists by ints

a = [0]\*7

but they are all references to the same object

a = [[1,0]]\*3

a[0][1] = 2 # see what happens!

Q. Write a function that takes a list as input and returns the minimum element of the list. Try to do it in different ways

A.

def find\_min(ls):

return min(ls)

def find\_min(ls):

return sorted(ls)[0]

min = ls[0]

for i in range(len(a)):

if a[i] < min:

min = a[i]

min = ls[0]

for i in ls:

min = i if i < min else min

tuples

------

same as lists

one exception: single element typles

a = (1) # assigns 1 to a

a = (1,) # assigns a one-element typle

a = (1,2,3,4,5)

a[2] = -1 # gives an error

a.append(True) # gives an error

a.sort() # no attribute

a.reversed() # no attribute

so, tuples are immutable! useful when building dicts

tuple(reversed(a))

tuple(sorted(a))

sorted(a)

can convert lists to tuples and vice versa

tuple([3,4,5])

strings

-------

a = 'this is a string'

a = "this is also a string"

a = "now we're using a single quotation mark inside a string, so we're using double quotes"

a = 'we can use "double quotation marks" inside single quotation marks'

a = 'or we can use escape characters like this: he\'s'

a = "this string has a newline character\n"

a = 'to use a backlash in a string we have to escape it twice: \\'

a = r'or we can use raw strings by putti\ng an r before the quotatio\n mark'

a = ' bcd '

a.strip() # does not change a

a = 'this is a sentence that i want to break into words'

a.split() # does not change a

we can access strings and slice them like lists

a = "hello world"

a[3:8]

a[-7]

strings are immutable

the join function:

string.join(<iterable>)

concatenates the elements of the iterable with the separator as 'string'

Q. Given a sentence, write a function to find the number of words it has

A.

def num\_words(sentence):

return len(sentence.split())

we can split according to different characters

a = 'this,is,a,comma,separated,sentence"

a.split(',')

a = '''this is a multi-line

string. it can span as many lines as

i want it to. it also includes newline

characters'''

loops

-----

for i in range(0,4):

print i

for i in range(0,4):

print i,

i = 0

while i < 10:

print i

i += 1

for i in range(5,19,4):

print i

for x in [3,4,5]:

pass

b = [x==2 for x in [3,4,5]]

Q. Write a function to zip two lists together

A.

def zip1(ls1, ls2):

i, result = 0, []

x = min([len(ls1), len(ls2)])

while i < x:

result.append((ls1[i], ls2[i]))

i += 1

return result

dictionaries

------------

lists are useful for storing ordered data

but what if we want to store a list of mappings?

say we want to store a list of courses for each student in IIT

we don't care about the ordering of the students

we can use a dict

a = {}

a['freshie'] = ['ma105', 'ph105', 'cs101', 'hs101', 'me119']

a['rohan'] = ['cs208', 'cs210', 'cs218', 'cs420', 'ma214', 'sc202']

and so on

>>> a

{'rohan': ['cs208', 'cs210', 'cs218', 'cs420', 'ma214', 'sc202'], 'freshie': ['ma105', 'ph105', 'cs101', 'hs101', 'me119']}

>>> a.keys()

['rohan', 'freshie']

The mapping can be from any type to any other type

However there are restrictions

We cannot map lists to other items using a dict

Since lists are mutable (type error: unhashable type: list)

But tuples aren't, so we can use tuples in dicts

dict comprehensions

a = {x:10-x for x in range(10)}

classes

-------

class Student:

def \_\_init\_\_(self, year, cpi, dept):

self.year = year

self.cpi = cpi

self.dept

input

-----

a = raw\_input()

print "Hello", a

files

-----

with open('path/to/file') as f:

file\_contents = f.readlines()

more functions: f.readline(), f.read(), f.write()

<code dealing with file\_contents)

a simple example of file handling: time\_scraper.py

older method (no need to use in python 2.7):

f = open('path/to/file')

try:

for line in f:

<some code>

finally:

f.close()

libraries

---------

Python's standard library is huge