PYTHON COMMANDS

*Import function from module:*

from os import getcwd

where\_am\_i = getcwd() (invoke the function)

*Import module:*

import os

os.getcwd()

‘/Users/jdalemans/python/headfirstpython’

os.environ

os.getenv

os.genenv(‘HOME’)

import sys

sys.platform

print(sys.version)

import datetime

>>> datetime.date.today().day

30

>>> datetime.date.today().month

10

>>> datetime.date.today().year

2017

>>> datetime.date.isoformat(datetime.date.today()

‘2017-10-30’

import time

>>> time.strftime(“%H:%M”)

’16:25’

>>> time.strftime(“%A %p”)

‘Monday PM”

>>> time.sleep(5)

import random

dir(random)

help(random)

help(random.randint)

random.randint(5,2)

*Blocks of code (suite) are always indented in Python:*

if minute in odds:

print(“This is odd”)

else:

print(“Not an odd minute”)

*A COLON INTRODUCES AN INDENTED SUITE OF CODE:*

if minute in odds:

print(“This is odd”)

elif minute > 10:

print(“Not an odd minute > 10”)

else:

print(“Not an odd minute <= 10”)

*For loop:*

>>> for I in [1, 2, 3]:

print(i)

1

2

3

>>> for ch in “Hi!”:

print(ch)

H

I

!

>>> for num in range(3):

print(‘Hello’)

Hello

Helo

Hello

Start from the end 99 down to 1 (0 not included, with step -1:

for I in range(99, 0, -1):

list(range(5))

list(range(5, 10) (start and stop value)

list(range(10, 0, -2 (start,stop and step value)

*Data structure built-in functions (BIFs):*

*List (ordered mutable list):*

>>> l = list()

>>> l

[] (an empty list)

>>> l = [ 1, 2, 3 ] (square brackets)

>>> l

[1, 2, 3]

>>> s.remove(3) (remove object 3)

[1, 2]

>>>

>>> s.pop() (pop the last item from the list)

2

>>> s

1

>>> s.extend([4, 5, 6])

[1, 4, 5, 6]

>>>s.append(8)

>>>s

[1, 4, 5, 6, 8]

>>> s.insert(4, 2) (insert 2 before 4)

[1, 2, 4, 5, 6]

Turn list ‘plist’ back into a string:

string\_new = ‘’.join(plist)

third = second (this references second to the second list – no copy of list)

third = second.copy() (copy the second list to the third list)

first = letters(0) (get the first object)

last = letters(-1) (get the last object)

List slices of a list:

letters[2:10:3] (start with 2 and end with 10 using step 3)

*Tuple (ordered unmutable list):*

>>> t = tuple()

>>> t

()

>>> t = (1, 2, 3) (parenthesis)

>>> t

(1, 2, 3)

Turn a string into a list of letters:

>>> book = “Okido”

>>> booklist = list(book)

>>> booklist

[‘O’, ‘k’, ‘i’, ‘d’, ‘o’]

>>> booklist[0:2]

[‘O’, ‘k’]

>>> ‘’.join(booklist[1:3])

‘kid’

>>> ‘’join(booklist[-5:1])

‘Okido’

>>> backwards = booklist[::-1]

>>> ‘’.koin(backwards)

‘odikO’

*Dictionary (unordered set of key/value pairs):*

>>> d = dict()

>>> d

{} (an empty dictionary)

>>> d = { ‘first’: 1, ‘second’: 2, ‘;third’: 3}

>>> d

{‘second’: 2, ‘third’: 3, ‘first’: 1}

*Set (unordered set of unique objects):*

>>> s= set()

>>> s

set() (an empty set)

>>> s = {1, 2, 3}

>>> s

{1, 2, 3}

*Read input from user:*

word = input(‘Provide a word: ‘)

*Create function:*

def search4letters(phrase, letters)

*Call function semantics:*

By-value argument passing (if value changes in function there is no effect for the variable in the code that called the function):

By-reference argument passing (if value changes in function there is effect for the variable in the code that called the function):

-> by-value is used if a new memory address is created (new object created, e.g. a = a \* 2)

-> by-reference is used if an existing address is changed (e.g. append to list)

*Add docstring (documentation only):*

def search4letters(phrase, letters)

\*\*\* Return any ‘letters’ in ‘phrase’ \*\*\*

*Add annotation (documentation only see pep 3107):*

def search4letters(phrase:str, letters:str) ->set:

\*\*\* Return any ‘letters’ in ‘phrase’ \*\*\*

*Specify default argument:*

def search4letters(phrase:str, letters:str=’aeiou’) -> set:

*Positional assignment:*

search4letters(‘galaxy’,’xyz’)

def search4letters(phrase:str, letters:str) ->set:

*Keyword assignment:*

Search4letters(letters=’xyz, phrase=’galaxy’)

def search4letters(phrase:str, letters:str) ->set:

*Sharing reusable code of functions in modules:*

Create 2 functions and save the file as vsearch.py:

def search4vowels(phrase:str) -> set:

""" Display any vowels found in supplied word."""

vowels = set('aeiou')

return vowels.intersection(set(phrase))

def search4letters(phrase:str, letters:str='aeiou') -> set:

""" Retrun a set of the 'letters' founbd in 'phrase'."""

return set(letters).intersection(set(phrase))

In another Python program, the module can be imported:

import vsearch

Order of search folders:

1. current directory
2. interpretors site-packages locations – third-party or own Python modules  
     
   cd /Users/jdalemans/python/headfirstpython/mymodules  
     
   echo “can be empty” > README.txt  
     
   Create setup.py:  
   from setuptools import setup

setup(

name='vsearch',  
 version='1.0',

description='Head First Python Search Tools',

author='JD'

author\_email='jaap.dalemans@gmail.com',

url='headfirstlabs.com',

py\_modules=['vsearch'],

)  
  
 python3 setup.py sdist

cd dist  
 python3 –m pip install vsearch-1.0.tar.gz

1. standard library locations

*Check PEP8 compliance:*

python3 –m pip install pytest

python3 –m pip install pytest-pep8

py.test –pep8 vsearch.py

for k in sorted(found):

print(k, 'was found', found[k], 'time(s).')

*You can avoid a KeyError by ensuring every key in your dictionary has a value associated with it before you try to access it. Although the in and not in operators can help here, the established technique is to use the setdefault method instead:*

found.setdefault(letter, 0)

found[letter] += 1