Omgamganapathayenamaha

Omgamganapathayenamaha

Omgamganapathayenamaha

Omgamganapathayenamaha

Good Points learned during Prep

1. Lists
   1. There is no ‘discard’ method in list. So, we have to use ‘remove’ method
   2. If the element we are trying to remove is not in the list, ‘remove’ method will throw ‘ValueError’. So, use ‘count’ method before applying ‘remove’ method
   3. There is no ‘find’ method in list. We have to use ‘index’ method
   4. ‘discard’ method is present in sets (don’t throw exception when the element we are trying to remove from the set is not present)
   5. ‘find’ method is present in string (don’t throw exception when the substring we are trying to find is not present. Just returns -1 in that case)
2. Rotate list: l.insert(0, l.pop())
3. List objects are not iterators by default (they are just iterables). Use iter(list\_object) to make it an iterator. Test it using next(list\_object)
4. Ways of creating an m x n matrix
5. m = 3
6. n = 4
7. print([[0 for \_ in range(n)] for \_ in range(m)])
8. print([[0] \* n for i in range(m)])
9. [[0] \* n] \* m
10. Lists can be passed to a function and we can change them in the function so that they will reflect in the original passed list
11. Tuples and Strings are immutable. They cannot be changed inside/outside function. We can only create a new tuple or a new string
12. Dict objects are not iterators by default (they are just iterables). Use iter(dict\_object) to make it an iterator. Test it using next(dict\_object)
13. Dicts can be passed to a function and we can change them in the function so that they will reflect in the original passed dict
14. Dict: setdefault, get, \_\_getitem\_\_, \_\_setitem\_\_, \_\_delitem\_\_ are few good methods (get method won’t give KeyError even though the key is not present in the Dictionary. It just returns None)
    1. Setdefault internally uses get method
15. collections.deque: appendleft, popleft, extendleft are the extra methods compared to list
16. discard method is present in the Sets class (it won’t throw KeyError even though the element we are trying to remove is not present in the set. ‘remove’ method throws KeyError Exception in this case)
17. Coming up with ‘Cartesian Product’ in Python

# Cartesian product

import itertools

print(list(itertools.product([1,2,3],[3,4,5])))

1. Binary Strings of length ‘n’ with Cartesian Product idea

# Binary Strings of length 'n'

n = 3

print(list(itertools.product(\*[[0,1]]\*n)))

1. K-ary Strings of length ‘n’ with Cartesian Product idea

# K-ary strings of length 'n'

k = ['a', 'b']

print(list(itertools.product(\*[k] \* n)))

1. Generator objects and Iterators are Memory Friendly
2. Decorator functions take function as input and always return function as output
3. Decorator will take parameters

Thanks and Regards,

Sastry

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**From:** Sastry K V S R <[kvsr@juniper.net](mailto:kvsr@juniper.net)>  
**Sent:** Monday, January 28, 2019 1:52 PM  
**To:** Sastry K V S R <[kvsr@juniper.net](mailto:kvsr@juniper.net)>; sivaram sastry <[sivaram.kavuri@gmail.com](mailto:sivaram.kavuri@gmail.com)>  
**Cc:** Sastry K V S R <[kvsr@juniper.net](mailto:kvsr@juniper.net)>  
**Subject:** Python Video Points

Omgamganapathayenamaha

Omgamganapathayenamaha

Omgamganapathayenamaha

Omgamganapathayenamaha

1. Guido Van Rossum in 1991 came up with Python
2. Difference between interpreter and compiler
3. Is Python “purely interpreted language”?
4. Installing Anaconda on Linux
5. <https://docs.python.org/2/tutorial/floatingpoint.html>
6. **PEP 8**. **PEP 8** is Python's style guide. It's a set of rules for how to format your Python code to maximize its readability. ... PEP is actually an acronym that stands for Python Enhancement Proposal
7. Dynamic Typing
8. 7 / 4 gives 1.75
9. 7 // 4 gives 1
10. String reverse = mystring[::-1]
11. String.format()
    1. print('Ganesh is the {pr} {pu}'.format(pu='pujya', pr='pratham'))
    2. print('The result was {n:1.2f}'.format(n=num))
12. f-strings (works from python3.6)
    1. name = ‘ganesh’ age = 37
    2. print(f’{name} is {age} years old’)
    3. print(f'num is {age/10}')
13. lists
14. dictionaries
15. sets
16. tuples
17. bool (True & False)
18. Files
    1. Try ‘r+’ and ‘w+’ options
       1. You can open a file in ‘r+’ only when that file exists
       2. File handle will be at the 0th position as soon as you open the file in ‘r+’
       3. You can read and write as you wish by seeking the filehandle
       4. When you open the file in ‘w+’, existing contents will be vanished if the file already exists. If the file is not present already, the file gets created
       5. File handle will be at the 0th position as soon as you open the file in ‘w+’
       6. You can read and write as you wish by seeking the filehandle
    2. Can we open a new file with ‘mode=a’? YES, we can
    3. f.read(), f.readlines(), f.seek()
19. min, max, in, enumerate, zip, range, input, sum(tuple)
20. random library
    1. from random import shuffle
    2. from random import randint
21. List Comprehensions
    1. mylist = [0,1,2,3,4,5,6,7,8]
    2. result = [ x for x in mylist if x % 2 == 0 ]
    3. result = [ x if x % 2 == 0 else ‘ODD’ for x in mylist ]
    4. result = [ x\*y for x in [2,4,6] for y in [1,10,1000] ]
22. join method works only when all the list elements are ‘strings’ (Good One)
23. Namespaces
    1. Local
    2. Enclosing Function
    3. Global
    4. Built-in
24. Variable Scope aka Variables’ visibility across the code
25. import string followed by string.ascii\_lowercase, string.ascii\_uppercase, string.ascii\_letters
26. map, filter, lambda
27. from functools import reduce eg., reduce(lambda x,y: x\*y, numbers, 1)
28. import operator eg., reduce(operator.mul, numbers, 1)
29. Pangrams are words or sentences containing every letter of the alphabet at least once
    1. Example: The quick brown fox jumps over the lazy dog
30. Object Oriented Programming
    1. Class
    2. Instance
    3. Class Attributes
    4. Object Attributes
    5. Methods
    6. \_\_init\_\_
    7. Inheritance
    8. Polymorphism
       1. len(list), len(str)
       2. print(list), print(dict)
    9. Special / Magic / Dunder Methods
       1. \_\_init\_\_ (gets called whenever we create an instance)
       2. \_\_str\_\_ (gets called whenever we call print() or str() on that class instance)
       3. \_\_len\_\_ (gets called whenever we call len() on that class instance)
       4. \_\_del\_\_ (gets called whenever we call del instance eg., del b)
    10. Abstract Base Class - Abstract Base Class is a Class where we would not like to instantiate objects out it. Helps in passing functionality and characteristics to derived classes
    11. Abstract Methods - Declared only to mandate defining them in derived classes
    12. Both Abstract Base Classes and Abstract Methods help in implementing ‘Polymorphism’
        1. Polymorphism – Same interface works in a different way when dealing with different objects (len(list), len(str), print(list), print(dict))
        2. Polymorphism – Same operator works in a different way when dealing with different objects (2 + 3, ‘ganesh’ + ‘kvsr’)
        3. ‘raise NotImplementedError’ to be used in Abstract Methods
31. PyPI contains third party packages for Python (It is similar to CPAN for Perl)
32. coloroma – package to play with colored text
    1. from colorama import init
    2. init()
    3. from colorama import Fore
    4. print(Fore.GREEN + “some text in green”)
33. Playing with Excel – openpyxl
34. Modules
35. Packages (Collection of modules in a hierarchical manner)
    1. \_\_init\_\_.py
36. If \_\_name\_\_ == “\_\_main\_\_”
37. Syntax Errors and Exceptions (run time errors and we can handle them using try .. except .. else .. finally)
38. Few Builtin Exception Classes
    1. SyntaxError
    2. ZeroDivisionError
    3. TypeError
    4. ValueError
    5. KeyError
39. pylint <script\_name> (grades your script based on the styling guidelines mentioned in PEP 8)
40. import unittest
    1. Inherit your Test Class from unittest.TestCase
    2. Use a Method from unittest.TestCase i.e., assertEqual to test the equality
    3. Run all your tests using ‘unittest.main()’
41. Given the below code, how to call myfunc by creating an object of class B?

Class A():

Class B():

def myfunc():

print(“ganesh”)

Solution 1: bobj = A.B()

bobj.myfunc()

Solution 2: obj = A()

bobj = obj.B()

bobj.myfunc()

1. Decorators
   1. Decorator functions take the originial function as a parameter, add some extra functionality and executes it
2. Python based Web Frameworks: Django and Flask
   1. Reddit is developed using Python
3. Generators use ‘yield’ statement to generate sequences over time. They are memoryefficient. For example,
   1. Generating a list of million numbers and keeping the list in memory for iterating over it later is not memory efficient
   2. Instead, generating a number one at a time remembering the previous value generated is memory efficient which Generator does
4. By default, we cannot apply next() on strings. If applied, they will fail saying ‘str’ object is not an iterator
   1. We have to apply iter(string) to make it iterable
5. Just like ‘list comprehensions’, we have ‘generator comprehensions’
   1. my\_list = [1,2,3,4,5]
   2. gencomp = (item for item in my\_list if item > 3)
   3. print(type(gencomp))
   4. print(next(gencomp))
6. from collections import Counter
   1. l = [1,1,1,2,2,4,4,4,3,3]
   2. Counter(l)
   3. s = ‘ddddggggggqwerrrtttt’
   4. Counter(s)
   5. sentence = ‘how many times how many times how how’
   6. c = Counter(sentence.split())
   7. c.most\_common(3)
   8. c.clear()
7. from collections import defaultdict
8. defaultdict: Never gives a KeyError even though the key is not present in the dictionary. If key is not present in the dictionary while accessing, it gives the ‘default value’ specified (zero in the below case)

<< OLE Object: Picture (Device Independent Bitmap) >>

1. from collections import OrderedDict
   1. OrderedDict retains the order in the which the elements are added while normal dictionary doesn’t
   2. While checking for equality of Dictionaries, OrderedDicts check will return False even though all keys and values are same if the order differs
2. from collections import namedtuple
3. Dog = namedtuple(‘Dog’, ‘name breed age’)
4. d = Dog(name=’sam’, breed=’Lab’, age = 3)
5. d.name
6. deque
7. Dealing with Time
   1. import datetime
   2. t = datetime.time(5,25,1,999999)
   3. t.hour t.second t.microsecond t.minute
   4. datetime.time.min
   5. datetime.time.max
   6. datetime.time.resolution - 1microsecond
8. Dealing with dates
   1. import datetime
   2. datetime.date.today()
   3. d = datetime.date(2018,1,25)
   4. d.year d.month d.day
   5. datetime.date.min
   6. datetime.date.max
   7. datetime.date.resolution - 1day
   8. d1 – d2
9. Python Debugger
   1. import pdb
   2. Insert the statement ‘pdb.set\_trace()’ before the code where it is failing so that the interactive mode starts
10. Timing your code
    1. import timeit
    2. timeit.timeit('"-".join([str(x) for x in range(100)])', number=10000)
11. Regular Expressions
    1. import re
    2. re.split(‘@+’, s) re.split takes regular expressions as spliiter where as string.split won’t take regular expressions as splitters
    3. re.search(pattern, text) returns Match Object if successful, returns None in case of failure
    4. match\_obj.start() match\_obj.end() match\_obj.span()
    5. re.findall(‘[\w\W]’, s)
12. Converting String to a File Handle and using it as a file

import io

st = 'Ganesh is the pratham pujya'

fh = io.StringIO(st)

fh.read()

fh.write('\nLalithamba is the Supreme Power')

fh.seek(0)

for line in fh:

print(line, end='')

io.StringIO.close(fh)

1. Use io.BytesIO for converting bytes t o a filehandle
2. hex, bin, abs, round
3. pow(a,b) equal to a\*\*b
4. pow(a,b,c) equal to (a\*\*b)%c
5. import math
   1. math.sqrt()
   2. math.pi
6. String Methods
   1. s.upper() s.lower() s.capitalize() s.title() s.count() s.find() s.join(list)
   2. ‘hello world’.center(20, ‘z’) ‘hello\thi’.expandtabs()
   3. s.islower() s.isupper() s.isalpha() s.isalnum() s.isspace() s.istitle()
   4. 'HELLO'.endswith('LLO') 'HELLO'.startswith('HE')
   5. s.split(‘e’) 'HELLO'.partition('LL')
7. Sets
   1. s = set()
   2. s.add(2)
   3. s.discard(2)
   4. s.clear()
   5. s1.union(s2) s1.update(s2)
   6. s1.intersection(s2) s1.intersection\_update(s2)
   7. s1.difference(s2) s1.difference\_update(s2)
   8. s1.symmetric\_difference(s2) s1.symmetric\_difference\_update(s2)
   9. s1.isdisjoint(s2)
   10. s1.issubset(s2) s1.issuperset(s2)
   11. sc = s.copy()
8. Dictionaries
   1. d = dict()
   2. d.items() d.keys() d.values()
   3. Dictionary Comprehensions : {x:x\*\*2 for x in range(8)}
9. Lists
   1. l = list()
   2. l.append() l.extend() l.insert() l.pop() l.remove() del l[2] l.count() l.reverse() l.sort()

**Basic Practice:**

<http://codingbat.com/python>

**More Mathematical (and Harder) Practice:**

<https://projecteuler.net/archives>

**List of Practice Problems:**

<http://www.codeabbey.com/index/task_list>

**A SubReddit Devoted to Daily Practice Problems:**

<https://www.reddit.com/r/dailyprogrammer>

**A very tricky website with very few hints and touch problems (Not for beginners but still interesting)**

<http://www.pythonchallenge.com/>

omgamganapathayenamaha

omgamganapathayenamaha

omgamganapathayenamaha

omgamganapathayenamaha