LET'S PYTHON

PYTHON CODEBOOK

Your way to Competitive Programming.

AUTHOR - CODE OF GEEKS

CODE OF GEEKS

PYTHON

CODEBOOK

In this e-book, we will look at different Python Hacks. This e-book is useful for anyone who wants to brush up Python concepts and that too in very less time. This will prove to be a great reference if you want to start competitive programming with Python.

PYTHON CODEBOOK by CODE OF GEEKS

www.codeofgeeks.com

For permissions contact:

support@codeofgeeks.com

Taking inputs :

s = input() // taking string as input

n = int(input()) // taking int as input

b = bool(input()) // taking boolean value as input

l = list(input().split(‘,’)) // taking list as a input where elements are seperated by comma

s = tuple(input().split(‘,’)) // taking tuple as a input where elements are seperated by comma

Taking Multiple input in one line :

a,b = input().split(separator, maxsplit)

Taking a list of ‘n’ integers as input :

list(map(int,input().strip().split()))[:n]

Printing a formatted output :

1. Let us assume that we have to print values of two variables – a=10 b=20 as 10—-20, We can do this by : print(a,b,sep=’—‘)
2. The output displayed by the print() method can be formatted as you like. ‘%’ operator is used for this purpose. It joins a string with a variable or value. Example :
   print(“string” % (variable-list))

Useful Basic Methods – Python

1. Converting a number from octal, binary and hexadecimal system to decimal number system.

n1 = 0o17 # representation of octal numbers

n3 = 0x1c2 # representation of hexadecimal number

We can do this with the help of int() method like int(n1), int(n2), int(n3).

2. Converting a decimal integer to octal, binary and hexadecimal system.

bin() : For decimal to binary conversion.

oct() : For decimal to octal conversion.

hex() : For decimal to hexadecimal conversion.

3. Mathematical methods

ceil(x) : It raises x value to the next higher integer value. For example, ceil(4.5) gives 5.

floor(x) : It decreases x value to previous integer value. For example, floor(4.5) gives 4.

degrees(x) : It converts angle value x from radians to degrees.

radians(x) : It converts x value from degree to radians.

sin(x) : It gives a sine value of x.

cos(x) : It gives a cosine value of x.

tan(x) : It gives a tan value of x.

exp(x) : It gives exponential of x.

fabs(x) : It gives absolute value of x. Like fabs(-4.53) gives 4.53.

factorial(x) : It gives the factorial of x.

fmod(x,y) : It gives remainder of division of x & y. Like, fmod(13.5,3) gives 1.5.

fsum(val) : It gives the accurate sum of floating point values.

log10(x) : It gives base-10 logarithm of x.

sqrt(x) : It gives the square-root of number x.

pow(x,y) : It raises x value to the power y.

pow(x,y,z) : It raises x value to the power y mod z

gcd(x,y) : It is used to find the greatest common divisor of x & y.

trunc(x) : It returns real value of x is truncated to integer value. Like trunc(43.545) returns 43.

isnan(x) : It returns True if x is not a number.

eval(expression) : It returns evaluated arithmetic expression. Like, eval(3*7) gives 21.

Strings Tip & Methods – Python

Finding the length of String : len(string_name)

Indexing in Strings :

0	1	2	3	4	5	6
p	Y	t	h	o	n	n
-7	-6	-5	-4	-3	-2	-1

Reversing a String :

```
i=1
n=len(s)
while i<=n:
    print(s[-i],end=' ')
    i+=1
```

Slicing a String :

string-name[start : stop : stepsize]

If given string is “pythonn” so s[0:7:2] gives pton as output.

s[::2] : access entire string in steps of 2.

s[2::] : access string s[2] to ending.

Repeating a String :

```
s = ‘pythons’
print(s*2)
```

gives “pythonspythons” as output.

Concatenation of Strings :

Strings can be concatenated with one another using ‘+‘ operator.

```
s1 = "string 1"
s2 = "string 2"
s3 = s1 + s2
print(s3)
```

Output : string 1string 2

Removing Spaces from String :

We can remove extra spaces with the help of lstrip(), rstrip(), strip() methods.

```
s = "     Python      "

print(s.lstrip())
print(s.rstrip())
print(s.strip())
```

Output :

Python

Python

Python

String Methods :

1. s.find(substring, beginning, ending) : It is used to find the first occurrence of given substring in a string. It returns -1 if given substring is not available.

2. s.count(substring, beginning, ending) : It is used to find the total number of occurrences of a substring in a main string.

3. s.replace(old, new) : It is used to replace a substring with another substring.

4. sep.join(str) : When a group of strings are given, it is possible to join them all and make a single string. Syntax : seperator.join(string)

```
l = ["one","two","three"]
s = "-".join(l)
print(s)
```

Output : one-two-three

5. s.upper() : It converts to upper-case string.

6. s.lower() : It converts to lower-case string.

7. s.swapcase() : It converts all lowercase letters to uppercase letters and vice versa.

8. s.title() : It converts a string in such way that first letter of a word in string is a uppercase letter.

```
s = "pYthon"

print(s.upper()) // PYTHON

print(s.lower()) // python

print(s.swapcase()) // PyTHON
```

print(s.title()) // Python

9. s.startswith() : It is used to know whether a string is starting with a substring or not. Like,

s.startswith('P') is used check whether a substring starts with 'P'.

10. s.endswith() : It is used to know whether a string is ending with a substring or not. Like,

s.endswith('P') is used check whether a substring ends with 'P'.

11. s.alnum() : It returns True if all characters in the string are alpha

numeric (A-Z, a-z, 0-9).

12. s.alpha() : It returns True if string has atleast one character and all other characters are alphabetic (A-Z, a-z).

13. s.isdigit() : It returns True, if the string contains only numeric digits (0-9).

14. s.islower() : It returns True, if at least one or more letters are in lowercase.

15. isupper() : It returns True, if at least one or more letters are in uppercase.

Formatting the Strings :

Formatting a string means presenting the string in a clearly understandable manner. The format() method is used to format strings.

```
id = 10

name = "Code of Geeks"

sal = 1345.345

s = '{},{},{}'.format(id,name,sal)

s1 = '{}-{}-{}'.format(id,name,sal)
```

Output :

10,Code of Geeks,1345.345

Sorting the String :

We can sort the string alphabetically using sort() method and sorted() method.

Creating Lists using range() function :

We can use range() function to generate a sequence of integers which can be stored in a list. The format of range() function is :

range(start, stop, stepsize)

If not mentioned, start is specified to be 0 and stepsize is taken 1.

Above code will result in a list of 10 elements - 0 to 9.

```
l1 = range(10)
for i in l1:
   print(i)
```

Accessing list elements :

```
l=[1,2,3,4,5]
i=0
while i<len(l):
   print(l[i])
   i+=1
```

Output : 1 2 3 4 5

Concatenating two lists :

```
l1 = [1,2,3,4]
l2 = [5,6,7]
```

```
print(l1+l2)
```

Output : [1,2,3,4,5,6,7]

Repetition of Lists :

```
l = [10,20]
print(l*2)
```

Output : [10,20,10,20]

Membership in Lists :

```
l = [10,20,30,40,50]
a = 30
print(a in l)
```

Output :

True

List Methods

1. list.index(x) : It returns the first occurance of x in list.
2. list.append(x) : It appends the element x at the end of list.
3. list.insert(i,x) : It inserts x in the i-th position of list.
4. list.copy() : It copies all elements of a list to a new list and returns it.
5. list.extend(list1) : It appends list1 to list.
6. list.count(x) : It returns the total occurrences of x in list.
7. list.remove(x) : It removes element ‘x’ from the list.
8. list.pop() : It removes the ending element from the list.
9. list.sort() : It is used to sort the element of lists.

10. list.reverse() : It is used to reverse a list.

11. list.clear() : It is used to delete all the elements of a list.

12. max() : It is used to find the maximum element in a list.

13. min() : It is used to find the minimum element in a list.

```
l=[2,4,6,23]
print(max(l))
print(min(l))
```

Output :

23

2

2D Lists :

Suppose we want to create a 3X3 matrix, so we can represent as list – of – lists. For example,

mat = [[3,4,5],[4,6,2],[4,7,2]]

Creation :

```
for r in mat:
   for c in r :
      print(c,end=' ')
   print()
```

Tuple Creation :

```
tup = tuple(range(4,9,2))
print(tup)
```

Output :

4,6,8

Note !! Many lists methods can be applied to tuples as well.

Sets – Python

A Set is an unordered collection data type that is iterable, mutable, and has no duplicate elements.

Basic set operations & methods :

1. Set.add() : If we want to add a single element to an existing set, we can use the .add() operation.

It adds the element to the set and returns 'None'.

```
set1 = set('codeofgeeks')
set1.add('z')
print(set1)
```

Output :

{'c','d','e','f','g','k','o','s','z'}

2. Set.remove() :

This operation removes element from the set. If element does not exist, it raises a KeyError.

```
s = set([1,2,3,4,5])
s.remove(4)
print(s)
```

Output :

{1,2,3,5}

3. Set.pop() : This operation removes and return an arbitrary element from the set. If there are no elements to remove, it raises a KeyError.

```
s = set([2,3,4,5])
print(s.pop())
```

Output :

2

4. Set.difference() : It defines the difference between the number of elements in two sets.

Set is immutable to the .difference() operation (or the – operation).

```
s = set("code of geeks")
print(s.difference("geeks "))
```

Output :

{'f','c','o','d'}

5. Set.union() : Union of two given sets is the smallest set which contains all the elements of both the sets.

```
s = set("code of geeks")
print(s.union("geeks "))
```

Output :

{'g', 'o', 'd', 'f', 'c', 'k', 'e', 's', ' '}

6. Set.intersection() : It is the largest set which contains all the elements that are common to both the sets.

```
s = set("code of geeks")
print(s.intersection("geeks "))
```

Output :

{'s', 'e', 'g', ' ', 'k'}

Dictionaries :

Dictionary represents a group of elements arranged in the form of key-pair pair. The Key and its value are seperated by a colon(:).

dict = {'Name' : 'Vikas', 'Id' : 20}

Dictionary Methods :

1. dict.clear() : It removes all key-value pairs from dictionary.

2. dict.copy() : It copies content of one dictionary to another one.

3. dict.get() : It returns the value associated with key ' k '.

4. dict.items() : It returns an object that contains key-value pairs of dictionary.

5. dict.keys() : It returns a sequence of keys from the dictionary 'd'.

6. dict.values() : It returns a sequence of values from the dictionary 'd'.

7. dict.pop(k,v) : It removes the key 'k' and its value from 'd'.

8. dict.update(x) : It adds all elements from dictionary 'x' to 'd'.

9. zip() : converts two lists to a dictionary.

Some miscellaneous concepts

itertools – Iterator functions for efficient looping.

1. itertools.product : This tool computes the cartesian product of input iterables. It is equivalent to nested for-loops.

```
from itertools import product
print(list(product([1,2,3],repeat = 2)))
```

Output :

[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]

2. itertools.permutations(iterable[, r]) :

It returns successive r length permutations of elements in the iterable.

If r is not specified or is None, then r defaults to the length of the iterable and all possible full-length permutations are generated.

```
from itertools import permutations
l=['1','2','3']
print(list(permutations(l)))
```

Output :

[('1', '2', '3'), ('1', '3', '2'), ('2', '1', '3'), ('2', '3', '1'), ('3', '1', '2'), ('3', '2', '1')]

3. itertools.combinations(iterable[, r]) :

This tool returns the length subsequences of elements from the input iterable.

Combinations are emitted in lexicographic sorted order.

```
from itertools import combinations
l=['1','2','3']
print(list(combinations(l,2)))
```

Output :

[('1', '2'), ('1', '3'), ('2', '3')]

Bisect

This module provides support for maintaining a list in sorted order without having to sort the list after each insertion. The module is called bisect because it uses a basic bisection algorithm to do its work.

The following functions are provided:

bisect.bisect_left(list, item[, lo[, hi]])

Locate the proper insertion point for item in list to maintain sorted order. The parameters lo and hi may be used to specify a subset of the list which should be

considered; by default the entire list is used. If item is already present in list, the insertion point will be before (to the left of) any existing entries.

bisect.bisect_right(list, item[, lo[, hi]])

Similar to bisect_left(), but returns an insertion point which comes after any existing entries of item in list.

bisect.bisect(…)

Alias for bisect_right().

bisect.insort_left(list, item[, lo[, hi]])

Insert item in list in sorted order. This is equivalent to list.insert(bisect.bisect_left(list, item, lo, hi), item). This assumes that list is already sorted.

bisect.insort_right(list, item[, lo[, hi]])

Similar to insort_left(), but inserting item in list after any existing entries of item.

bisect.insort(…)

Alias for insort_right().

```
import bisect
l = []
print(" ENTER ANY 5 ELEMENTS ")
for i in range(0,5):
  c=int(input())
  bisect.insort(l,c)
print(l)
```

Output

ENTER ANY 5 ELEMENTS :

-> 5

-> 3

-> 8

-> 9

-> 2

[2, 3, 5, 8, 9]

Python Regex

ReGex, also termed as Regular Expression, is the efficient way of handling regular expressions in a Python Program. It generally involves a sequence of characters that forms a search pattern.
To work with regular expressions, we have to import **re** module.

Important Regex Functions :

re.findall() : It returns a list containing all matches.
Example : Below code will the find the existence of word “geek” in a string.

```
import re
s = ‘code of geeks is for programming geeks’
x = re.findall(‘geek’,s)
print(x)
```

Output
[‘geek’,’geek’]

re.search() : It searches the string for a match. In the case of more than 1 match only the first occurrence of the match will be returned.
Example : Below code will the find the existence of word ‘a’ in a string.

```
import re
s = ‘sdadfghja’
x = re.search(‘a’,s)
print(x.start())
```

Output
2

re.split() : It is used to split a string into list, as per the separator specified.
Example : Below code will split the string into a list taking ‘-‘ as a separator.

```
import re
txt = "code-of-geeks"
x = re.split("-", txt)
print(x)
```

Output
[‘code’, ‘of’, ‘geeks’]

re.sub() : It replaces the matches as per the text specified.
Example : Below code will replace “geeks” to “god”

```
import re
s = "code of geeks"
x = re.sub("geeks","god",s)
print(x)
```

Output
code of god

****