Python_Interview_Coding

April 27, 2021

```
[ ]: thislist = ["apple", "banana", "cherry"]
     print(thislist[::-1])
```

```
['cherry', 'banana', 'apple']
```

0.0.1 List Operations

Append

```
[ ]: x = [1,2,3]
```

```
[ ]: x.append(4)
     x
```

```
[ ]: [1, 2, 3, 4]
```

Clear

```
[ ]: x.clear()
     x
```

```
[ ]: []
```

Copy

```
[ ]: x = [1,2,3]
     y = x.copy()
```

```
[ ]: y
```

```
[ ]: [1, 2, 3]
```

Count

```
[ ]: x.count(3)
```

```
[ ]: 1
```

Extend

```
[ ]: x.extend(y)
[ ]: x
```

```
[ ]: [1, 2, 3, 1, 2, 3]
```

Index

```
[ ]: x.index(1)
```

```
[ ]: 0
```

Pop

```
[ ]: x.pop(2)
```

```
[ ]: 3
```

```
[ ]: x
```

```
[ ]: [1, 2, 1, 2, 3]
```

Remove

```
[ ]: x.remove(1)
     x
```

```
[ ]: [2, 1, 2, 3]
```

Reverse

```
[ ]: x.reverse()
```

```
[ ]: x
```

```
[ ]: [3, 2, 1, 2]
```

Sort

```
[ ]: x.sort()
     x
```

```
[ ]: [1, 2, 2, 3]
```

0.0.2 String Operations

capitalize() Converts the first character to upper case
casefold() Converts string into lower case
center() Returns a centered string count() Returns the number of times a specified value occurs in a string
encode() Returns an encoded version of the string
endswith() Returns true if the string ends with the specified value
expandtabs() Sets the tab size of the string
find() Searches the string for a specified value and returns the position of where it was found
format() Formats specified values in a string

format_map() Formats specified values in a string
index() Searches the string for a specified value and returns the position of where it was found
isalnum() Returns True if all characters in the string are alphanumeric
isalpha() Returns True if all characters in the string are in the alphabet
isdecimal() Returns True if all characters in the string are decimals
isdigit() Returns True if all characters in the string are digits
isidentifier() Returns True if the string is an identifier
islower() Returns True if all characters in the string are lower case
isnumeric() Returns True if all characters in the string are numeric
isprintable() Returns True if all characters in the string are printable
isspace() Returns True if all characters in the string are whitespaces
istitle() Returns True if the string follows the rules of a title
isupper() Returns True if all characters in the string are upper case
join() Joins the elements of an iterable to the end of the string
ljust() Returns a left justified version of the string
lower() Converts a string into lower case
lstrip() Returns a left trim version of the string
maketrans() Returns a translation table to be used in translations
partition() Returns a tuple where the string is parted into three parts
replace() Returns a string where a specified value is replaced with a specified value
rfind() Searches the string for a specified value and returns the last position of where it was found
rindex() Searches the string for a specified value and returns the last position of where it was found
rjust() Returns a right justified version of the string
rpartition() Returns a tuple where the string is parted into three parts
rsplit() Splits the string at the specified separator, and returns a list
rstrip() Returns a right trim version of the string split() Splits the string at the specified separator, and returns a list
splitlines() Splits the string at line breaks and returns a list
startswith() Returns true if the string starts with the specified value
strip() Returns a trimmed version of the string
swapcase() Swaps cases, lower case becomes upper case and vice versa
title() Converts the first character of each word to upper case
translate() Returns a translated string
upper() Converts a string into upper case
zfill() Fills the string with a specified number of 0 values at the beginning
Note: All string methods returns new values. They do not change the original string.

[ ]:
```
s = "satwik"
```

capitalize() Converts the first character to upper case

[ ]:
```
s.capitalize()
```

[ ]:
```
'Satwik'
```

[ ]:
```
s
```

[ ]: 'satwik'

casefold() Converts string into lower case

[ ]:
```
s = s.capitalize()
s
```

[ ]: 'Satwik'

[ ]:
```
s.casefold()
```

[ ]: 'satwik'

center() Returns a centered string

[ ]:
```
s = "satwik ram k"
```

[ ]:
```
s.center(20, "0")
```

[ ]: '0000satwik ram k0000'

count() Returns the number of times a specified value occurs in a string

[ ]:
```
s.count("a")
```

[ ]: 2

encode() Returns an encoded version of the string

[ ]:
```
print(s.encode())
```

```
b'satwik ram k'
```

endswith() Returns true if the string ends with the specified value

[ ]:
```
s.endswith("a")
```

[ ]: False

isupper() Returns True if all characters in the string are upper case

[ ]:
```
s.isupper()
```

[ ]: False

islower() Returns True if all characters in the string are lower case

[ ]:
```
s.islower()
```

[ ]: True

Upper() Returns all the Charecters with Upper

```
[ ]: s = s.upper()
```

```
[ ]: s
```

```
[ ]: 'SATWIK RAM K'
```

#### lower() Returns all the Charecters with lower

```
[ ]: s = s.lower()
```

```
[ ]: s
```

```
[ ]: 'satwik ram k'
```

join() Joins the elements of an iterable to the end of the string

```
[ ]: myTuple = ("John", "Peter", "Vicky")


x = " ".join(reversed(myTuple))

print(x)
```

```
Vicky Peter John
```

replace() Returns a string where a specified value is replaced with a specified value

```
[ ]: txt = "I like bananas"

y = txt.replace("bananas", "apples")

print(y)
```

```
I like apples
```

0.0.3 Removing Stop Words

```
[ ]: import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[ ]: True
```

```
[ ]: stopwords = stopwords.words()
```

```
[ ]: type(stopwords)
```

```
[ ]: list
```

[ ]:
```
a = "I am Machine Learning Engineer at Google"
```

[ ]:
```
a = a.split()
```

[ ]:
```
len(a)
```

[ ]:
```
7
```

[ ]:
```
b = []
```

[ ]:
```
for i in range(0, len(a)):
  if a[i] not in stopwords:
    b.append(a[i])
```

[ ]:
```
b = " ".join(b)
```

[ ]:
```
b
```

[ ]:
```
'I Machine Learning Engineer Google'
```

0.0.4 Regular Expression

[ ]:
```
import re
```

Re Methods findall Returns a list containing all matches
search Returns a Match object if there is a match anywhere in the string
split Returns a list where the string has been split at each match
sub Replaces one or many matches with a string

[ ]:
```
tweet = "@satwikram29 Happy Birthday!!!!!"
```

findall Returns a list containing all matches

[ ]:
```
t = re.findall("Happy", tweet) # case sensitive
print(t)
```

```
['Happy']
```

search Returns a Match object if there is a match anywhere in the string

[ ]:
```
t = re.search("Happy", tweet)
print(t.start(), t.end())
```

```
13 18
```

split Returns a list where the string has been split at each match

[ ]:
```
t = re.split(" ", tweet, 2)
```

[ ]:
```
t
```

[ ]:
```
['@satwikram29', 'Happy', 'Birthday!!!!!']
```

sub Replaces one or many matches with a string Very Important

[ ]:
```
tweet
```

[ ]:
```
'@satwikram29 Happy Birthday!!!!!'
```

[ ]:
```
sub = re.sub(r"@", "", tweet)
sub = re.sub(r"!", "", sub)
sub = re.sub(r"[0-9]+", "", sub)
```

[ ]:
```
sub
```

[ ]:
```
'satwikram Happy Birthday'
```

[ ]:
```
sub = re.sub(r"@[a-zA-z0-9]+", "", tweet)
sub = re.sub(r"!", "", sub)
sub = re.sub(r"[0-9]+", "", sub)
sub = sub.strip()
```

[ ]:
```
sub
```

[ ]:
```
'Happy Birthday'
```

0.0.5 OOPS

[ ]:
```
class student:

  def __init__(self, fname, lname):
    self.fname = fname
    self.lname = lname

  def display(self):
    print(self.fname,"\n",self.lname)
```

[ ]:
```
obj = student("Satwik", "Ram")
```

[ ]:
```
obj.display()
```

```
Satwik
 Ram
```

[ ]:
```
obj.fname = "Satwik Ram"
```

[ ]:
```
obj.lname = "Kodandaram"
```

[ ]:
```
obj.display()
```

```
Satwik Ram
 Kodandaram
```

[ ]:
```
class studentemp(student):

  def display(self):
    print(self.fname+self.lname)
```

[ ]:
```
st = studentemp("Satwik", "Ram")
```

[ ]:
```
st.display()
```

```
SatwikRam
```

super()

[ ]:
```
class Person:

  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname):
    super().__init__(fname, lname)
```

[ ]:
```
x = Student("Mike", "Olsen")
x.printname()
```

```
Mike Olsen
```

0.0.6 Dictionary Methods

clear() Removes all the elements from the dictionary
copy() Returns a copy of the dictionary
fromkeys() Returns a dictionary with the specified keys and value
get() Returns the value of the specified key
items() Returns a list containing a tuple for each key value pair
keys() Returns a list containing the dictionary's keys
pop() Removes the element with the specified key
popitem() Removes the last inserted key-value pair
setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update() Updates the dictionary with the specified key-value pairs
values() Returns a list of all the values in the dictionary

clear() Removes all the elements from the dictionary

[ ]:
```
z = {"fname": "Satwik",
     "lname": "Ram"}
```

[ ]:
```
z.clear()
```

[ ]:
```
z
```

[ ]:
```
{}
```

copy() Returns a copy of the dictionary

[ ]:
```python
z = {"fname": "Satwik",
     "lname": "Ram"}
```

[ ]:
```python
y = z.copy()
y
```

[ ]:
```
{'fname': 'Satwik', 'lname': 'Ram'}
```

items() Returns a list containing a tuple for each key value pair

[ ]:
```python
a = z.items()
z['fname'] = "SSS"
```

[ ]:
```python
a
```

[ ]:
```
dict_items([('fname', 'SSS'), ('lname', 'Ram')])
```

keys() Returns a list containing the dictionary's keys

[ ]:
```python
z.keys()
```

[ ]:
```
dict_keys(['fname', 'lname'])
```

values() Returns a list of all the values in the dictionary

[ ]:
```python
z.values()
```

[ ]:
```
dict_values(['SSS', 'Ram'])
```

update() Updates the dictionary with the specified key-value pairs

[ ]:
```python
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

car.update({"color": "White"})

print(car)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'White'}
```

0.0.7 Lambda Functions

[ ]:
```
l = lambda x: x+10
```

[ ]:
```
l(10)
```

[ ]:
```
20
```

Interview Questions, Most Asked

0.0.8 Removing Duplicate Elements from List

[ ]:
```
list1 = [1,1,1,2,3,4,5,3,3,3,4,5,5,4,3]
```

[ ]:
```
list1 = list(set(list1))
```

[ ]:
```
list1
```

[ ]:
```
[1, 2, 3, 4, 5]
```

0.0.9 Print 0-20 numbers without using any numbers

[ ]:
```
test = "jjdnnffinifninfininfinf"

for i in range(len(test)):
  print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

0.0.10 Reverse a String or a List

[ ]:
```
list2 = [1,2,3,4,5]
list2 = list2[::-1] # Same for String also
list2
```

[ ]:
```
[5, 4, 3, 2, 1]
```

0.0.11 Remove all the odd elements from list using only one line code

[ ]:
```
b = [1,2,3,4,5,6,7,8,9,10]
```

[ ]:
```
b = [i for i in b if i % 2 == 0]
```

[ ]:
```
b
```

[ ]:
```
[2, 4, 6, 8, 10]
```

0.0.12 Interchange first and last element of the list

[ ]:
```
b
```

[ ]:
```
[2, 4, 6, 8, 10]
```

[ ]:
```
def interchange(x):

  last = len(x) - 1
  a = x[0]
  x[0] = x[last]
  x[last] = a

  return x
```

[ ]:
```
b = interchange(b)
```

[ ]:
```
b
```

[ ]:
```
[10, 4, 6, 8, 2]
```

0.0.13 find smallest number in a list without using builtin

[ ]:
```
b
```

[ ]:
```
[10, 4, 6, 8, 2]
```

[ ]:
```
def minimum(x):

  min = x[0]

  for i in x:
    if i < min:
      min = i

  return min
```

[ ]:
```
print(minimum(b))
```

```
2
```

0.0.14 Sort a list without using Builtin

[ ]:
```
b
```

[ ]:
```
[10, 4, 6, 8, 2]
```

[ ]:
```
def sortlist(x):

  dummy = []

  while x:
    mini = min(x)
    dummy.append(mini)
    x.remove(mini)

  return dummy
```

[ ]:
```
b = [10,9,8,7,6,7,8,4,3,3,2,1]
```

[ ]:
```
sortlist(b)
```

[ ]:
```
[1, 2, 3, 3, 4, 6, 7, 7, 8, 8, 9, 10]
```

[ ]:
```

```

[ ]:
```

```