**Python exercises**

1. **Generate the following dictionary with 2 lists: {'a': 1, 'b': 2, 'c': 3}**

L1 = [‘a’,’b’,’c’]  
L2 = [1,2,3]  
>>> dict(zip(l1,l2))  
{'a': 1, 'b': 2, 'c': 3}

1. Write a Python program to sum all the items in a list.

>>> sum([randrange(1,1000) for \_ in range(10)])

Sum([84, 636, 698, 772, 987, 151, 474, 549, 150, 635])

1. Write a Python program to count the number of strings where the string length is 2 or more and the first and last character are same from a given list of strings. [Go to the editor](https://www.w3resource.com/python-exercises/list/#EDITOR)   
   Sample List : ['abc', 'xyz', 'aba', '1221']  
   Expected Result : 2

l1 = [**'aba'**,**'xx'**,**'12521'**,**'z'**,**'ooopppp'**]  
count = 0  
**for** word **in** l1:  
 **if** word[0]==word[-1] **and** len(word)>2:  
 count+=1  
print(count)

1. Write a Python program to get a list, sorted in increasing order by the last element in each tuple from a given list of non-empty tuples. [Go to the editor](https://www.w3resource.com/python-exercises/list/#EDITOR)   
   Sample List : [(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]  
   Expected Result : [(2, 1), (1, 2), (2, 3), (4, 4), (2, 5)]

l1 = [(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]  
l1.sort(key=**lambda** x:x[-1])  
print(l1)

1. Write a Python program to remove duplicates from a list.

L1 = [1,2,4,4,5,6]

Print(set(l1))

1. Write a Python program to find the list of words that are longer than n from a given list of words.

L1 = [‘words’,’are’,’the’,’expression’] and n=3

**for** word **in** l2:  
 **if** len(word)>n:  
 print(word)

1. Write a Python function that takes two lists and returns True if they have at least one common member.
2. Write a Python program to create a lambda function that adds 15 to a given number passed in as an argument, also create a lambda function that multiplies argument x with argument y and print the result. [Go to the editor](https://www.w3resource.com/python-exercises/lambda/index.php#EDITOR).

>>> func = lambda x: x+15  
>>> func(5)  
20

>>> func2 = lambda x,y: x\*y  
>>> func2(5,6)  
30

1. Write a Python program to create a function that takes one argument, and that argument will be multiplied with an unknown given number.

>>> func3 = lambda x: x\*random()  
>>> func3(2)

1. Write a Python program to sort a list of tuples using Lambda.

another = [(-78, 83), (-1, 90), (1, 2), (23, 54)]

>>> another.sort(key=lambda x:x[1])

>>> another

1. Write a Python program to sort a list of dictionaries using Lambda. Sort by ascending and descending order.

**Ascending order (order by make and model)**

**models = [{'make':'Nokia', 'model':216, 'color':'Black'}, {'make':'Mi Max', 'model':2, 'color':'Gold'}, {'make':'Samsung', 'model': 7, 'color':'Blue'}]**

>>> sorted(models, key= lambda x:x['make'])

[{'make': 'Mi Max', 'model': 2, 'color': 'Gold'}, {'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}]

>>> sorted(models, key=lambda x:x['model'])

[{'make': 'Mi Max', 'model': 2, 'color': 'Gold'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Nokia', 'model': 216, 'color': 'Black'}]

**Descending order (order by make and model)**

>>> sorted(models, key=lambda x:x['model'], reverse=True)  
[{'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Mi Max', 'model': 2, 'color': 'Gold'}]

>>> sorted(models, key=lambda x:x['make'], reverse=True)  
[{'make': 'Samsung', 'model': 7, 'color': 'Blue'}, {'make': 'Nokia', 'model': 216, 'color': 'Black'}, {'make': 'Mi Max', 'model': 2, 'color': 'Gold'}]

**Ascending order by value**

Dict1 = {'KIA': 130000, 'Subaru': 145000, 'Honda': 0}

Sorted\_dict= sorted(dict1.items(), key=lambda x:x[1])

**Descending order by value**

Sorted\_dict= sorted(dict1.items(), key=lambda x:x[1], reverse=True)

**Map and Filter**

1. Write a Python program to filter a list of integers using Lambda.

L22 = [2,3,4,5,2,3,1]

odd\_nums = list(filter(lambda x: x%2 !=0, L22))

[3, 5, 3, 1]

Don’t display the number 2

do\_not\_display\_number\_two = list(filter(lambda x:x!=2, l22))

1. Write a Python program to square and cube every number in a given list of integers using Lambda.

>>> square\_nums = list(map(lambda x:x\*\*2, L22))

>>> square\_nums

[4, 9, 16, 25, 4, 9, 1]

>>> cube\_nums = list(map(lambda x:x\*\*3,L22))

>>> cube\_nums

[8, 27, 64, 125, 8, 27, 1]

1. Write a Python program to find if a given string starts with a given character using Lambda.

S = ‘aaaerer’

starts\_with = lambda x: True if x.startswith(‘a’) else False

True

1. Write a Python program to extract year, month, date and time using Lambda.

>>> from datetime import datetime

>>> print(datetime.today())

2020-10-15 18:12:52.241354

>>> print(datetime.now())

2020-10-15 18:12:58.203412

>>> year = lambda x:x.year

>>> month = lambda x:x.month

>>> day = lambda x:x.day

>>> year(datetime.now())

2020

>>> month(datetime.now())

10

>>> day(datetime.now())

15

1. Write a Python program to check whether a given string is number or not using Lambda.

is\_number = lambda x: True if type(x)==int or type(x)==float else False

1. Write a Python program to create Fibonacci series up to n using Lambda.

>>> from functools import reduce

>>> fib\_series = lambda n: reduce(lambda x, \_:x+[x[-1]+x[-2]], range(n-2), [0,1])

1. Write a Python program to count the even, odd numbers in a given array of integers using Lambda.

L = [2,3,4,5,5,6,7,8,9,9]

even\_number = len(list(filter(lambda x:x if x%2==0 else '',L)))

odd\_number = len(list(filter(lambda x:x if x%2!=0 else ‘’, L))

1. Write a Python program to find the values of length six in a given list using Lambda.

other = ['oi','t11111r','343','43435llpp']

>>> value\_six = filter(lambda x:x if len(x)>6 else '',other)

>>> for d in value\_six:

... print(d)

1. Write a Python program to find the second lowest grade of any student(s) from the given names and grades of each student using lists and lambda. Input number of students, names and grades of each student.
2. Write a Python program to find numbers divisible by nineteen or thirteen from a list of numbers using Lambda

l2 = [13,23,36,11,90]  
res = list(filter(lambda x: (x%13==0 or x%19==0),l2))

1. Write a Python program to find the numbers of a given string and store them in a list, display the numbers which are bigger than the length of the list in sorted form. Use lambda function to solve the problem.

>>> str1 = "sdf 23 safs8 5 sdfsd8 sdfs 56 21sfs 20 5"  
>>> str\_num = [i for i in str1.split(' ')]  
>>> str\_num  
['sdf', '23', 'safs8', '5', 'sdfsd8', 'sdfs', '56', '21sfs', '20', '5']  
>>> numbers = sorted([int(x) for x in str\_num if x.isdigit()])  
>>> numbers  
[5, 5, 20, 23, 56]

1. Write a Python program that multiply each number of given list with a given number using lambda function. Print the result.

>>> desired\_number = 7

>>> multi = list(map(lambda x:x\*desired\_number, numbers))

>>> multi

[35, 35, 140, 161, 392]

1. Write a Python program that sum the length of the names of a given list of names after removing the names that starts with an lowercase letter. Use lambda function

sample\_names = ['sally', 'Dylan', 'rebecca', 'Diana', 'Joanne', 'keith']

>>> res = list(filter(lambda el:el[0].isupper() and el[1:].islower(),sample\_names))

>>> res

['Dylan', 'Diana', 'Joanne']

>>>len(‘’.join(res))

1. Replace the \n from the l = ['Welcome to w3resource.com.\n', 'Append this text.Append this text.Append this text.\n', 'Append this text.\n']

L2 = list(map(lambda x:x.replace(“\n”,””),l))

**JSON**

1. Write a Python program to convert JSON data to Python object.

json\_obj = '{"Name":"David","Class":"II","Age":6}'

import json

python\_obj = json.loads(json\_obj)

print(“\nName:”,python\_obj[“Name”])

1. Write a Python program to convert a Python object to JSON.

import json  
python\_obj\_dict = {"name":"David O.","class":"IV","age":6}  
json\_data = json.dumps(python\_obj\_dict)

1. Write a Python program to convert Python dictionary object (sort by key) to JSON data.

import json

>>> j\_str = {1:'89',-2:'90',0:'po'}

>>> json\_object = json.dumps(j\_str,sort\_keys=True)

>>> json\_object

'{"-2": "90", "0": "po", "1": "89"}'

1. Write a Python program to create a new JSON file from an existing JSON file.

import json

with open(‘states.json’) as f:

state\_data = json.load(f)

for state in state\_data[‘states’]:

del state[‘area\_codes’]

1. Write a Python program to check whether a JSON string contains complex object or not.

import json

def is\_complex\_num(objct):

if '\_\_complex\_\_' in objct:

return complex(objct['real'], objct['img'])

return objct

complex\_object =json.loads('{"\_\_complex\_\_": true, "real": 4, "img": 5}', object\_hook = is\_complex\_num)

simple\_object =json.loads('{"real": 4, "img": 3}', object\_hook = is\_complex\_num)

print("Complex\_object: ",complex\_object)

print("Without complex object: ",simple\_object)

1. Write a Python program to check whether an instance is complex or not.

import json

def encode\_complex(object):

# check using isinstance method

if isinstance(object, complex):

return [object.real, object.imag]

# raised error if object is not complex

raise TypeError(repr(object) + " is not JSON serialized")

complex\_obj = json.dumps(2 + 3j, default=encode\_complex)

print(complex\_obj)

1. Access the value of key2 from the following JSON

{"key1": "value1", "key2": "value2"}

import json

sampleJson = """{"key1": "value1", "key2": "value2"}"""

# write code to print the value of key2

json\_loaded = json.loads(sampleJson)

json\_loaded[“key2”]

1. Pretty printing following json data with indent level 2and key value separators should be (“,”,”=”).

import json

sampleJson = {"key1" : "value2", "key2" : "value2", "key3" : "value3"}

prettyPrintedJson = json.dumps(sampleJson, indent=2, separators=(",", " = "))

print(prettyPrintedJson)

1. Sort following JSON data alphabetical order of keys

sampleJson = {"id" : 1, "name" : "value2", "age" : 29}

import json

json\_obj = json.dumps(sampleJson, sort\_keys=True)

1. Traverse the following json object to get the salary.

import json

sampleJson = """{

"company":{

"employee":{

"name":"emma",

"payble":{

"salary":7000,

"bonus":800

}

}

}

}"""

loaded\_json = json.loads(sampleJson)

loaded\_json[‘company’][‘employee’][‘name’][‘payable’][‘salary’]

1. Convert the following vehicle object into JSON.

import json

from json import JSONEncoder

class Vehicle:

def \_\_init\_\_(self, name, engine, price):

self.name = name

self.engine = engine

self.price = price

class VehicleEncoder(JSONEncoder):

def default(self, o):

return o.\_\_dict\_\_

vehicle = Vehicle("Toyota Rav4", "2.5L", 32000)

vehicle2json = json.dumps(vehicle, indent=4, cls=VehicleEncoder)

1. Convert the following JSON object to class.

{ "name": "Toyota Rav4", "engine": "2.5L", "price": 32000 }

import json

class Vehicle:

def \_\_init\_\_(self, name, engine, price):

self.name = name

self.engine = engine

self. Price = price

def vehicleDecoder(obj):

return Vehicle(obj[‘name’],obj[‘engine’],obj[‘price’])

json\_to\_class = json.loads(‘{ "name": "Toyota Rav4", "engine": "2.5L", "price": 32000 }

’, object\_hook=vehicleDecoder)

1. Check whether the following json is valid or invalid. If it is invalid, correct it.

{

"company":{

"employee":{

"name":"emma",

"payble":{

"salary":7000

"bonus":800

}

}

}

}

import json

def validateJSON(jsonData):

try:

json.loads(jsonData)

except ValueError as err:

return False

return True

InvalidJsonData = """{ "company":{ "employee":{ "name":"emma", "payble":{ "salary":7000 "bonus":800} } } }"""

isValid = validateJSON(InvalidJsonData)

1. Parse the following JSON to get all the values of a key ‘name’ within an array.

Sample = “””[

{

"id":1,

"name":"name1",

"color":[

"red",

"green"

]

},

{

"id":2,

"name":"name2",

"color":[

"pink",

"yellow"

]

}

]”””

import json

data = []

try:

data = json.loads(sampleJson)

except Exception as error:

print(error)

dataList = [item.get[‘name’]for item in data]

print(dataList)

1. How do you make a get and post request?

# get request

import requests

# api end-point

URL = ‘http://www.my-global-tickets.com’

# define parameters to extract

PARAMS = {‘location’}

response = requests.get(url = URL, params= PARAMS)

# extract the data into json

data = response.json()

# extracting the values of the location

department = data[‘results’][0][‘building’]

city = data[‘results’][0]

# post request

Import requests

API\_ENDPOINT = ‘http://…’

# API key

API\_KEY = XXXXXXX

# data to be sent

payload = {

‘api\_dev\_key’:API\_KEY,

}

response = requests.post(url = API\_ENDPOINT, data=payload)

# extracting response to text

Txt\_data = response.text

1. How do you read an XML file?

import xmltodict

**def** read\_xml\_file(xmlfile):  
 **with** open(xmlfile) **as** fd:  
 doc = xmltodict.parse(fd.read())  
  
 **for** index **in** range(len(doc[**'CATALOG'**][**'CD'**])):  
 print(doc[**'CATALOG'**][**'CD'**][index][**'TITLE'**])  
 print(doc[**'CATALOG'**][**'CD'**][index][**'TITLE'**])

**Strings**

1. Given a string of odd length greater 7, return a string made of the middle three chars of a given string.

Input: str1 = "JhonDipPeta"

Output: Dip

Input: str2 = "JaSonAy"

Output: Son

From math import floor

Length\_str2 = len(str2)/2

str2[**floor(length\_str2)-1**:floor(length\_str2)+2]

The -1 is used because we are counting from 0 to n, so I need to exclude 0.

1. Given a string of odd length greater 7, return a string made of the middle three chars of a given string.

s1 = "Ault"

s2 = "Kelly"

'AuKellylt'

s1[0:int(len(s1)/2)]+s2+s1[int(len(s1)/2):]

1. Given 2 strings, s1 and s2 return a new string made of the first, middle and last char each input string:

S1 = “Mexico”

S2 = “Germany”

**S3 = “MGxmoy”**

S1 = “Japan”

S2 = “America”

**S3 = “AJrpan”**

**def** merge\_strings(s1, s2):  
  
 first\_char = s1[0]+s2[0]  
 **if** len(s1)%2 == 0: *# even* middle\_char = s1[int(len(s1)/2)-1:int(len(s1)/2)]

**else**:  
 middle\_char = s1[int(len(s1)/2):int(len(s1)/2)+1]  
  
 **if** len(s2)%2 == 0: *# evens* middle\_char += s2[int(len(s2)/2)-1:int(len(s2)/2)]

**else**:  
 middle\_char += s2[int(len(s2)/2):int(len(s2)/2)+1]  
  
 last\_char = s1[-1]+s2[-1]  
 **return** first\_char + middle\_char + last\_char

def main():

s1 = **"America"** s2 = **"Japan"** res = merge\_strings(s1, s2)  
 print(res)

I am using -1 in even because I need to exclude the 0, otherwise I would end up with the “i” from Mexico instead of the “x”. I am using +1 in odd to consider the number that I got from the division, because it goes **[0:1] the fist value is inclusive but the second one is not.**

1. Given a string, return the number of occurrences of each letter.

str1 = “Apple”

output: {‘A’:1, ‘p’:2, ‘l’:1, ‘e’:1}

There are two ways: normal dict and a defaultdict

# normal dict

str1 = “apple”

count = dict()

for l in str1:

if l in count:

count[l]+=1

else:

count[l]=1

print(count)

# defaultdict

from collections import defaultdict

count2 = defaultdict(int)

for l in str1:

count2[l]+=1

print(count2)

In a defaultdict, you will not have the problem of a missing key when it doesn’t exist previously. In contrast, a normal dict cannot have an increase of value if the key does not exist previously.

1. Reverse a string.

str1 = “Pynative”

output = “evitanyP”

# convert this into a list, so you can swap the letters.

# string indexes cannot be swapped, you need to convert the string into a list

str1\_list = list(str1)

left = 0

right = len(str1)-1

while left<=right:

str1\_list[left], str1\_list[right] = str1\_list[right], str1\_list[left]

left +=1

right -=1

print(“”.join(str1\_list))

1. Remove all the characters other than integers from string.

Input: s= ‘I am 25 years and 10 months old’

output: 2510

res = “”.join([x for x in s if x.isdigit()])

1. Find words with both alphabets and numbers.

Input: str1 = "Emma25 is Data scientist50 and AI Expert"

Output:

Emma25  
scientist50

1. From given string replace each punctuation with #.

Input: str1 = '/\*Jon is @developer & musician!!'

Output: ##Jon is #developer # musician##

res = map(replace(), str1)

>> res = ‘’

>>> for c in str1:

... if c in values:

... res += c.replace(c,'#')

... else:

... res+=c

1. Find words with both alphabets and numbers.

str1 = "Emma25 is Data scientist50 and AI Expert"

1. Given two lists, create a third list by picking an odd-index element from the first list and even index elements from second.

Input:

listOne = [3, 6, 9, 12, 15, 18, 21]

0 1 2 3 4 5 6

listTwo = [4, 8, 12, 16, 20, 24, 28]

0 1 2 3 4 5 6

Output:

Element at odd-index positions from list one

[6, 12, 18]

Element at even-index positions from list two

[4, 12, 20, 28]

Printing Final third list

[6, 12, 18, 4, 12, 20, 28]

for i, x in enumerate(listOne):

# make the index+1 to get the even index position

if (i+1)%2==0:

res.append(x)

for i, x in enumerate(listTwo):

if (i+1)%2!=0:

res.append(x)

>>> oddElements = listOne[1::2]

>>> oddElements

[6, 12, 18]

>>> evenElements = listTwo[0::2]

>>> evenElements

[4, 12, 20, 28]

>>> res.extend(oddElements)

>>> res

[6, 12, 18]

>>> res.extend(evenElements)

>>> res

[6, 12, 18, 4, 12, 20, 28]

1. Leetcode list from Fb leetcode
2. DevOps questions (python with apis)
3. Python exercises (strings, arrays)

What is json serialized?

What is a complex object in Python?

Create a 20 random numbers from 0 to 1000, then