**Python LC’s and CC’s**

**Variables and Operators/ LC**

1. Automatic volume detection

length = int(input())

width = int(input())

height = int(input())

print(length\*width\*height )

2. The Cargo Coordinates

import sys

x = int(input())

y = int(input())

z = int(input())

s= x/y

oneRow = s\*s

total = s\*s\*s\*s

num = 0

zvalue = 0

yvalue = 0

xvalue = 0

if(z<=total):

num=z%oneRow

zvalue=int((z//oneRow))

yvalue=int((num//s))

xvalue=int((z%s)-1)

if((z%s)== 0):

xvalue=4

yvalue=yvalue-1

n1=str(xvalue)

n2=str(yvalue)

n3=str(zvalue)

print("("+n1+","+n2+","+n3+")"+"\n")

else:

print("Invalid input")

3.The Cargo Coordinates :

import sys

x = input()

y = input()

z = input()

s= x/y

oneRow = s\*s

total = s\*s\*s

num = 0

zvalue = 0

yvalue = 0

xvalue = 0

if(z<=total):

num=z%oneRow

zvalue=int((z//oneRow))

yvalue=int((num//s))

xvalue=int((num%s)-1)

n1=str(xvalue)

n2=str(yvalue)

n3=str(zvalue)

sys.stdout.write("("+n1+","+n2+","+n3+")"+"\n")

**Decision Making / LC**

1. Identify the Container type

x=int(input())

if x<1001:

print("Saver")

if x>1000 and x<10001:

print("Economy")

if x>10000:

print("Flexi")

2. Locate the Container

x=int(input())

y=int(input())

z=int(input())

a=int(input())

b=int(input())

if b<x-1:

print(1)

elif b<(x+y-1) and b>(x-1 ):

print(2)

elif b<=(x+y+z-1) and b>(x+y-1):

print(3)

elif b<=(x+y+z+a-1) and b>(x+y+z-1):

print(4)

else:

print("Not Possible")

**Looping\_Statements**

**1.** **The Cargo arrangement**

i=int(input())

j=((i+i)-1)

for x in range(i,0,-1):

a='#'\*(x-1)

b='\*'\*(j-(x+x)+2)

c='+'\*(x-1)

print ("%s%s%s" %(a,b,c))

**2.** **The Weight Detection Sensor**

x=int(input())

y=0

while True:

z=int(input())

if z<x:

x=x-z

y+=1

elif z==x:

y+=1

break

else:

break

print (y)

**Working with Lists / LC**

1. Sequencing the Cargos

def main():

L=[]

x=input()

k=int(input())

y=x.split(',')

count=0

while count<len(y):

newInts=int(y[count])

count=count+1

L.append(newInts)

L.sort()

n=len(L)

sum=0

for i in range((n-k),n):

sum=sum+L[i]

print(sum)

main()

2. The Lounge Numbers

data=input()

myList=data.split(",")

srch=input()

indices=[i for i,x in enumerate(myList) if x==srch]

if len(indices)>0:

for y in indices:

print(y+1)

else:

print(0)

3. Time to the Destination

import math as m

import sys

ports = int(input())

flag=0

if ports >100:

print("Invalid input")

flag=1

if flag==0:

a = []

for \_ in range(ports):

temp = list(map(int,input().split()))

if temp[0]< 0 or temp[0]>1000 or temp[1]<1 or temp[1]>1000 or temp[2]<1 or temp[2]>1000:

print("Invalid input")

flag=1

break

a.append(temp)

if flag==0:

totalTime=a[0][1]

for i in range(1,ports):

if totalTime <= a[i][0]:

totalTime+= a[i][0]-totalTime + a[i][1]

else:

x = (totalTime - a[i][0])/a[i][2]

x = m.ceil(x)

arrival = (a[i][2]\*x)+a[i][0]

diff = abs(totalTime-arrival)

totalTime += diff+a[i][1]

print(totalTime)

**Working with Strings / LC**

1. Validation of Employee Email ID

import re

x=str(input())

y=x.split('.')

z=y[0].split('@')

k=len(z[0])

L2=['com','edu','in']

a=0

b=0

c=0

if(re.match("^[a-zA-Z0-9][\w\.]+[a-zA-Z0-9]$",z[0])!=None) and (z[0].islower()==True) and (len(z[1])>3) and (y[1] in L2):

print("Valid")

else:

print("Invalid")

if(re.match("^[a-zA-Z0-9][\w\.]+[a-zA-Z0-9]$",z[0])==None) or z[0].islower()==False:

a=3

if(len(z[1])<=3):

b=2

if y[1] not in L2:

c=1

if c!=0:

print(c)

if b!=0:

print(b)

if a!=0:

print(a)

**2.** **Validation of Invoice Number**

import re

invoice=input()

try:

src,dest=[s for s in re.split('[0-9]',invoice) if s]

if not invoice[-1].isnumeric():raise RuntimeError('Invalid Input')

print(src,'to',dest)

except:

print('Invalid Input')

**Working with Tuples / LC**

1. Commodity Weight

data=input()

N=int(input())

nu=data.split(" ")

count=0

myList=[]

while count<len(nu):

newInt=int(nu[count])\*N

count=count+1

myList.append(newInt)

print(tuple(myList))

2.List of Tuples

from datetime import datetime,time

n=int(input())

mytuple=[]

for x in range(n):

ui1=tuple(map(str,input().split(',')))

mytuple.append(ui1)

date1=input()

print (mytuple)

for itm in mytuple:

fildat=datetime.strptime(date1,"%d-%m-%Y")

imdat=datetime.strptime(itm[1],"%d-%m-%Y")

if imdat>fildat:

print (itm[0])

**Dictionary Date and Time**

1. Expiry Date

from datetime import datetime, timedelta

expiry = input()

dep = input()

num = int(input())

exp = datetime.strptime(expiry, '%b %d %Y')

departure = datetime.strptime(dep, '%b %d %Y')

arr = departure + timedelta(days=num)

if arr < exp:

print("Yes")

else:

print("No")

2. Tracking the Shipment Dates

import datetime

from datetime import date

from datetime import timedelta

start=input()

n=int(input())

date=input()

date2=datetime.datetime.strptime(date,"%d-%m-%Y")

date3=date2.date()

work={'Monday':0,'Tuesday':1,'Wednesday':2,'Thursday':3,'Friday':4,'monday':0,'tuesday':1,'wednesday':2,'thursday':3,'friday':4}

for day,count in work.items():

if day==start:

weekday=work[day]

while(n-1!=0):

if (weekday==0) or (weekday==1) or (weekday==2) or (weekday==3):

j=date3+timedelta(days=1)

date3=j

output\_format=datetime.datetime.strftime(j,'%d-%m-%Y')

print (output\_format)

weekday=weekday+1

else:

j=date3+timedelta(days=3)

date3=j

output\_format=datetime.datetime.strftime(j,'%d-%m-%Y')

print (output\_format)

weekday=0

n=n-1

**Files, Input, Output**

**1.** **File Copy**

import shutil

shutil.copy2('file\_in.txt','file\_out.txt')

print(open('file\_out.txt').read())

2. **Heaviest Cargos**

import string

import collections

import re

from heapq import nlargest

k=int(input("Enter the n value\n"))

def heaviest():

myDict={}

line\_list=[line.rstrip('\n') for line in open("readlines.txt").readlines()]

for value in line\_list:

x=int(re.search(r'\d+',value).group())

myDict[value]=x

three\_largest=nlargest(k,myDict,key=myDict.get)

for i in reversed(three\_largest):

print(i)

heaviest()

3. **XML to CSV**

import xml.etree.ElementTree as ET

tree=ET.parse('product.xml')

root=tree.getroot()

print ('id,productName,cost,weight')

for product in root.findall('product'):

print (product.find('id').text+','+product.find('productName').text+','+product.find('cost').text+','+product.find('weight').text)

**Functions**

1. Loading Expressions

def EvaluateB1(str) :

open = False

count = 0

for x in str :

if (x == '(') :

open = True

count += 1

elif (open ==True) and (x == ')') :

count -= 1

if count == 0:

open = False

elif count < 0 :

#print("x")

return False

elif (open ==False) and (x == ')') :

#print("y")

return False

elif (open == True) and (( x == '{') or ( x == '}') or ( x == '[') or ( x == ']')) :

#print("z")

return False

return count == 0

def EvaluateB2(str):

open = False

count = 0

for x in str :

if (open ==False) and (x == '[') :

open = True

count += 1

elif (open ==True) and (x == ']') :

open = False

count -= 1

elif (open ==False) and (x == ']') :

#print("y")

return False

elif (open == True) and (( x == '{') or ( x == '}') or ( x == '[') ) :

#print("z")

return False

return count == 0

def EvaluateB3(str):

open = False

count = 0

for x in str :

if (open ==False) and (x == '{') :

open = True

count += 1

elif (open ==True) and (x == '}') :

open = False

count -= 1

elif (open ==False) and (x == '}') :

#print("y")

return False

elif (open == True) and ( x == '{') :

#print("z")

return False

return count == 0

def Validate(str):

#print EvaluateB1(str)

#print EvaluateB2(str)

#print EvaluateB3(str)

if EvaluateB1(str) and EvaluateB2(str) and EvaluateB3(str) :

print ("Valid")

else :

print("Invalid")

x1 =input()

Validate(x1)

2. Palindromic Cargo

import string

import re

x=str(input())

y=''.join(e for e in x if e.isalnum())

z=re.sub('[^a-zA-Z0-9]','',y)

a=z.lower()

if (a==a[::-1]):

print("Yes")

else:

print("No")