Program Exercises

Programs for List, Set, Dictionary, While loop, Functions

List programs **1.** program to make an empty list and fill it with values to a total of 10 and show output

**PRGORAM**

a=[]

print("Enter a set of values to make a list")

while len(a)<=9:

b=input("enter a value: ")

a.append(b)

print(a)

**OUTPUT**

Enter a set of values to make a list

enter a value: 1

enter a value: 2

enter a value: 3

enter a value: 4

enter a value: 5

enter a value: 6

enter a value: 7

enter a value: 8

enter a value: 9

enter a value: 10

['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']

**2.** Program to create two lists with even and odd numbers**,** by getting all the values from the user in a single line separated by commas**.** Predefine only two empty lists**,** remove the numbers that are divisible by 3 or 7**.** Make a sum of the even listand multiply the largest value with the smallest value in the odd list, combine both into a single listand print them using FOR and IN loops

**PROGRAM**

a=[] #Predefined empty lists

c=[]

#to get all the values in a single line from the user

b=input("Enter a series of numbers seperated by commas \",\": ")

a=b.split(",")

for i in a:

c.append(float(i))

a.clear() #clearing the list a to put into reuse

b=[] #converting the string b into a list

for j in c: #seperating the odd and even numbers into lists

if j%2==0:

a.append(j)

else:

b.append(j)

for k in a: #removing the values divisible by 3 and 7

if k%3==0 or k%7==0:

a.remove(k)

for l in b:

if l%3==0 or l%7==0:

b.remove(l)

c=0 #converting the list c into an integer

for i in a: #making a sum of the even list

c+=i

else:

a=0 #converting the list a into an integer

a=max(b)\*min(b) #multiplying the maximum and minimum values of the odd list

b.clear() #clearing list b for reuse

b.append(a)

b.append(c)

for j in b: #to output using FOR IN

print(j)

**OUTPUT**

Enter a series of numbers seperated by commas ",": 12,13,14,15,16,17,18,19,2,3,4,5,6,7,8,9,20,21,22,23,24,25,26,27,28

125.0

112.0

# Dictionary Programs

Predefine a dictionary with two elements {1:777,666:2}. Unpack and make a sum of both keys and values, create a series of values with multipliers from 10! (10 to 1). Multiply each value in the above series by the sum of unpacked dictionary. Assign a key to each value, keys must be strings from the user. Make a sorting mechanism where user can give sort commands, by keys sorted by ascending or descending order. The output must be received using For loop.

**PROGRAM**

a={1:777,666:2} #Predefined dictionary

b=list(a.items()) #unpacking a dictionary

a=list(b[0]+b[1])

b=0

for i in a:

b+=i #sum of keys and values

c=[]

d=11 #create a series of numbers from 10 to 1

while len(c)<=9:

e=d-1

c.append(e)

d=e-1

c.append(d)

f=[]

for j in c: #multiplying each value of the list by the sum

a=j\*b

f.append(a)

a=input("Enter 10 names seperated by commas \",\" to be associated to the values: ") #obtaining keys from user

b=a.split(",")

a= {b[i]: f[i] for i in range(len(b))} #creating a dictionary from two lists

#sorting mechanism for the dictionary a

c=list(a.items())

d=input(" ascending (a) or descending (d)? a/d ")

if d=="a":

c.sort()

else:

c.sort(reverse=True)

for i in c: #output by for\_in

print(i)

**OUTPUT**

Enter 10 names seperated by commas "," to be associated to the values: qwer,wer,ert,qwe,wert,ertty,sdf,dgh,ghj,asde

ascending (a) or descending (d)? a/d a

('asde', 1446)

('dgh', 4338)

('ert', 11568)

('ertty', 7230)

('ghj', 2892)

('qwe', 10122)

('qwer', 14460)

('sdf', 5784)

('wer', 13014)

('wert', 8676)

# Set Programs

To create a set by iterating a string of sentence. You are only allowed to create a maximum of two sets, make a frozen set of vowels or random letters, remove the letters from the frozen set that are intersecting with the primary set. Iterate the elements of the altered primary set to output

**PROGRAM**

#string of sentence

a="Caught in a landslide, no escape from reality. Open your eyes, look up to the skies and see"

b=set(a)

print(b)

a=input("enter random letters to remove: ")

a=frozenset(a) #frozenset

print(a)

#the below block to remove the intersecting using “remove” method will

#return a RuntimeError: Set changed size during iteration

#because a new set should have been provided.

##for i in b:

## if i in a:

## b.remove(i)

#So simple difference will solve it

b-=a

for i in b:

print(i)

**OUTPUT**

{'e', 'g', 'a', 'c', 'k', 'p', 'C', 'l', 'h', 's', 't', 'd', 'o', 'O', '.', 'f', ' ', 'y', 'i', 'r', 'm', 'u', ',', 'n'}

enter random letters to remove: asdfg

frozenset({'f', 'g', 's', 'a', 'd'})

e

c

k

p

C

l

h

t

o

O

.

y

i

r

m

u

,

n

>>>

# While Loop

# Fibonacci series

A series of numbers in which each number is the sum of the preceding two numbers

It starts with 0 and goes on as 0,1,1,2,3,5,8,13,21,34,55,89,144,……n.

**PROGRAM**

ff=0 #first of fibonacci series

sf = 1 #second of fibonacci series

fnums=1 #Counter or counting variable

n =int(input("Enter the total fibonacci numbers to be generated: "))

#In fibonacci series, the fibonacci number is the sum of two previous/preceding numbers

while(fnums<= n): #to limit the generated numbers to "n" value

print(ff) #gettin output at the start of each loop

fnums += 1 #Counter gets incremented by 1 for each loop

a = sf #assigning the second fibonacci number to variable "a"

sf = ff #assigning the first fibonacci number to "sf"

ff = a + sf #sum of two preceding numbers

#which becomes the first number of preceding numbers needed to find the next number

**OUTPUT**

Enter the total fibonacci numbers to be generated: 8

0

1

1

2

3

5

8

13

# Iterate a given list and if there is a number divisible by 10, print “There is a number divisible by 10, at [index.no]”.

# a=[23,56,21,23,50,213,23,10,14,100,4135,500,12,14,20]

**PROGRAM**

a=[23,56,21,23,50,213,23,10,14,100,4135,500,12,14,20] #given random list of numbers

i = 0 #counter variable

while i<len(a): #counter is compared to the length of the given list,

#the loop ends when the counter meets the length of the list

if a[i]%10==0: #testing if the iterator value is divisible by 10

print("There is a number divisible by 10, at " + str(i)) #prints the index of the value divisble by 10

#the index number is converted to string so as to concatenate it to the string "There is a number divisible by 10, at ".

i = i+1 #increasing counter by 1 for each loop

**OUTPUT**

There is a number divisible by 10, at 4

There is a number divisible by 10, at 7

There is a number divisible by 10, at 9

There is a number divisible by 10, at 11

There is a number divisible by 10, at 14

# Write a program with while loop to find prime numbers and add them to a list.

A prime number is a natural number greater than 1 that is not a product of two smaller natural numbers, they have only two factors 1 and themselves .

**PROGRAM**

a=[] #Empty list to append prime numbers

i = 1 #Counter and numbers to be checked

n=int(input("To search prime numbers between 0 and "))

while i<= n:

if i> 1: #Defining "prime" with boolean values

#the condtion can be just "i", but i>1 is used to remove the number 1 from the list

prime = True

else:

prime = False

# prime numbers have only two factors 1 and themselves, and not divisible by any other number.

j = 2 # "j" is the whole number, prime numbers start from 2

while j < i: #the loope executes only when the value of "i"-number to be checked is greater than "j"-whole number

if i%j == 0: #when the number to be checked is divisible by the whole number, prime becomes "False" and loop breaks

prime = False

break

j += 1 #if the whole number cannot divide i, it is incremented

if prime: #if prime is not declared False, the number to be checked should be prime and is added to the list

a.append(i)

i += 1 #i is incremented to check the next number

print(a)

**OUTPUT**

To search prime numbers between 0 and 20

[2, 3, 5, 7, 11, 13, 17, 19]

# Factorial of a number

A factorial is the product of the number along with all whole numbers less than that number.

That is, If 5! Is 120, it is because 5! = 5 x 4 x 3 x 2 x 1 = 120.

**PROGRAM**

n=int(input("Enter the number to find its factorial: ")) #n is converted to “int” because, Python reads inputs as strings

num = 1

#In a basic calculation 5! can be calculated as,

# 5 x 4 x 3 x 2 x 1

# step a, 5 x 4 = 20

# step b, 20 x 3 = 60

# step b, 60 x 2 = 120

# step b, 120 x 1 = 120

# the basic step explanation :

#step a: value\_of\_n x decrement\_of\_n = new\_product

#step b: previous\_product x decrement\_of\_n = new\_product

while n >= 1: #the loop ends when the value of "n" reaches zero

num = num \* n #num is the product, which is used as the "previous\_product" in the next loop

n = n - 1 #the value of n is decremented

print(num)

**OUTPUT**

Enter the number to find its factorial: 10

3628800

# 5.To find the Sum and Average of numbers.

Using while loop find the sum and average of numbers 0 to “n”. Where n is the value given by the user.

**PROGRAM**

n=int(input("To find the Sum and average of numbers 0 to ")) #the value of n is converted to "int" since Python reads input in strings

no\_of\_values = n

total\_sum = 0

while n >= 0: #loop ends when number of values becomes zero

total\_sum += n #every loop, the numbers from of "0" to "n" are being added to "total\_sum".

n -= 1 #number of values n is decremented

print("sum =", total\_sum)

average = total\_sum / no\_of\_values

print("Average = ", average)

**OUTPUT**

To find the Sum and average of numbers 0 to 20

sum = 210

Average = 10.5

# Function Programs

## Stone Paper Scissor

A function that returns an argument from three choices, the user enter a data in real-time.

If the argument matches to that of the player, the round is ignored. If the argument meets the win condition, the player wins. If the argument meets the lose condition, the player loses

**PROGRAM**

import random

print("Play . . .")

c=["stone","paper","scissor"] #the ai’s choices are randomly selected from this list

st="stone"

pa="paper"

ci="scissor"

aw="AI WINS" #printed when lose conditions are met

pw="PLAYER WINS" #printed when win conditions are met

def player():

global turn

global roll

global st

global pa

global ci

turn=input("") #Calling input from the user where the player has to type stone/paper/scissor.

if turn in c:

roll=(random.choice(c)) #random function was used using the library random

print(roll)

else: #an else clause is used to find errors or mistakes in the program

print("Stopped due to invalid input")

play()

player()

def play(): #the win and lose conditions are declared using If and elif statements.

if turn==roll:

print("draw")

elif turn==st:

if roll==pa:

print(aw)

elif roll==ci:

print(pw)

else:

print("malfuncs")

elif turn==pa:

if roll==ci:

print(aw)

elif roll==st:

print(pw)

else:

print("malfuncs")

elif turn==ci:

if roll==st:

print(aw)

elif roll==pa:

print(pw)

else:

print("malfuncs")

else: #an else clause is used to find errors or mistakes in the conditions of the program in realtime

print("Check Program")

player()

**OUTPUT**

Play . . .

stone

paper

AI WINS

## Palindrome

Palindrome are words that when all the letters are reversed, becomes the same word.  
eg: pop,

**PROGRAM**

def palin():

a=input("Enter a word: ")

a=list(a)

b=a.copy()

a.reverse()

if a==b:

print("the word is a palindrome")

elif a!=b:

print("the word is not a palindrome")

else:

print("unexpected error")

palin() #calling this function will ask for a word to check whethers it’s a palindrome

**OUTPUT**

Enter a word: radar

the word is a palindrome

## Pangrams

Pangrams are sentences that uses all the letters from the English alphabets.

Example: “The quick brown fox jumps over the lazy dog”

**PROGRAM**

def pang():

c="qwertyuiopasdfghjklzxcvbnm"

a=input("Enter a sentence check whether if its a pangram, ")

a=list(a)

c=list(c)

pangs=True

for i in c:

if i not in a:

pangs=False

if pangs==False:

print("The sentence is not a pangram")

elif pangs==True:

print("The sentence is a pangram")

pang()#calling this function will ask for a sentence to check whethers it’s a pangram

**OUTPUT**

Enter a sentence check whether if its a pangram, the quick brown fox jumps over the lazy dog

The sentence is a pangram

## Program to find LCM, HCF