**1-SWAPING OF TWO NUMBERS USING EXOR**

a=10

b=9

a=a^b

b=a^b

a=a^b

print(a)

print(b)

**OUTPUT:**

9

10

**2-PRINT GIVEN NUMBERS ARE EVEN OR ODD**

n=90

if(n&1==0):

print("even")

else:

print("odd")

**OUTPUT:**

Even

**3-One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed *w* kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.**

**Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.**

**Input**

**The first (and the only) input line contains integer number *w* (1 ≤ *w* ≤ 100) — the weight of the watermelon bought by the boys.**

**Output**

**Print YES, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and NO in the opposite case.**

n=int(input("enter the value"))

if n>=4 and n%2==0:

print("yes")

else:

print("no")

**OUTPUT:**

enter the value8888

yes

**another one:**

w=int(input("enter the numbers"))

if(w%2!=0):

print("no")

else:

x=w/2

if(w/2==0):

print("weight of 2 pieces are:",x,x)

else:

print("weight of 2 pieces are:",x-1,x+1)

**OUTPUT**:

enter the numbers88

weight of 2 pieces are: 44.0 44.0

**5-An elephant decided to visit his friend. It turned out that the elephant's house is located at point 0 and his friend's house is located at point *x*(*x* > 0) of the coordinate line. In one step the elephant can move 1, 2, 3, 4 or 5 positions forward. Determine, what is the minimum number of steps he need to make in order to get to his friend's house.**

**Input**

**The first line of the input contains an integer *x* (1 ≤ *x* ≤ 1 000 000) — The coordinate of the friend's house.**

**Output**

**Print the minimum number of steps that elephant needs to make to get from point 0 to point *x*.**

n=int(input("enter the numbers"))

b=n%5

if b>0:

print((n//5)+1)

else:

print(n/5)

**OUTPUT:**

enter the numbers5

1.0

**ANOTHER ANSWER:**

x=int(input("enter the numbers"))

if(x<=5):

print(1)

elif(x%5==0):

print(x/5)

else:

print(x//5+1)

**OUTPUT:**

enter the numbers66

14

**5-PROGRAM**

n=float(input("enter the number:"))

print(round(n,-2))

**OUTPUT:**

enter the number:846

800.0

**The police department of your city has just started its journey. Initially, they don’t have any manpower. So, they started hiring new recruits in groups.**

**Meanwhile, crimes keeps occurring within the city. One member of the police force can investigate only one crime during his/her lifetime.**

**If there is no police officer free (isn't busy with crime) during the occurrence of a crime, it will go untreated.**

**Given the chronological order of crime occurrences and recruit hirings, find the number of crimes which will go untreated.**

**Input**

**The first line of input will contain an integer *n* (1 ≤ *n* ≤ 105), the number of events. The next line will contain *n* space-separated integers.**

**If the integer is -1 then it means a crime has occurred. Otherwise, the integer will be positive, the number of officers recruited together at that time. No more than 10 officers will be recruited at a time.**

**Output**

**Print a single integer, the number of crimes which will go untreated.**

**Examples**

**input**

**Copy**

**3  
-1 -1 1**

**output**

**Copy**

**2**

**input**

**Copy**

**8  
1 -1 1 -1 -1 1 1 1**

**output**

**Copy**

**1**

**input**

**Copy**

**11  
-1 -1 2 -1 -1 -1 -1 -1 -1 -1 -1**

**output**

**Copy**

**8**

**Note**

**Lets consider the second example:**

1. **Firstly one person is hired.**
2. **Then crime appears, the last hired person will investigate this crime.**
3. **One more person is hired.**
4. **One more crime appears, the last hired person will investigate this crime.**
5. **Crime appears. There is no free policeman at the time, so this crime will go untreated.**
6. **One more person is hired.**
7. **One more person is hired.**
8. **One more person is hired.**

**The answer is one, as one crime (on step 5) will go untreated.**

n=int(input())

availableOfficers = 0

untreatedCrimes = 0

for \_ in range(n):

event = int(input())

if event == -1:

if availableOfficers >0:

aailableOfficers -=1

else:

untreatedCrimes +=1

else:

availableOfficers +=event

print(untreatedCrimes)

**6-SUM OF FIRST N NATURAL NUMBERS**

n=int(input("enter the number:"))

for i in range(n):

print(i)

**OUTPUT:**

enter the number:8

0

1

2

3

4

5

6

7

**FIBONANCE SERIES**

n=int(input("enter the number of terms:"))

a,b=0,1

for i in range(n):

print(a,end=" ")

a, b = b, a + b

**OUTPUT:**

enter the number of terms:5

0 1 1 2 3

**COUNT OF DIGITS**

num = int(input("enter a number:"))

count = 0

while num!=0:

num//=10

count+=1

print("number of digits:",count)

**OUTPUT:**

enter a number:888888

number of digits: 6

**MULTIPLICATION TABLE**

num=int(input("enter a number:"))

print("multiplication table of",num)

for i in range(1,11):

print(num,"x",i,"=",num\*1)

**OUTPUT:**

enter a number:9

multiplication table of 9

9 x 1 = 9

9 x 2 = 9

9 x 3 = 9

9 x 4 = 9

9 x 5 = 9

9 x 6 = 9

9 x 7 = 9

9 x 8 = 9

9 x 9 = 9

9 x 10 = 9

**REVERSE MULTIPLICATION**

num=int(input("enter a number:"))

rev=0

while num>0:

digit=num%10

rev=rev\*10+digit

num=num//10

print("reversed number:",rev)

**OUTPUT:**

enter a number:16887

reversed number: 78861

**BROTUE FORCE METHOD**

n=int(input("enter the value:"))

x=0

for i in range (1,n+1):

x=x^i

print(x)

**OUTPUT:**

enter the value:6

7

**BY USING EXOR**

n=int(input("enter the number:"))

if (n%4==1):

print(1)

if(n%4==2):

print(n+1)

if(n%4==3):

print(0)

if(n%4==0):

print(n)

**OUTPUT:**

enter the number:3

0

**IF L=3 AND R=6 SHOW IT IN THE EXOR IN A BRUTEFORCE METHOD**

r=int(input("enter the value:"))

l=3

for i in range(l,r+1):

l=l^i

print(l)

**OUTPUT:**

enter the value:1

3

**[ORRRRRRRRRRRRRR]**

L=int(input("enter the L"))

R=int(input("enter the R"))

x=0

for i in range(L,R+1):

x^=i

print(x)

**OUTPUT:**

enter the L3

enter the R6

3

7

2

4

**LISTS**

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

for i in range(1,4):

L[i-2]=L[i]

for i in range(0,4):

print(L[i],end=" ")

**OUTPUT:**

3 4 3 4

**ANTHOER ONE**

**##list of operations**

l = [10,11,12,13,14,15,16]

print(l[::2])

print(l[::-1])

print(l[1:6:3])

#print(l[2:])

l = [0,5,10,15,20,25,30]

print(l[:30:5])

res={s for s in [1,2,3,4,5,6,7,8,9,10] if s%2==0}

print(res)

**ACCESSING ELEMENTS TO DICTIONARY**

d={1:'hi','a':123,100:32.4}

print(d[1])

print(d['a'])

print(d[100])

**output:**

hi

123

32.4

**ANOTHER ONE:**

d={1:'hi','a':123,100:32.4}

print(d.get('a'))

**OUTPUT:**

123

**MODIFY AND DELETE A DICTIONARY:**

Only values in dictionary can be changed but keys should be immutable.

d={1:'hi','a':123,100:32.4}

d[1]=1000

print(d)

**output:**

{1: 1000, 'a': 123, 100: 32.4}

Adding an element to the dictionary:

d={1:'hi','a':123,100:32.4}

d[2]='python'

print(d)

**output:**

{1: 'hi', 'a': 123, 100: 32.4, 2: 'python'}

**Get:** it is the conventional method to access a value for a key.

Update: adds dictionary dict2’s key-values pairs to dict.

**Copy:** they copy method returns

**Pop**: removes and returns an element from a dictionary having the given key.

**Popitem:** removes the arbitrary key-value pair from the dictionary and returns is as tuple.

**Clear**: the clear method removes all items from the dictionary.

d={1:'hi','a':123,100:32.4}

d.pop('a')

print(d)

d.popitem()

print(d)

d.clear()

print(d)

**output:**

{1: 'hi', 100: 32.4}

{1: 'hi'}

{}

**Keys:** returns list of dictionary dict’s keys.

**Values:** returns a list of all the values available in a given dictionary.

**Items**: returns a list of dict’s (key, value) tuple pairs.

keys():

d={1:'hi','a':123,100:32.4}

print(d.keys())

**O/P:**

dict\_keys([1, 'a', 100])

values():

d={1:'hi','a':123,100:32.4}

print(d.values())

**O/P:**

dict\_values(['hi', 123, 32.4])

items():

d={1:'hi','a':123,100:32.4}

print(d.items())

**O/P:**

dict\_items([(1, 'hi'), ('a', 123), (100, 32.4)])

**DICTIONARY METHODS:**

**Fromkeys():**

d={'a','b','c','d'}

d=dict.fromkeys(d,10)

print(d)

**OUTPUT:**

{'a': 10, 'd': 10, 'c': 10, 'b': 10}

**Setdefault():**

d={1:'hi','a':123,100:32.4}

print(d.setdefault('a'))

print(d.setdefault('b'))

print(d)

print(d.setdefault('c',200))

print(d)

**OUTPUT:**

123

None

{1: 'hi', 'a': 123, 100: 32.4, 'b': None}

200

{1: 'hi', 'a': 123, 100: 32.4, 'b': None, 'c': 200}

**NESTED DICTIONARY:**

d={1:'hi','a':123,100:{2:'abc','x':452,2.3:120},5.4:'python'}

print(d[100]['x'])

**OUPUT:**

452

**DICTIONARY COMPREHENSION:**

Dictionary comprehension is a method for transforming a one dictionary into another dictionary.

d={'a':1,'b':2,'c':3,'d':4,'e':5}

d1={k:v\*2 for (k,v) in d.items()}

print(d1)

output:

{'a': 2, 'b': 4, 'c': 6, 'd': 8, 'e': 10}

**OUTPUT:**

Code excuted successfully

**PROBLEMS:**

***WRITE A PYTHON PROGRAM TO ADD A KEY TO A DICTIONARY***

import operator

d={}

n=int(input(("no of elements:")))

for i in range(n):

k=input("enter key for first dict:")

v=input("enter its value:")

d[k]=v

key=input()

value=input()

d.update({key:value})

print(d)

**OUTPUT:**

no of elements:3

enter key for first dict:3

enter its value:4

enter key for first dict:6

enter its value:8

enter key for first dict:2

enter its value:82

2

{'3': '4', '6': '8', '2': '8'}

=== Code Execution Successful ===

***PROBLEM STATEMENT-2***

***WRITE A PYTHON PROGRAM TO CHECK WHTHER A GIVEN KEY ALREADY EXISTS IN A DICTIONARY***

***INPUT FORMAT:***

*INPUT A DICTIONARY AND KEY TO CHECK*

***OUTPUT:***

*PRINT WHETHER KEY IS PRESENT IN DICTIONARY*

import operator

d={}

n=int(input(("no of elements:")))

for i in range(n):

k=input("enter key for dict:")

v=input("enter its value:")

d[k]=v

key=input()

if key in d:

print('key is present in the dictionary')

else:

print('key is not present in the dictionary')

**OUTPUT:**

no of elements:3

enter key for dict:2

enter its value:4

enter key for dict:2

enter its value:4

enter key for dict:6

enter its value:7

key is not present in the dictionary

=== Code Execution Successful ===

***PROBLEM SATETMENT-3***

***write a python program to generate and print a dictionary that contains a number (between 1 and n) in the form (x and x square of x).***

n=int(input("input a number"))

d=dict()

for x in range(1,n+1):

d[x]=x\*x

print(d)

***OUTPUT:***

input a number4

{1: 1, 2: 4, 3: 9, 4: 16}

***PROBLEM STATEMENT-4***

***WRITE A PYTHON PROGRAM TO MERGE TWO PYTHON DICTIONARIES***

***INPUT FORMAT:***

*READ TWO DICTIONARIES*

***OUTPUT FORMAT:***

*PRINT THE MERGED DICTIONARIES*

d={}

d1={}

n=int(input("no of elements:"))

for i in range(n):

k=input("enter key for dict:")

v=input("enter its value:")

d[k]=v

for i in range(n):

k=input("enter key for dict:")

v=input("enter its value:")

d1[k]=v

d2=d.copy()

d2.update(d1)

print(d2)

**OUTPUT:**

no of elements:2

enter key for dict:8

enter its value:9

enter key for dict:6

enter its value:8

enter key for dict:4

enter its value:2

enter key for dict:9

enter its value:6

{'8': '9', '6': '8', '4': '2', '9': '6'}

***PROBLEM STATEMENT-5***

***WRITE A PYTHON PROGRAM TO REMOVE A KEY FROM DICTIONARY***

***INPUT FORMAT:***

*READ DICTIONARY AND KEY TO REMOVE*

***OUPUT FORMAT:***

*PRINT THE UPDATED DICTIONARY*

d={}

n=int(input("no of elements:"))

for i in range(n):

k=input("enter the key for dict:")

v=input("enter its value:")

d[k]=v

key=int(input())

if key in d:

del d[key]

print(d)

***OUTPUT:***

no of elements:2

enter the key for dict:62

enter its value:

enter the key for dict:

enter its value:8

8

{'2': '', '6': '8'}

***PROBLEM STATEMENT-6***

***WRITE A PYTHON PROGRAM TO MAP TWO LISTS INTO A DICTIONARY***

***INPUT FORMAT:***

*READ TWO LISTS*

***OUPUT FORMAT:***

*PRINT THE DICTIONARY*

keys=input('enter the values').split()

values=input('enter the values').split()

d=dict(zip(keys,values))

print(d)

**OUTPUT:**

enter the values1 12 3 4 4

enter the values4 32

{'1': '4', '12': '3'}

***ANOTHER ONE***

l1=[1,2,3]

l2=[5,6,7]

d={}

for i in range(0,3):

key=l1[i]

value=l2[i]

d[key]=value

print(d)

***OUPUT:***

{1: 5, 2: 6, 3: 7}

***PROBLEM STATEMENT-7***

***WRITE A PYTHON PROGRAAM TO COMBINE TWO DICTIONARY ADDING VALUES FOR COMMON KEYS***

***INPUT FORMAT:***

*READ TWO DICTIONARIES*

***OUTPUT FORMAT:***

*PRINT THE NEW DICTIONARY*

#from collection import Counter

d={}

d1={}

n=int(input("no of elements:"))

for i in range(n):

k=input("enter key for firsst dict:")

v=input("enter its value:")

d[k]=v

for i in range(n):

k=input("enter key for second dict:")

v=input("enter its value:")

d2=Counter(d)+Counter(d1)

print(d2)

***OUTPUT:***

no of elements:2

enter key for firsst dict:4

enter its value:5

enter key for firsst dict:3

enter its value:1

enter key for second dict:4

enter its value:2

enter key for second dict:4

enter its value:3

5ERROR!

Traceback (most recent call last):

File "<main.py>", line 14, in <module>

***ANOTHER ONE***

dict1={'a':10,'b':20,'c':30}

dict2={'b':5,'c':15,'d':25}

combined\_dict={}

for key in dict1:

if key in dict2:

combined\_dict[key]=dict1[key]+dict2[key]

else:

combined\_dict[key]=dict1[key]

for key in dict2:

if key not in combined\_dict:

combined\_dict[key]=dict2[key]

print("Combined Dictionary:",combined\_dict)

***OUTPUT:***

Combined Dictionary: {'a': 10, 'b': 25, 'c': 45, 'd': 25}

**STRINGS:**

**Strings**: strings are immutable, hence elements of a string cannot be changed once it has been assigned.

* Only new strings can be reassigned to the
* We can directly delete a string using del keyword.
* String methods and function:
* String.ascii\_letters: Concatation of the ascii\_lowercase and ascii\_uppercase constants.
* String.ascii\_lowercase: Concatenation of lowercase letters.
* String.ascii\_uppercase: concatenation of uppercase letters.
* String.punctuations: ASCII characters having punctuation characters.
* [3:54 pm, 31/5/2024] Akshaya ECE SRU(LE): String.digits: digit in strings.
* String.hexdigits: hexadigit in strings.
* String.octdigits: octadigit in a sting
* [3:55 pm, 31/5/2024] Akshaya ECE SRU(LE): import string
* print(string.digits)
* print(string.hexdigits)
* print(string.octdigits)
* output:
* 0123456789
* 0123456789abcdefABCDEF
* 01234567

**String.endswitch**: returns true if a string ends with the given suffix otherwise returns false

**String.startswith:** returns true if a string with the given prefix otherwise returns false.

**Replace**: returns a copy of he sting where all occurrences of a substring is replaced with another substring.

print('hello'.endswith('o'))

print('hello'.startswith('h'))

print('hello'.replace('l','t'))

True

True

Hetto

**String.isdigit**: returns “true” if all characters in the string are digits, otherwise, it returns “False”.

**String.isalpha:** returns “true” if all characters in the string are alphabets, otherwise it returns “false”.

**String.isdecimal:** returns true if all characters in a string are decimal

print('100'.isdigit())

print('abc'.isalpha())

print('72'.isdecimal())

True

True

True.

**String.isalnum:** returns true if all the characters in a given string are alphanumeric

**String.istitle**: returns true if the string is a titlecased string.

print('ab12'.isalnum())

print('Hello World'.istitle())

True

True

**String,partition:** splits the string at the first occurrence of the separator and returns tuple.

**String.index:** returns the position of the first occurrence of substring in a string.

**String.rindex:** returns the highest index of the substring inside the string if the substring is found.

**String.splitlines:** returns a lsit of lines in the string.

print('hello'.upper())

print('HELLo'.lower())

print('hElLo'.swapcase())

HELLO

hello

HeLlO

**PROGRAM:**

str="pranathi"

print(str.index('a'))

print(str.rindex('a'))

**OUTPUT:**

str="pranathi"

print(str.index('a'))

print(str.rindex('a'))

**STRING METHODS AND FUNCTIONS:**

**String.capitalize**: return the word with its first character capitalized.

**String.find:** return the lowest index in a sub string.

**String.rfinf**: find the highest index.

**String.count**: returns the number of non- overlapping occurrences of substring sub in string.

**PROGRAM:**

print('python programming'.capitalize())

print('python cython'.find('th'))

print('python cython'.rfind('th'))

print('python cython'.count('th'))

Python programming

**OUTPUT:**

2

9

2

**Len**: returns the length of the string.

**Max**: returns the highest alphabetical character in a string

**Min**: returns the minimum alphabetical character in a string.

**Slicing of a string:**

**S[:]:** prints all the elements from the string.

**S[2:]:** prints all the elements from the string starting from index 2.

**S[:5]:** prints all the elements from the string till the ending index 4.

**S[2:6]:** prints the elements from the string from index 2 till index 5.

**S[-4:2]:** prints the elements from the string from index -4 till index -3.

**Program:**

s='pranathi'

print(s[:])

print(s[2:])

print(s[:6])

print(s[2:4])

print(s[-3:-5])

**OUTPUT:**

pranathi

anathi

pranat

an

**PROGRAM:**

**S[::2]:** prints all the elements from the string with the step 2 from the beginning.

**S[::-1]:** prints all the elements from the string with step 1 from the last.

**PROGRAM:**

s='python'

print(s[::])

print(s[1::4])

**OUTPUT:**

python

yn

**PROGRAM:**

str1="This is 2020"

str2="2020"

print(str1.isdigit())

print(str2.isdigit())

output:

False

True

**OUTPUT:**

str1="This is 2020"

str2="2020"

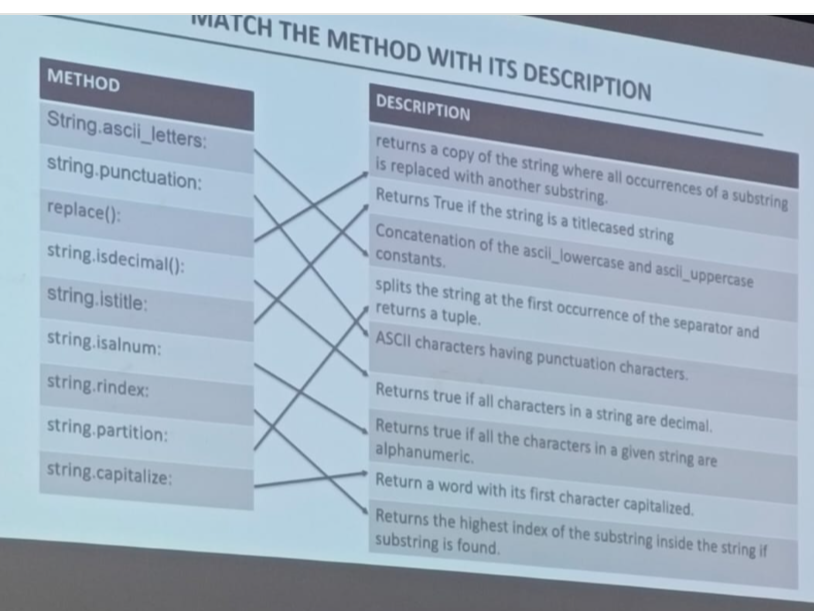
print(str1.isdigit())

print(str2.isdigit())

output:

False

True

****

**PROBLEM SATEMENT-1**

***WRITE A PYTHON PROGRAM TO CALCULATE THE LENGTH OF THE STRING***

***CODE:***

def len1(str1):

count=0

for char in str1:

count+=1

return count

str1=input("enter the value")

print(len(str1))

***OUTPUT:***

enter the valuecoder123

8

***PROBLEM SATEMENT-2***

***WRITE A PYTHON SCRIPT THAT TAKES INPUT FROM THR USER AND DISPLAYS THE INPUT BACK IN UPPER CASE AND LOWER CASE***

***INPUT FORMAT:***

*READ A STRING*

***OUTPUT FORMAT:***

*PRINT THE STRING IN BOTH UPPER AND LOWER CASE*

***CODE:***

s=input("enter the word:")

print(s.upper())

print(s.lower())

***OUTPUT:***

enter the word:PRNATHI

PRNATHI

Pranathi

***ANOTHER ONE***

***CODE:***

s1="my car is bmW"

s2="bmW"

s3=s1.replace(s2,s2.upper())

print(s3)

***OUTPUT:***

my car is BMW

***ANOTHER ONE***

***PROBLEM SATEMENT-4***

***WRITE A PYTHON PROGRAM TO REMOVE ALL CONSECUTIVE DUPLICATES FROM A GIVEN STRING***

***INPUT FORMAT:***

*READ A STRING*

***OUTPUT FORMAT:***

*PRINT THE STRING AFTER REMOVING ALL CONSECUTIVE DUPLICATES*

str=input()

result=str[0]

for i in range(1,len(str)):

if tr[i]!=result[-1]:

result+=str[i]

print(result)

"""s=input()

spaces="""

result=""

for char in s:

if char ==" ":

***OUTPUT:***

***PROBLEM STATEMENT-5***

***WRITE A PYTHON PROGRAM TO MOVE ALL SPACES TO THE FRONT OF A GIVEN IN SINGLE TRAVERSAL***

***INPUTFORMAT:***

*READ A STRING*

***OUTPUT FORMAT:***

*PRINT THE STRING AFTER ELIMINATING SPACES*

***CODE:***

s=input("enter the value:")

spaces=" "

result=" "

for char in s:

if char ==" ":

spaces+=char

else:

result+=char

final\_string=spaces+result

print(final\_string)

***OUTPUT:***

enter the value:k a d p i k o n d a p r a n a t h i r e d d y

kadpikondapranathireddy

***PROBLEM SATEMENT-6***

***WRITE A PYTHON PROGRAM TO CREATE A STRING FROM TWO GIVE STRINGS CONCATENATIN UNCOMMON CHARACTERS OF THE SAID STRINGS***

***INPUT FORMAT:***

*READ A STRING*

***OUTPUT FORMAT:***

*PRINT THE NEW STRING*

***CODE:***

str1="Akshaya"

str2="Sravanthi"

set1=set(str1)

set2=set(str2)

uncommon\_chars=set2.symmetric\_difference(set1)

print(uncommon\_chars)

uncommon\_string=''.join(uncommon\_chars)

print(uncommon\_string)

***OUPUT:***

{'k', 't', 'v', 'S', 'y', 'r', 's', 'i', 'A', 'n'}

ktvSyrsiAn

***ANOTHER ONE:***

s1=input("enter the value1:")

s2=input("enter the value2:")

uncommon\_chars="".join(set(s1)^set(s2))

print(uncommon\_chars)

***OUTPUT:***

enter the value1:pranathi

enter the value2:kadipikonda

hotkrd

***PROBLEM SATEMENT-7***

***WRITE A PYTHON PROGRAM TO FIND THE MAXMIMUM OCCURING CHARACTER IN A GIVEN STRING***

***INPUT FORMAT:***

*READ A STRING*

***OUTPUT FORMAT:***

*PRINT THE MAXIMUM OCCURING CHARACTERS*

***CODE:***

s=input("enter the value:")

max\_char = max(s, key=s.count)

print(f"Maximum occurring character: {max\_char}")

**output:**

enter the value:Treat

Maximum occurring character: T

**PYTHON FUNCTION DECLEARTION**:

**Types of functions in python:**

Built-in library function: These are standard function in Python that are available to use.

User-defined function: We can create a own functions based on our requirements.

**Print the numbers as prime numbers are not:**

def check\_prime():

num=int(input("enter a number:"))

if num>1:

for i in range(2,int(num\*\*0.5)+1):

if num%i==0:

print(num,"is not a prime number")

break

else:

print(num,"is a prime number")

else:

print(num,"is not a prime number")

check\_prime()

**OUTPUT:**

enter a number:27

27 is not a prime number

**ARGUMENTS:**

def myfun(x, y=50):

print("x:",x)

print("y:",y)

myfun(10)

**OUTPUT:**

x: 10

y: 50

**PROGRAM:**

def student(firstname, lastname):

print(firstname, lastname)

student(firstname='akshaya', lastname='pallamaina')

student(lastname='sravanthi',firstname='baikanai')

**OUTPUT:**

akshaya pallamaina

baikanai sravanthi

**PROGRAM:**

def nameAge(name, age):

print("Hi, I am ", name)

print("My age is",age)

print("case-1:")

print("pranathi",20)

print("/ncase-2:")

nameAge(20, "pranathi")

**OUTPUT:**

case-1:

pranathi 20

/ncase-2:

Hi, I am 20

My age is pranathi

**PYTHON ARGS:**

def my\_sum(\*args):

result=0

for x in args:

result+=x

return result

print(my\_sum(1,2,3))

6

**PROGRAM:**

def concatenate(\*\*words):

result=""

for arg in words.values():

result+=arg

return result

**OUTPUT:**

realpythonisgreat!

**RECURSSION**

**PROGRAM:**

#recursive function

def recursive\_factorial(n):

if n == 1:

return n

else:

return n\* recursive\_factorial(n-1)

#user input

num=6

#check if the input is valid or not

if num<0:

print("invalid input! please enter a positive number")

elif num==0:

print("factorial of number 0 is 1")

else:

print("factorial of number",num,"=",recursive\_factorial(num))

**OUTPUT:**

factorial of number 6 = 720

**WRITE A PROGRAM FOR THE FACTORIAL OF 1ST 5 NATURAL NUMBERS:**

def factorial(n):

if n == 1:

return 1

else:

return n \* factorial(n - 1)

result = factorial(5)

print(result)

**OUTPUT:**

120

**FACTORIAL REVERSE:**

def factorial\_expression(n):

components = []

factorial = 1

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

components.append(str(i))

factorial \*= i

expression = '\*'.join(components)

print(f"The factorial of {n} is: {expression}")

print(f"Which equals: {factorial}")

factorial\_expression(5)

**OUTPUT:**

The factorial of 5 is: 5\*4\*3\*2\*1

Which equals: 120

**HEAD FUNCTION:**

Factorial\_expression(5)

def fun(n):

if(n>0):

print(n)

fun(n-1)

n=int(input("enter the value:"))

fun(n)

**OUTPUT:**

enter the value:7

7

6

5

4

3

2

1

**TAIL FUNCTION:**

def fun(n):

if(n>0):

fun(n-1)

print(n)

n=int(input("enter the value:"))

fun(n)

**OUTPUT:**

enter the value:5

1

2

3

4

5

**FIBONACCI SERIES USING RECURSION:**

**TREE RECURSION:**

def fibonacci(n):

if n<=1:

return n

else:

return fibonacci(n-1)+fibonacci(n-2)

num\_terms=int(input("enter the number of terms:"))

for i in range(num\_terms):

print(f"fibonacci({i}))={fibonacci(i)}")

**OUTPUT:**

enter the number of terms:4

fibonacci(0))=0

fibonacci(1))=1

fibonacci(2))=1

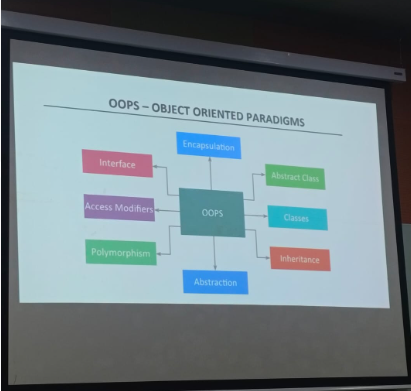
fibonacci(3))=2

***PYTHON AND DATA ANALYSIS***

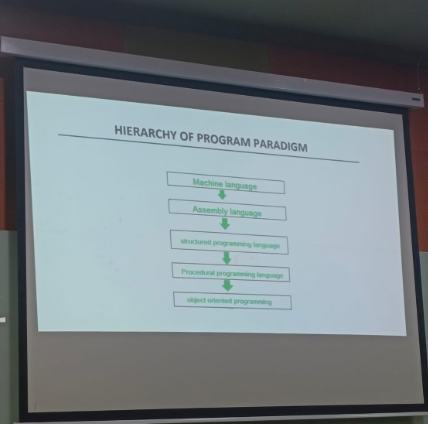
***DATA ANALYSIS:***

* *Python is the leading language of choice for many data scientists.*
* *Python has grown in popularity ,within this field due to its excellent libraries including:*
* *NumPy and pandas and its superb libraries for data visualization like Matplotl*

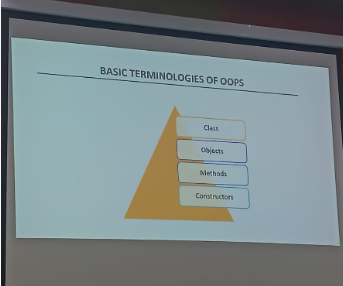
***OOPS-OBJECT ORIENTED PARADIGMS***

**

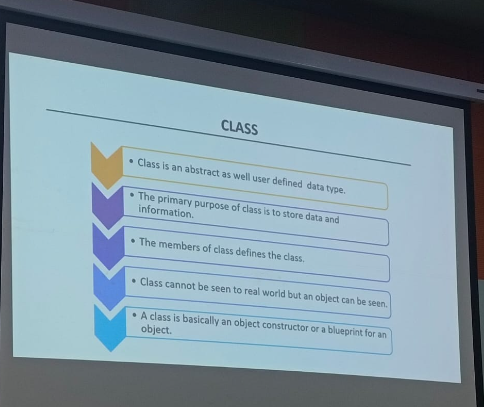
***HIERARCHU PF PROGRAM PARADIGAM***

**

***BASIC TERMINOLOGIES OF OOPS***

**

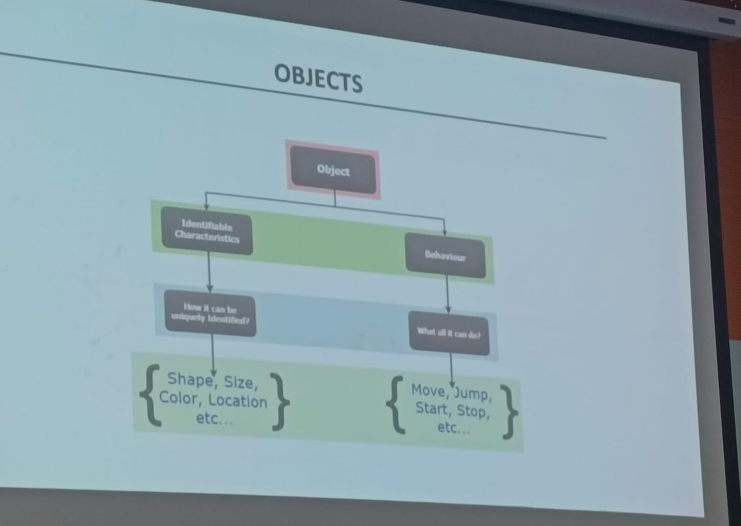
***CLASS***

******

* *Class is an abstract as well user defined data type*
* *The primary purpose of class is to store data and information*
* *The members of class defined the class*
* *Class cannot be seen to real world but an object can be seen*
* *A class is basically an object constructor or blueprint for an object*

***CREATING AN INSTANCE OF THE CLASS***

*A class needs to be*

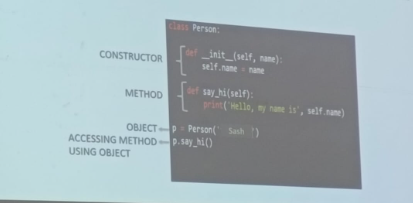
**

***OBJECTS***

* *The implementation of class is an object*
* *Objects are an encapsulation of variables and functions into a single entity.*
* *Objects get their variables and functions from class.*

***OBJECT CREATION USING\_INIT\_METHOD***

*The\_\_init \_\_method is similar to constructors in c++and java.It runs as soon as an object of class is instatiated.The method is useful to do any initialization you want to do with your object.*

*****PROGRAM:***

*class Person:*

*def\_\_init\_\_(my,name):*

*my.name=name*

*def say\_hi(my):*

*print('hello,my name is',my.name)*

*p=Person('pranathi')*

*p.say\_hi()*

***OUTPUT:***

*hello,my name is pranathi*

***what is the output?***

*the thing required to create an object is the* ***constructor***

***PROGRAM:***

*class test:*

*def \_\_init\_\_(self,a="hello world"):*

*self.a=a*

*def display(self):*

*print(self.a)*

*obj=test()*

*obj.display()*

***OUPUT:***

*Hello world*

***OOPS CONCEPTS***

* *Inheritance*
* *Polymorphism*
* *Data encapsulation*
* *Data abstraction*
* *Exception handling*

***INHERITANCE:***

* *Inheritance allows us to define a class that inherits*

***TYPES OF INHERITANCE:***

1. *SINGLE*
2. *MULTIPLE*
3. *HERACHICAL*
4. *4.*

***5…***

***SINGLE INHERITANCE***

*class Parent():*

*def first(self):*

*print('first function')*

*class Child(Parent):*

*def second(self):*

*print('second function')*

*ob=child()*

*ob.first()*

*ob.second()*

***OUTPUT:***

*irst function*

*second function*

***MULTIPLE INHERITANCE***

*When a child class inherits from more than one parent class,it is multiple inheritance*

***PROGRAM:***

*class parent:*

*def func1(self):*

*print("this is function 1")*

*class parent2:*

*def func2(self):*

*print("this is function 2")*

*class Child(parent, parent2):*

*def func3(self):*

*print("this is function 3")*

*ob = Child()*

*ob.func1()*

*ob.func2()*

*ob.func3()*

***OUTPUT:***

*this is function 1*

*this is function 2*

*this is function 3*

***HIERARCHICAL INHERITANCE***

*Hi*

***HYBRID INHERITANCE***

*Hybrid inheritance involves multiple inheritance taking place in single program*

***PROGRAM:***

*class Parent:*

*def func1(self):*

*print("this is function 1")*

*class Child(Parent):*

*def func2(self):*

*print("this is function 2")*

*class Child(Parent):*

*def func3(self):*

*print("this is function 3")*

*class Child(Parent):*

*def func4(self):*

*print("this is function 4")*

*ob = Child3()*

*ob.func1()*

***SUPER METHOD***

***PROGRAM:***

*class Parent:*

*def func1(self):*

*print("this is function 1")*

*class Child(Parent):*

*def func2(self):*

*super().func1()*

*print("this is function 2")*

*ob = Child*

*ob.func2()*

***OUTPUT:***

*This is function*

***you are required to create a simple zoo management system using python inheritance. the system should manage different types of animals and their unique behaviors***

***PROGRAM:***

*class Animal:*

*def \_\_init\_\_(self, name, species, age):*

*self.name = name*

*self.species = species*

*self.age = age*

*def make\_sound(self):*

*pass*

*def move(self):*

*pass*

*def info(self):*

*return f"Name: {self.name}, Species: {self.species}, Age: {self.age}"*

*class Mammal(Animal):*

*def \_\_init\_\_(self, name, species, age, fur\_color):*

*super().\_\_init\_\_(name, species, age)*

*self.fur\_color = fur\_color*

*def make\_sound(self):*

*return "Mammal sound"*

*def move(self):*

*return "Walk or run"*

*def info(self):*

*return super().info() + f", Fur Color: {self.fur\_color}"*

*class Bird(Animal):*

*def \_\_init\_\_(self, name, species, age, wing\_span):*

*super().\_\_init\_\_(name, species, age)*

*self.wing\_span = wing\_span*

*def make\_sound(self):*

*return "Bird sound"*

*def move(self):*

*return "Fly"*

*def info(self):*

*return super().info() + f", Wing Span: {self.wing\_span}"*

*class Fish(Animal):*

*def \_\_init\_\_(self, name, species, age, water\_type):*

*super().\_\_init\_\_(name, species, age)*

*self.water\_type = water\_type*

*def make\_sound(self):*

*return "Fish sound"*

*def move(self):*

*return "Swim"*

*def info(self):*

*return super().info() + f", Water Type: {self.water\_type}"*

*lion = Mammal("Leo", "Lion", 5, "Golden")*

*eagle = Bird("Eddie", "Eagle", 3, "2 meters")*

*shark = Fish("Sammy", "Shark", 8, "Saltwater")*

*animals = [lion, eagle, shark]*

*for animal in animals:*

*print(animal.info())*

*print(f"Sound: {animal.make\_sound()}")*

*print(f"Movement: {animal.move()}")*

*print("-" \* 20)*

***OUTPUT:***

*Name: Leo, Species: Lion, Age: 5, Fur Color: Golden*

*Sound: Mammal sound*

*Movement: Walk or run*

*--------------------*

*Name: Eddie, Species: Eagle, Age: 3, Wing Span: 2 meters*

*Sound: Bird sound*

*Movement: Fly*

*--------------------*

*Name: Sammy, Species: Shark, Age: 8, Water Type: Saltwater*

*Sound: Fish sound*

*Movement: Swim*

***PROGRAM:***

*class person:*

*def \_\_init\_\_(self,name,age):*

*self.name=name*

*self.age=age*

*def speak(self):*

*print(f"hi my name is{self.name}and my language is{self.lang}")*

*pranathi=Person(name:"Pranathi",age:20)*

*srujana=Person(name:"srujana",age:21)*

*p1.speak("telugu")*

***PROGRAM:***

*class Animal:*

*def \_\_init\_\_(self, name, age):*

*self.name = name*

*self.age = age*

*def description(self):*

*print(f"Hi, my name is {self.name} and I am {self.age} years old.")*

*class Dog(Animal):*

*def \_\_init\_\_(self, name, age, breed):*

*super().\_\_init\_\_(name, age)*

*self.breed = breed*

*def description(self):*

*print(f"This is {self.name}, a cute dog who is {self.age} years old. His breed is {self.breed}.")*

*dog = Dog(name="Yuvraj", age=5, breed="Pomeranian")*

*animal = Animal(name="Leo", age=3)*

*dog.description()*

*animal.description()*

***OUTPUT:***

*This is Yuvraj, a cute dog who is 5 years old. His breed is Pomeranian.*

*Hi, my name is Leo and I am 3 years old.*

***PROGRAM:***

*arr = input("Enter the list of numbers separated by spaces:").split()*

*arr = [int(x) for x in arr]*

*target = int(input("Enter the target value to search:"))*

*found = False*

*for i in range(len(arr)):*

*if arr[i] == target:*

*print(f"Target{target} found at index{i}.")*

*found = True*

*break*

*if not found:*

*print(f"Target{target} not found in the array.")*

***OUTPUT:***

*Enter the list of numbers separated by spaces:1 2 3 4*

*Enter the target value to search:3*

*Target3 found at index2.*

***PROGRAM:***

*arr = [int(x) for x in input("Enter the sorted list of numbers separated by spaces:").split()]*

*target = int(input("Enter the target value to search:"))*

*start, end = 0,len(arr)-1*

*found = False*

*while start <=end:*

*mid = (start + end)//2*

*if arr[mid] == target:*

*print(f"Target{target}found at index{mid}.")*

*found = True*

*break*

*elif arr[mid] < target:*

*start = mid + 1*

*else:*

*end = mid - 1*

*if not found:*

*print(f"Target {target} not found in the array.")*

***OUTPUT:***

*Enter the sorted list of numbers separated by spaces:1 2 34 4*

*Enter the target value to search:3*

*Target 3 not found in the array.*

***PROGRAM:***

*def authenticate():*

*password ="pranathi123"*

*attempts = 3*

*while attempts > 0:*

*user\_input = input("Enter the password:")*

*if user\_input == password:*

*print("welcome!")*

*break*

*else:*

*attempts -= 1*

*if attempts > 0:*

*print(f"wrong password! you have{attempts} attempts left.")*

*else:*

*print("account blocked.")*

*break*

*authenticate()*

***OUTPUT:***

*Enter the password:1234*

*wrong password! you have2 attempts left.*

*Enter the password:shashi123*

*wrong password! you have1 attempts left.*

*Enter the password:pranathi123*

*welcome!*

***ACCESS***

*class Student:*

*def \_init\_(self, name ,roll\_number , password):*

*self.name = name #public attribute*

*self.\_roll\_number = roll\_number #protected attribute*

*self .\_\_password = password #private attribute*

*def display\_details(self):*

*print("Name:", self.name)*

*print("Roll number:", self.\_roll\_number)*

*#private attribute accessed within the class*

*print("Password:", self.\_\_password)*

***OUPUT:***

*Code excuted successfully*

***authenticator.py***

*from abc import ABC, abstractmethod*

*class Authenticator(ABC):*

*def \_init\_(self, password):*

*self.\_password = password*

*self.\_attempts = 3*

*@abstractmethod*

*def verify\_password(self, user\_input):*

*pass*

*def authenticate(self):*

*while self.\_attempts > 0:*

***simple\_authenticator.py:***

*from authenticator import Authenticator*

*class SimpleAuthenticator(Authenticator):*

*def verify\_password(self, user\_input):*

*return user\_input == self.\_password*

*from simple\_authenticator import SimpleAuthenticator*

***main.py:***

*def main():*

*password = "password123"*

*authenticator = SimpleAuthenticator(password)*

*authenticator.authenticate()*

*if \_name\_ == "\_main\_":*

*main()*

***SORTING***

***BUBBLE SORT:***

*def bubble\_sort(arr):*

*n= len(arr)*

*#traverse through all elements in the array*

*for i in range(n):*

*#last i elements are already sorted,so we dont need to check them*

*for j in range(0, n-1 -1):*

*#swap if the elements found is greter than the next time*

*if arr[j] > arr[j+1]:*

*arr[j],arr[j+1]=arr[j+1],arr[j]*

*#example usage:*

*my\_list = [64,34,25,12,22,11,90]*

*print("Original List:",my\_list)*

*bubble\_sort(my\_list)*

*print("Sorted List:",my\_list)*

***OUTPUT:***

*Original List: [64, 34, 25, 12, 22, 11, 90]*

*Sorted List: [11, 12, 22, 25, 34, 64, 90]*

***SELECTION SORT:***

*def selection\_sort(arr):*

*n=len(arr)*

*#traverse through all elements in the array*

*for i in range(n):*

*#find the minimum element in the unsorted part*

*min\_index=i*

*for j in range(i + 1,n):*

*if arr[j] < arr[min\_index]:*

*min\_index = j*

*#swap the found minimum element with the first element of the unsorted part*

*arr[i], arr[min\_index]=arr[min\_index],arr[i]*

*#example usage:*

*my\_list = [64,34,25,12,22,11,90]*

*print("Original List:", my\_list)*

*selection\_sort(my\_list)*

*print("Sorted List:",my\_list)*

***OUTPUT:***

*Original List: [64, 34, 25, 12, 22, 11, 90]*

*Sorted List: [11, 12, 22, 25, 34, 64, 90]*

***INSERTION SORT:***

*def insertion\_sort(arr):*

*n=len(arr)*

*#traverse through all elements in the array starting from second element*

*for i in range(1,n):*

*key = arr[i]*

*#move elements of arr[0..i-1]that are greater than key to one postion ahead of their current position*

*j = i-1*

*while j >=0 and key < arr[j]:*

*arr[j+1]=arr[j]*

*j-=1*

*arr[j+1]=key*

*#example usage:*

*my\_list=[64,34,25,12,22,11,90]*

*print("Original List:",my\_list)*

*insertion\_sort(my\_list)*

*print("Sorted list:",my\_list)*

***OUTPUT:***

*Original List: [64, 34, 25, 12, 22, 11, 90]*

*Sorted list: [11, 12, 22, 25, 34, 64, 90]*

***MERGE SORT:***

*def merge\_sort(arr):*

*if len(arr)>1:*

*#divide the array into two halves*

*mid = len(arr)//2*

*left\_half = arr[:mid]*

*right\_half = arr[mid:]*

*#recursive call on each half*

*merge\_sort(left\_half)*

*merge\_sort(right\_half)*

*def merge(arr,left,right):*

*i=j=k=0*

*#compare and merge the elements from left and right halves*

*while i<len(left) and j<leb(right):*

*if left[i]<right[j]:*

*arr[k]=left[i]*

*i+=1*

*else:*

*arr[k]=right[j]*

*j+=1*

*k+=1*

*#if there are any remanining elements in left or right halves,add the to the merged array*

*while i < len(left):*

*arr[k] = left[i]*

*i += 1*

*k += 1*

*while j < len(right):*

*arr[k] = right[j]*

*j+=1*

*k+=1*

*#example usage:*

*my\_list=[64,334,25,12,22,11,90]*

*print("Original List:", my\_list)*

*merge\_sort(my\_list)*

*print("Sorted List:",my\_list)*

***OUTPUT:***

*Original List: [64, 334, 25, 12, 22, 11, 90]*

*Sorted List: [64, 334, 25, 12, 22, 11, 90]*

***POLYMORPHISM:***

***OVER RIDING:***

*class Animal:*

*def speak(self):*

*return"Animal makes a sound"*

*class Dog(Animal):*

*def speak(self):*

*return "Woof"*

*class Cat(Animal):*

*def speak(self):*

*return "Meow"*

*class Cow(Animal):*

*pass*

*dog = Dog()*

*cat = Cat()*

*cow = Cow()*

*print("Dog says:",dog.speak())*

*print("Cat says:",cat.speak())*

*print("Cow says:",cow.speak())*

***OUTPUT:***

*Dog says: Woof*

*Cat says: Meow*

*Cow says: Animal makes a sound*