#

a=int(input())

if a==1:

print("sunday")

elif a==2:

print("monday")

elif a==3:

print("tuesday")

elif a==4:

print("wednesday")

elif a==5:

print("thursday")

elif a==6:

print("friday")

elif a==7:

print("saturday")

else:

print("no day")

# SET

s={10,50,30}

s1={300,500}

s.update(s1)

print(s)

s={10,40,80,50}

s.remove(40)#raise an error when data is not available

s.discard(50)#does not raise an error

print(s)

s={10,50,30,60,80}

s.pop()#remove random number

print(s)

s.clear()

print(s)

del s

s1={220,440,660}

s2={220,440,6,8}

s1.intersection\_update(s2)#numbers which are same in both sets it display the same numbers

print(s1)

s={10,200,300}

s1={2,4,10,8}

s.symmetric\_difference\_update(s1)#it display the numbers which are different in both sets

print(s)

s1={3,46,2}

s2={2,3,6,8}

s3=s1.intersection(s2)

print(s3)

s={100,400,700}

s1={20,40,700}

s2=s.symmetric\_difference(s1)

print(s2)

#STRING

text="divyasara234"

#upper=text.upper()#stored in another variable

print("upper=",text.upper())

print("lower=",text.lower())

print("capitalize=",text.capitalize())#first letter changes to capital

print("casefold=",text.casefold())

print("center=",text.center(20))

print("count=",text.count("divya"))#count the number of times the text appear

print("encode=",text.encode())

print("endswith=",text.endswith("a"))#check the last letter if true or false

print("startswith=",text.startswith("i"))#check the first letter if true or false

print("find=",text.find("divya sara"))#true=0& false=-1

print("index=",text.index("v"))#index starts from 0

print("isalnum=",text.isalnum())#include alphabet and number without giving any space

print("isalpha=",text.isalpha())#give only alphabet without space

print("isascii=",text.isascii())

print("isdecimal=",text.isdecimal())

print("isdigit=",text.isdigit())

#TUPLE

t=()

print(t)

t=("Vijay","Trisha","Samantha","Anushka")#create a tuple

print(t)

t1=(10,)#tuple with single value

print(t1)

print(t)

t=("Vijay","Trisha","Samantha","Anushka")

L=list(t)#t--name of tuple,convert the tuple to list

L.append("Sai Pallavi")

L.append("Dulquer")

t=tuple(L)#convert back to tuple

print(t)

x=("Keerthi Suresh",)

t+=x

print(t)

L=list(t)#convert to list

L.remove("Samantha")

t=tuple(L)#convert to tuple

print(t)

t=("Vijay","Trisha","Samantha","Anushka")

(V,T,S,A)=t

print(V)

print(T)

print(S)

#OOPS

class student:

s\_name="Divya"

s\_dept="AI"

s=student() #object creation -> objectname=classname()

print(f"Name={s.s\_name}\nDepartment={s.s\_dept}")

#without a method(Function)

class student:

s\_name=""

s\_deppt=""

s1=student() # object 1

s2=student() # object 2

s1.s\_name="Divya"

s1.s\_dept="AI"

s1.s\_name="Anu"

s1.s\_dept="AI"

print(f"Name={s1.s\_name}\nDepartment={s1.s\_dept}")

print("\nAssign values based on object2")

#with a method(Function)

class student:

name="divya"

dept="AI"

def display(self):

print(f"Name={self.name}\nDepartment={self.dept}")

s=student() #object creation

s.display() #method call

#

class student:

# constructor ---assign values to the class attributes

def \_\_init\_\_(self,name=""):

self.name=name

def display(self):

print("Student detail")

print("Name=",self.name)

s\_name=input()#first line of execution

s=student(s\_name)

print("Outside the class")

s.display()

class student:

ID="" #class variable

#constructor----assign values to the attributes

def \_\_init\_\_(self,name=""):

self.name=name #instance variable

def getData(self):

self.ID=input("Enter ID")

def display(self):

print("Student detail")

print("Name=",self.name)

print("ID=",self.ID)

s\_name=input("Enter the name")

s=student(s\_name)

s.getData()

s.display()

#non-parameterized

class student:

def \_\_init\_\_(self): #constructor

self.name="Anu"

self.dept="AI"

def display(self): #display method

print(f"Name={self.name}\nDepartment={self.dept}")

stu=student() #object creation

stu.display() #method call

class library:

def \_\_init\_\_(self):

self.author\_name="Ruskin Bond"

self.book\_name="The Blue Umbrella"

def getBookDetails(self):

self.book\_details=input()

def display(self):

print("Book Details")

print(f"Author Name={self.author\_name}")

print(f"Book Name={self.book\_name}")

li=library()

li.display()

#paramterised constructor

class student:

def \_\_init\_\_(self,name,dept):

self.name=name

self.dept=dept

def display(self):

print(f"Name={self.name}\nDepartment={self.dept}")

stu=student("Divya","AI") #object creation

stu.display()

#parameterised constructor(default):

class student:

def \_\_init\_\_(self,dept,name="Divya",age=18):

self.age=age

self.name=name

self.dept=dept

def display(self):

print(f"Name={self.name}\nDepartment={self.dept}\nAge={self.age}")

stu=student("AI") #object creation

stu.display()

#

class student:

def \_\_init\_\_(self,dept,name="Divya",age=18):

self.age=age

self.name=name

self.dept=dept

def display(self):

print(f"Name={self.name}\nDepartment={self.dept}\nAge={self.age}")

def \_\_del\_\_(self): #destructor method

print("Object is destroyed")

stu=student("AI") #object creation

stu.display()

del stu

#single inheritance

class student: #parent class

def display(self):

print("Base class - Parent")

class student\_derived(student):

def show(self):

print("Derived class - Child")

s=student\_derived()

s.display()

s.show()

class person: #parent class

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

self.name=name

self.age=age

def display(self):

print(f"Name={self.name}\nAge={self.age}")

class student(person):

def \_\_init\_\_(self,name,age,stu\_id,stu\_dept):

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

self.stu\_dept=stu\_dept

self.stu\_id=stu\_id

def printDetails(self):

self.display()

#DATE & TIME

import datetime

#current date and time

current\_datetime=datetime.datetime.today()

print(f"current datetime={current\_datetime}")

#current date only

current\_date=datetime.date.today()

print("current date:",current\_date)

#current time only

current\_time=datetime.datetime.today().time()

print("current time:",current\_time)

#create specific date(year,month,day)

s\_date=datetime.date(2024,10,7)

print("Specific date:",s\_date)

#create specific time(hour,minute,second)

s\_time=datetime.time(5,10,45)

print("Specific time:",s\_time)

#create specific date and time(year,month,day,hour,minute,second)

s\_datetime=datetime.datetime(2024,10,7,5,10,45)

print("Specific datetime:",s\_datetime)

#formatting date and time into a string

f\_date=current\_datetime.strftime("%Y-%m-%d%H:%M:%S")

print("Formatted date and time:",f\_date)

#parsing a date string into a datetime object

date\_string="2024-10-09 09:05:10"

parsed\_date=datetime.datetime.strptime(date\_string,"%Y-%m-%d %H:%M:%S")

print("Parsed datetime object:",parsed\_date)

#create a timedelta of 10 days

ten\_days=datetime.timedelta(days=10)

#subtract 10 days from the current date

date\_10\_days\_ago=current\_date- ten\_days

print("Date 10 days ago:",date\_10\_days\_ago)

#add 10 days to the current date

date\_10\_days\_later=current\_date+ten\_days

print("Date 10 days later:",date\_10\_days\_later)

#calculate the difference between two dates

date1=datetime.date(2024,10,7)

date2=datetime.date(2023,10,7)

difference=date1-date2

print("Difference between two dates:",difference.days,"days")

#different timezone

import pytz

#Get current time in UTC

current\_utc=datetime.datetime.today(pytz.utc)

print("current UTC time:",current\_utc)

#convert UTC time to a specific time zone(eg.US/Eastern)

eastern=pytz.timezone('US/Eastern')

eastern\_time=current\_utc.astimezone(eastern)

print("Eastern time:",eastern\_time)"""

import calendar

year = 2024

month = 10

print(calendar.month(year,month))

print(calendar.calendar(year))

# Print the calendar for the given year

print(calendar.calendar(year))

# Check if the year is a leap year

is\_leap = calendar.isleap(year)

print(f"{year} is a leap year: {is\_leap}")

# Get the first weekday and the number of days in the month

first\_weekday, num\_days = calendar.monthrange(year, month)

print(f"First weekday: {first\_weekday}, Number of days: {num\_days}")

for day in calendar.Calendar().itermonthdays(year, month):

print(day)

import calendar

# Create a TextCalendar instance

text\_cal = calendar.TextCalendar(calendar.SUNDAY) # SUNDAY as the starting day of the week

# Generate a plain-text calendar for a specific month

year = 2024

month = 10

plain\_text\_cal = text\_cal.formatmonth(year, month)

# Print the plain-text calendar

print(plain\_text\_cal)

from datetime import datetime, timedelta

# Get the current date and time

now = datetime.now()

print(f"Current date and time: {now}")

# Specify how many days/hours/minutes in the past you want to go

days\_in\_past = 5

hours\_in\_past = 3

minutes\_in\_past = 30

# Subtract time using timedelta

past\_time = now - timedelta(days=days\_in\_past, hours=hours\_in\_past, minutes=minutes\_in\_past)

print(f"Past date and time (5 days, 3 hours, 30 minutes ago): {past\_time}")

#Example : Change the first letter to caps..

# Example of ca