DATETIME

```
import datetime
import re
#print my birthdate
#parameters in following order year,month,day
#bday=datetime.date(2000, 8,3)
#print(bday)

#print todays date
#today=datetime.date.today()
#print(today)

#days since birth
#int()=converts string to int
#dates can be subtracted
#month and day boundaries are already set
#date=input("Enter Date in MM-DD-YYYY: ").strip()
#match=re.findall(r"(\d+)", date)
#month=int(match[0])
#day=int(match[1])
#year=int(match[2])
#date1=datetime.date(year, month, day)
#today=datetime.date.today()
#days=today-date1
#print(days)

#print indiviual components
#if documentation does not use brackets then they dont have to be written
#date=datetime.date.today()
#year=date.year
#print(year)

#days till birthday
#bday=input("Enter Your Birthday: Month and Day")
#date=datetime.date.today()
#match=re.findall(r"\d+", bday)
#month=int(match[0])
#day=int(match[1])
#year=date.year
#party=datetime.date(year, month,day)
#total=party-date
#print(total)
```

SET8 Problem 1

import datetime

```
import re
import inflect
p = inflect.engine()
#days since birth
#int()=converts string to int
#dates can be subtracted
#month and day boundaries are already set
#once calculation is complete datetime converts to timedelta
#timedelta can return weeks days hours and seconds
date=input("Enter Date in MM-DD-YYYY: ").strip()
match=re.findall(r"(\d+)", date)
month=int(match[0])
day=int(match[1])
year=int(match[2])
date1=datetime.date(year, month, day)
today=datetime.date.today()
days=today-date1
st=days.days
total=p.number_to_words(st)
print(total)
```

CLASSMETHOD AND INHERITANCE

```
#when using a class method use cls instead of
#use inheritance for any shared elements between classes
# def __add__(self,other)-use to add values together with plus sign
#works with other operators as well
#import random
#class Hat:
#    houses = ["Cookie","Burger","Sandwich","Juice"]
#
#    @classmethod
#    def sort(cls, name):
#        print(name, "wants a" , random.choice(cls.houses))

#Hat.sort("Harry")

class Wizard:
    def __init__(self,name):
        if not name:
            raise ValueError("Missing Name")
        self.name=name

class Student(Wizard):
    def __init__(self,name,house):
```

```
        super().__init__(name)
        self.house = house

class Pro(Wizard):
    def __init__(self,name,subject):
        super().__init__(name)
        self.subject=subjec
    def __str__(self):
        return f"{self.name}"


wizard = Wizard("Albus")
student = Student("Harry","Slyterin")
pro=Pro("Severus","Defense")
print(pro)
```

EXTRA NOTES

```
#global variable-variable that the entire class can use
#bal=0
#def main():
#    print(bal)
#    deposit(30)
#    print(bal)
#def deposit(n):
#    global bal
#    bal+=n
#constants-varible that is set to a certain value that will most likely not change
#cat = 3
#for i in range(cat):
#   print("*")
"""
other ways of reading notes
"""
```