READ NUMBER OF LINES FROM FILE

```
import sys

def main():
    y=count()
#number in brackets represents space in list
    f=sys.argv[-1].split('.')[-1]

    if len(sys.argv)>2:
        print("Not Valid")
        sys.exit()
    else:
        print(f'{y}')
    if f!=("py"):
        sys.exit("Not A File")
    else:
         count()



def count():
     try:
         with open(sys.argv[1], "r") as file:
            lines = file.read().splitlines()
            totalCount = len(lines)
            whitespace = 0
            comments = 0

         for line in lines:
            lineCheck = line.rstrip().strip().split('\n')
            for x in lineCheck:
                if len(x) < 1:
                    whitespace += 1
                elif len(x) > 0 and x.startswith('#'):
                    comments += 1

            finalCount = totalCount - whitespace - comments
            print(finalCount)

     except FileNotFoundError:
           print("File Not Found")
           sys.exit()

main()
```

PIZZA TABULATE

```
import csv
import sys
from tabulate import tabulate

try:
    x=sys.argv[1].split(".")[-1]
    if len(sys.argv) != 2:
        print("not enough")
        sys.exit()
    elif x != "csv":
        print("File Not found")
        sys.exit()
    elif sys.argv[1]=="sicilian.csv":
        with open("sicilian.csv","r") as file:
            reader=csv.reader(file)
            print(tabulate(reader,headers='firstrow',showindex='always',tablefmt='grid'))
    elif sys.argv[1]=="regular.csv":
        with open("regular.csv","r") as file:
            reader=csv.reader(file)
            print(tabulate(reader,headers='firstrow',showindex='always',tablefmt='grid'))
except FileNotFoundError:
        print("File Not Found")
        sys.exit()
```

READ FROM ONE FILE AND WRITE TO ANOTHER

```
#Read file with csv,split it create new rows,attach it to new file,refer to notes
import sys
import csv
def main():
    if len(sys.argv) != 3:
        print("Enter 3 arguments")
    else:
        write()

def write():
    three=[]
    try:
        x=sys.argv[1].split(".")[1]
        y=sys.argv[1].split(".")[-1]
        if x != "csv":
            print("Enter Valid File")
            sys.exit
        elif y != "csv":
            print("Enter a valid file type")
        else:
            with open(sys.argv[1],"r") as file:
#read data with dictreader
                reader=csv.DictReader(file)
                for row in reader:
#state each row as it appears
                    names = row['name'].split(", ")
                    first=names[1]
                    last=names[0]
                    house=row["house"]
                    three.append({"first": first,"last": last,"house": house})


            with open(sys.argv[2],"w")as file:
                  writer=csv.DictWriter(file)
                  writer.writerows(three)
```

REGULAR EXPRESSIONS

```
 import re

email = input("What is your Email").strip()
#.=any charater can be inputed
#+ one or more repetitions or inputs
#/ = breaks repetions
#r makes string ignore backslashes\
#raise value error caen be done to break code
#^,$ use these two to makes sure input matches string format
#[]= add specific parameters,what can be inputed
#[^@]=anything but @ is allowed
#[a-zA-Z_]=range of characters allowed
#\w=any word charater or number or underscore
#(com|edu|org)=either a or b
#.lower()=lowercase all imput
#flags=third parameter that has many functions

if re.search(r"^.+@.+\.edu$", email, re.IGNORECASE):
    print("Valid")
else:
    print("Invalid")
```

RE.search pt2

```
import re

email = input("What is your Email").strip()
#()=group a set of inputs together
#()?=group can be present one or not at all
#re.match=automatically matches strings


if re.search(r"^\w+@(\w+\.)?\w+\edu$", email, re.IGNORECASE):
    print("Valid")
else:
    print("Invalid")
```

FORMAT OPTIONS

```
import re
name = input("What is your name?:").strip()
# in statement looks for object in input
#if "," in name:
#    name=name.split(",")
#    first=name[1]
#    last=name[0]
#print(f"Hello,{first} {last}")
#:= assign value and ask a boolean question about it
if matches:=re.search(r"^(.+), *(.+)$", name):
    name=matches.group(2) + " " + matches.group(1)

print(f"Hello, {name}")
```

RE SUB

```
import re
url=input("What is the url:").strip()
#re.sub = use to replace stirng from input
#(.+)=a match
#(?: )=dont capture this match
name=re.sub(r"^(https?://)?(www\.)?twitter\.com/", "", url)
print (f"{name}")
```

NUMB3RS
import re

```
def main():
    z=input(("IPv4 Address: ").strip())
    print(validate(z))
```

```
def validate(ip):
    z = re.search(r"^(.+)\.(.+)\.(.+)\.(.+)$", ip)
    if z:
        number1=z.group(1)
        number2=z.group(2)
        number3=z.group(3)
        number4=z.group(4)

        if int(number1) > 255:
            return False
        else:
            if int(number2) > 255:
                return False
            else:
                if int(number3) > 255:
                    return False
                else:
                    if int(number4) > 255:
                        return False
                    else:
                        return True


    if __name__ == "__main__":
        main()

main()
```

SUB URL YOUTUBE

```
import re
#? means precedding value may or may not be present
def main():
    url=input("What is the url:").strip()
    y=re.search(r"^(.+)?(https?://)?(www\.)?youtube\.com/(.+)?$",url)
    if y:
        print(is_valid(url))
    else:
        print("None")

def is_valid(url):
    name=re.sub(r"^(.+)?(https?://)?(www\.)?youtube\.com/(.+)?",
"https://youtu.be/xvFZjo5PgG0", url)
```

```
    ner=f"{name}"
    return ner




if __name__ == "__main__":
        main()


import re
import sys
#returns specific letter in phrase
#str=input("Enter Phrase")
#match=re.findall(r"a",str)
#print(match)

#returns gruop of letters
#str=input("Enter Phrase")
#match=re.findall(r"um",str)
#print(match)

#returns adverbs(word ends with ly)
#only use ^ and $ for strings
#\b breaks is used to break a string(use for finish with)
#\w+ means any letter one or more times
#str=input("Enter Phrase")
#match=re.findall(r'\w+ly\b',str)
#print(match)

#returns prefix(word starts with im)
#use single quotations for regular expressions
#str=input("Enter Phrase")
#match=re.findall(r'im\w+',str)
#print(match)

#returns words not letters
#\b before the string means begins with
#str=input("Enter Phrase")
#match=re.findall(r'\bum\b',str)
#print(match)

#returns find all length
#str=input("Enter Phrase")
```

```
#match=re.findall(r'\bum\b',str)
#print(len(match))

#use sys to get input
#sys.argv=system argument added in terminal
#can only be used on simple cases
#regex=sys.argv[1]
#input1=sys.argv[2]
#if len(sys.argv)!=3:
#    sys.exit("Not enough arguments")
#else:
#match=re.findall(regex,input1)
#    print(match)
#return remainder of word
#()= a match that is captured and returned
#d+=any digit use one or more times
#? means may or may not be used
#str=input("Enter Phrase")
#match=re.findall(r'im(\w+?\d+)',str)
#print(match)

# find string
#re.IGNORECASE=ignores wheter a word is uppercase or lowercase
#str=input("Enter Phrase")
#match=re.findall(r'i am a (\w+)',str,re.IGNORECASE)
#print(match)
```

Validators

```
import validators
#max=limit
#can be used with dates

def main():
    print(validate(input("What's your email address? ")))


def validate(s):
    if validators.email(s) == True:
        return f"Valid"
    else:
        return f"Invalid"
```

```
if __name__ == "__main__":
    main()
```

OOP

```
#init sets class method
#str returns values as str
#set getter with @property
#set setter with __name__.setter
#setters and getters are used to error check
class Pet:
    def __init__(self,name,animal):
        self.name=name
        self.animal=animal
    def __str__(self):
        return f'{self.name} is a {self.animal}'
    @property
    def name(self):
        return self._name

    @name.setter
    def name(self, name):
        if not name:
            raise ValueError("Missing Name")
        self._name = name

    @property
    def animal(self):
        return self._animal
    @animal.setter
    def animal(self, animal):
        if animal not in ["dog","cat","bird","turtle"]:
            raise TypeError("Not valid animal")
        self._animal=animal

def main():
    pet=get_pet()
    print(pet)

def get_pet():
    name=input("What is the name of your pet")
    animal=input("What is the specices of the animal").lower()
    return Pet(name, animal)
```

```
if __name__=="__main__":
    main()
```
OOP part 2

```
#only set setters and getters for material in init
class Bank:
    def __init__(self,balance=0,limit=1000):
        self.balance=balance
        self.limit=limit

    def __str__(self):
        return f'Your current balnce is {self.balance}'
    def deposit(self, amount):
        if amount > self.limit:
            raise ValueError("Over the limit")
        self.balance=self.balance+amount

    def withdraw(self, amount):
        if amount <= 0 :
            raise ValueError("Not possible")
        self.balance=self.balance-amount

    @property
    def balance(self):
        return self._balance

    @balance.setter
    def balance(self,balance):
        if balance <0:
            raise ValueError("Not Possible")
        self._balance=balance

    @property
    def limit(self):
        return self._limit
    @limit.setter
    def limit(self, limit):
        self._limit=limit

def main():
    bank=Bank()
    print(bank)
    while True:
            atm=input("Would you like to withdraw(w) deposit(d) or Exit(e)").lower()
            if atm == "d":
                amount=int(input("How much would you want to deposit"))
                bank.deposit(amount)
                print(bank)
            elif atm=="w":
                amount=int(input("How much would you want to Withdraw"))
                bank.withdraw(amount)
                print(bank)
            elif atm=="e":
                break
            else:
                print("Please enter a valid letter")
                pass



if __name__=="__main__":
    main()
```