```
import logging

logging.basicConfig(filename="Lists.log", level=logging.DEBUG,
format="%(asctime)s %(levelname)s %(message)s")


# List_s Class contains lists previous tasks with logging and exception
handling.
class List_s:
    logging.info("Accessing List_s class")

    def extract_list(self, l):
        """ Extract list entity from collections"""
        logging.info("We are in the extract_list function")
        try:
            self.l = l
            for i in self.l:
                if type(i) == list:
                    print(i)
        except Exception as e:
            logging.exception(e, " Please enter iterable collections like :-
lists, tuples, set, dictionary")

    def extract_ineuron(self, m):
        """ Extract 'ineuron' string data from collections"""
        logging.info("Inside function Extract_ineuron ")

        try:
            self.m = m
            for i in self.m:
                if type(i) != dict:
                    for j in i:
                        if j == 'ineuron':
                            logging.info(j)
                            return j
                else:
                    for j in i:
                        if i[j] == 'ineuron':
                            logging.info(i[j])
                            return i[j]
        except Exception as e:
            logging.exception(e)

    def flat_list(self, l):
        """ To create flat list from nested collection"""
        logging.info("Inside flat_list function")
        try:
            self.l = l
            self.p = []
            for i in l:
                if type(i) != dict:
                    logging.info("data :- %s", i)
                    self.p.extend(i)
                else:
```

```
                    logging.info("values inside dict %s", i)
                    self.p.extend(i.keys())
                    self.p.extend(i.values())
            return self.p
            logging.info("Output %s", p)
        except Exception as e:
            logging.exception(e)
            return e

    def print_prime(self):
        """  To create list of prime numbers between 1 to 1000"""
        logging.info("Inside print_prime function")
        try:
            self.l = []
            for i in range(1, 1000):
                c = 0
                for j in range(2, i):
                    if i % j == 0:
                        c = 1
                if c != 1:
                    self.l.append(i)
            logging.info("Output %s",self.l)
            return self.l
        except Exception as e:
            logging.exception(e)
            return e


l = [[1, 2, 3, 4], (2, 3, 4, 5, 6), (3, 4, 5, 6, 7), set([23, 4, 5, 45, 4, 4, 5,
45, 45, 4, 5]),
     {'k1': "sudh", 'k2': "ineuron", 'k3': "kumar", 3: 6, 7: 8}, ["ineuron",
"data science"]]

list_var = List_s()

# print(list_var.flat_list(l))
# print(list_var.extract_list(l))
print(list_var.extract_ineuron(l))
print(list_var.print_prime())
```