Problem Statement Solution

SUBMITTED BY : SANJANA DOSS ; CB.EN.U4CCE19049

VIP 22

CISCO - AICTE

VIRTUAL INTERNSHIP PROGRAM

Prog Ess - Python Problem Statement

Python Problem Statement

Task 1: Ask the user to input 10 ipv4 addresses.

Task 2: Check if the addresses are valid ipv4 addresses.

Task 3: Convert the ipv4 addresses which are in decimal format to Binary, Octal and Hexadecimal format

For conversion use functions (inbuilt or library)

Task 4:Create a list which will hold the addresses. [Decimal, Binary, Octal and Hexadecimal]

Task 5: Transfer the contents of the list to a file named conversion.txt

Task 6: Print the following output on the screen

What is an IP address...

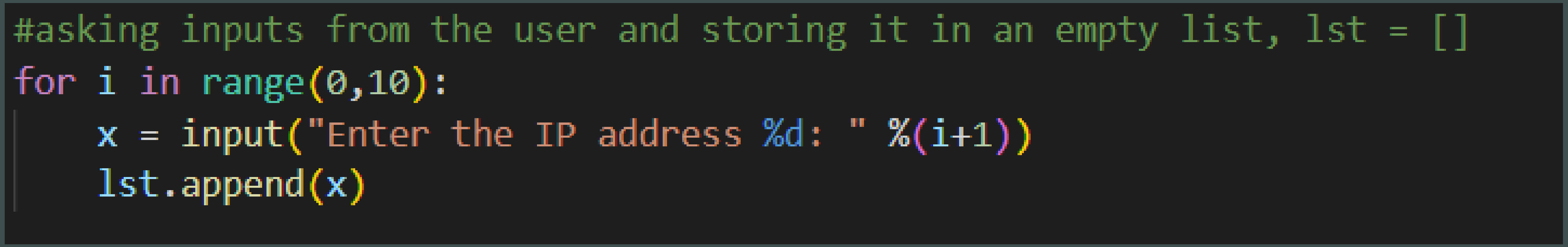
A device on the internet or a local network can be identified by its IP address, which is a unique address. The rules defining the format of data delivered over the internet or a local network are known as "Internet Protocol," or IP.

A four-digit IP address is a series of integers separated by periods; an example address would be 192.158.1.38. The range of each number in the set is 0 to 255. Therefore, the complete IP addressing range is 0.0.0.0 to 255.255.255.255.

IP addresses are not random. The Internet Assigned Numbers Authority (IANA), a part of the Internet Corporation for Assigned Names and Numbers, produces and distributes them mathematically (ICANN).

T1

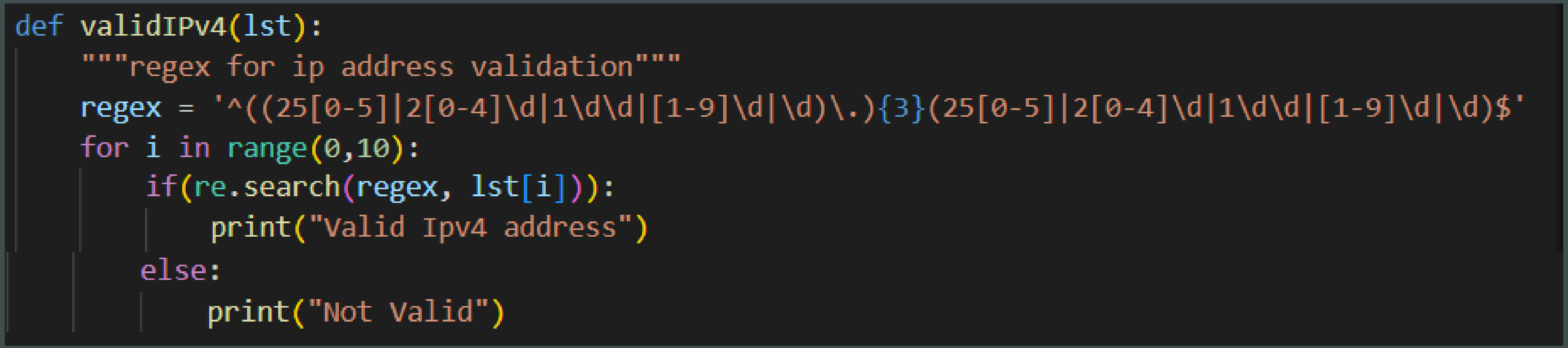
TASK 1 : ask the user 10 ipv4 addresses



```
#asking inputs from the user and storing it in an empty list, lst = []
for i in range(0,10):
    x = input("Enter the IP address %d: " %(i+1))
    lst.append(x)
```

Using an for loop with range (0,10) we ask the user to enter an IP address so we obtain 10 inputs. We use an empty list, lst to store the inputted IP addresses by the .append() function.

TASK 2 : check if the addresses are valid ipv4 addresses



```python
def validIPv4(lst):
    """regex for ip address validation"""
    regex = '^((25[0-5]|2[0-4]\d|1\d\d|[1-9]\d|\d)\.){3}(25[0-5]|2[0-4]\d|1\d\d|[1-9]\d|\d)$'
    for i in range(0,10):
        if(re.search(regex, lst[i])):
            print("Valid Ipv4 address")
        else:
            print("Not Valid")
```

T2

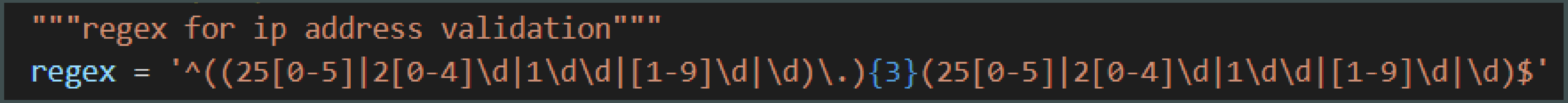
rules of ipv4:

- no leading 0s
- the range of numbers should be 0 to 255
- it should have exactly 4 cells
- should have an INT in each cell

validation of the ipv4 addresses using RegEx (built in package) :

A string of characters that creates a search pattern is known as a regex, or regular expression.
RegEx can be used to determine whether a string includes a given search pattern.
Regular Expressions can be used in Python by using the built-in re module.

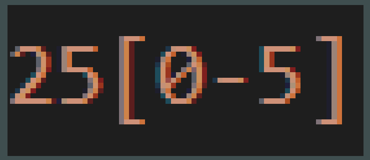
TASK 2 : check if the addresses are valid ipv4 addresses



```
"""regex for ip address validation"""
regex = '^((25[0-5]|2[0-4]\d|1\d\d|[1-9]\d|\d)\.){3}(25[0-5]|2[0-4]\d|1\d\d|[1-9]\d|\d)$'
```

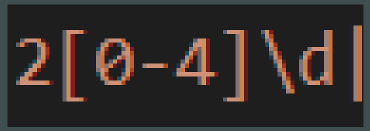
my approach to validation of the ipv4 addresses using RegEx (built in package) :

- ^ character represents "Starts with"
- $ represents "Ends With"
- \. represents the string "."
- To represent each cell in ipv4, I split the cases as one-digit, two-digit, three-digit numbers.
- | represents or condition
- \d = [1-9] digit range



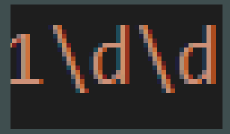
`25[0-5]`

- represents the range 255 - 250



`2[0-4]\d|`

- represents the range 249-200



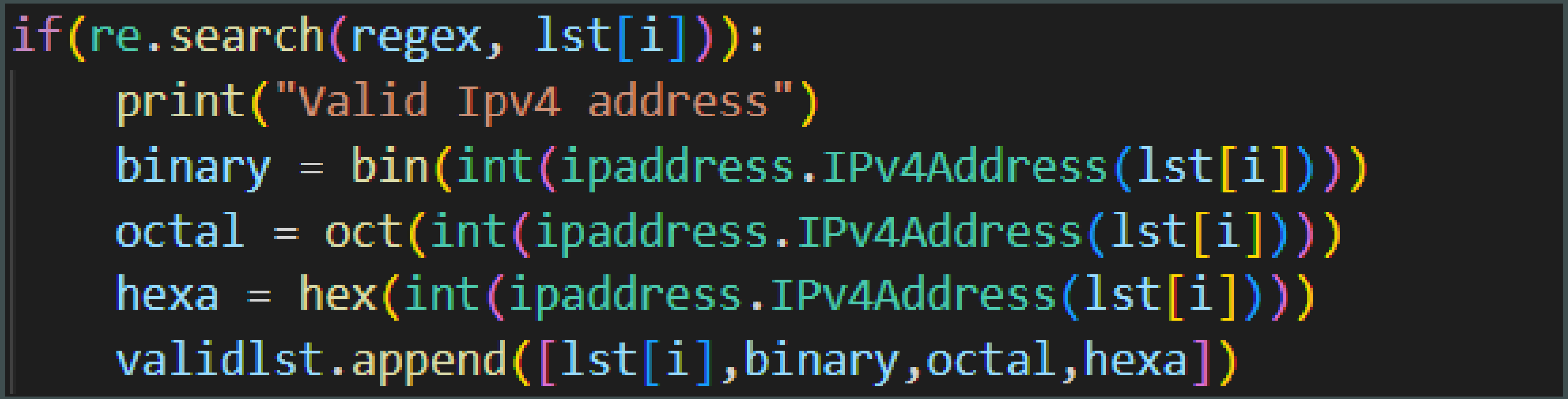
`1\d\d`

- represents the range 199-100

- {3} represents the repetition of expressions. Since there are only 3 dots separating the cells of IPv4. We repeat the digit sequence at the end again.

T3

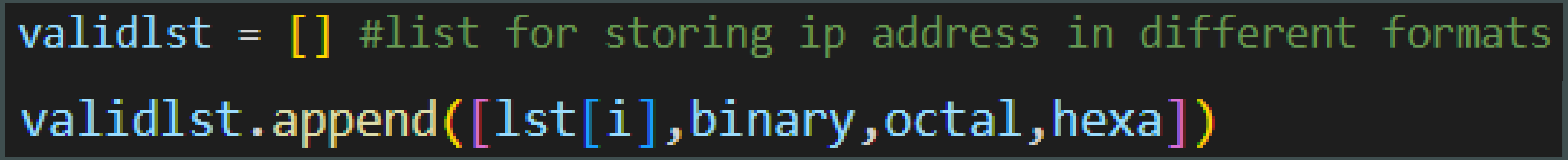
TASK 3 : convert the ipv4 addresses which are in decimal format to Binary, Octal and Hexadecimal format



```
if(re.search(regex, lst[i])):
    print("Valid Ipv4 address")
    binary = bin(int(ipaddress.IPv4Address(lst[i])))
    octal = oct(int(ipaddress.IPv4Address(lst[i])))
    hexa = hex(int(ipaddress.IPv4Address(lst[i])))
    validlst.append([lst[i],binary,octal,hexa])
```

- I have decided to use ipaddress, the IPv4/IPv6 manipulation library.
- ipaddress.ip_address(address) : Returns an IPv4Address object
- If the IP address is valid, then we convert the default string inputted address to integers using int()
- Binary, Octal and Hexa conversions are done by built in python functions, bin(), oct(), hex()

TASK 4 : Create a list which will hold the addresses. [Decimal, Binary, Octal and Hexadecimal]



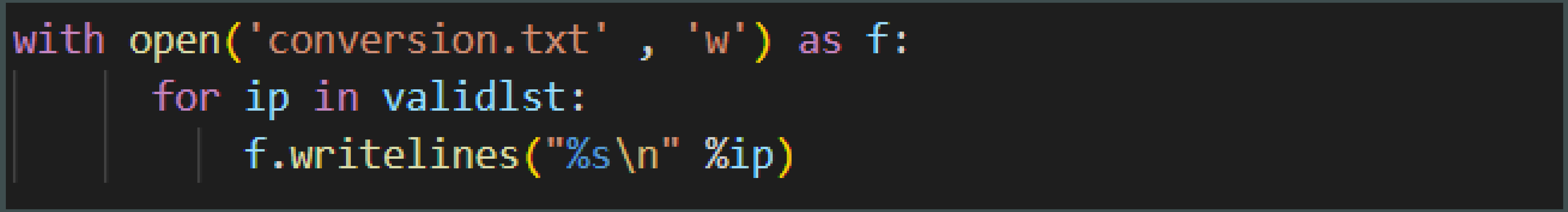
```python
validlst = [] #list for storing ip address in different formats

validlst.append([lst[i],binary,octal,hexa])
```

- validlst holds the [decimal, binary, octal and hexadecimal] whenever an IPv4 address is validated

T4

T5

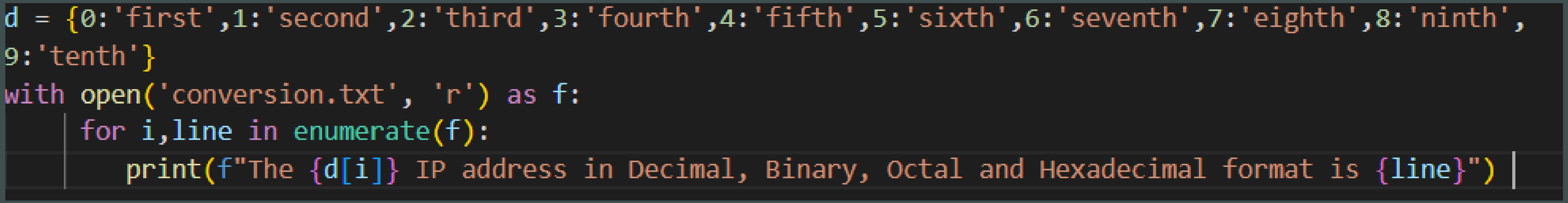
TASK 5 : transfer the contents of the list to a file named conversion.txt



```
with open('conversion.txt' , 'w') as f:
    for ip in validlst:
        f.writelines("%s\n" %ip)
```

- I have utilized python's file handling
- TO open the file, use the built-in open() function and to write to it we use 'w'
- For each element in the validlst list, we write to the conversion.txt
- The writelines() method writes the items of a list to the file.

TASK 6 : print the following output on the screen



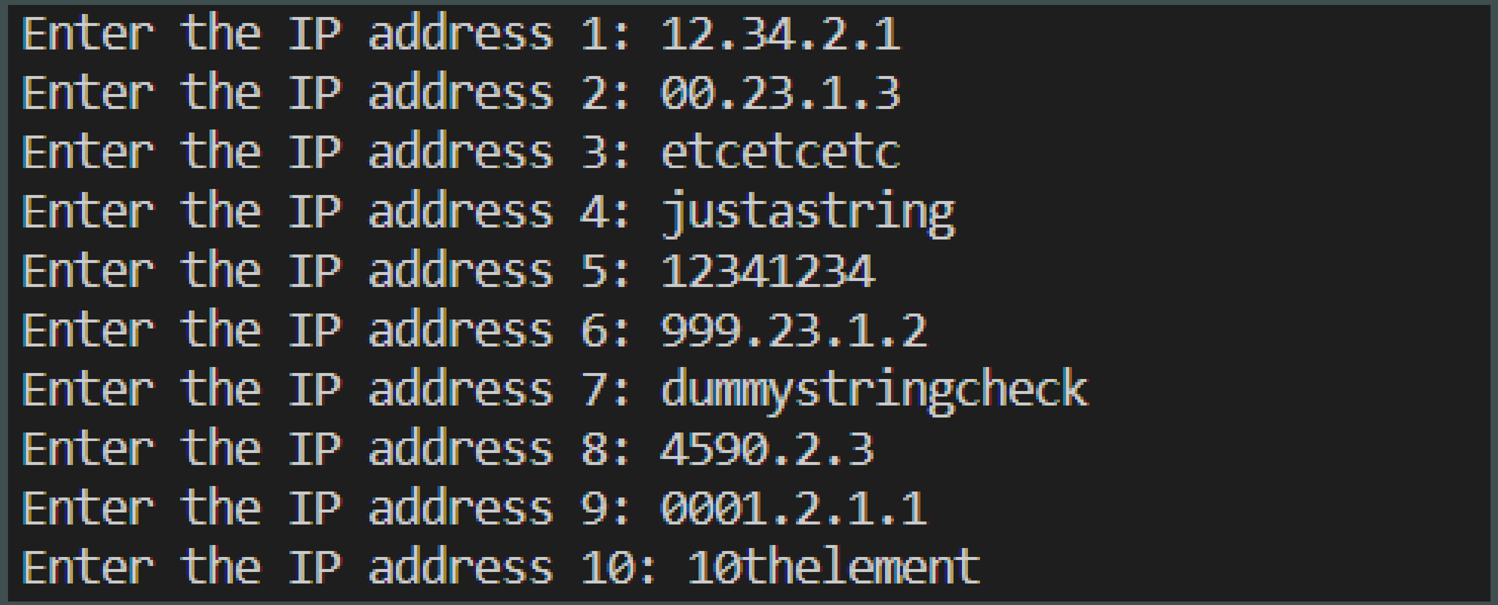
```python
d = {0:'first',1:'second',2:'third',3:'fourth',4:'fifth',5:'sixth',6:'seventh',7:'eighth',8:'ninth',
9:'tenth'}
with open('conversion.txt', 'r') as f:
    for i,line in enumerate(f):
        print(f"The {d[i]} IP address in Decimal, Binary, Octal and Hexadecimal format is {line}")
```

- I have initialized a python dictionary, d
- With the conversion.txt as the read source, we print all the IP addresses present in it
- We output in the form : The nth IP address in Decimal, Binary, Octal and hexadecimal format is <output from the file conversion. txt>

T6

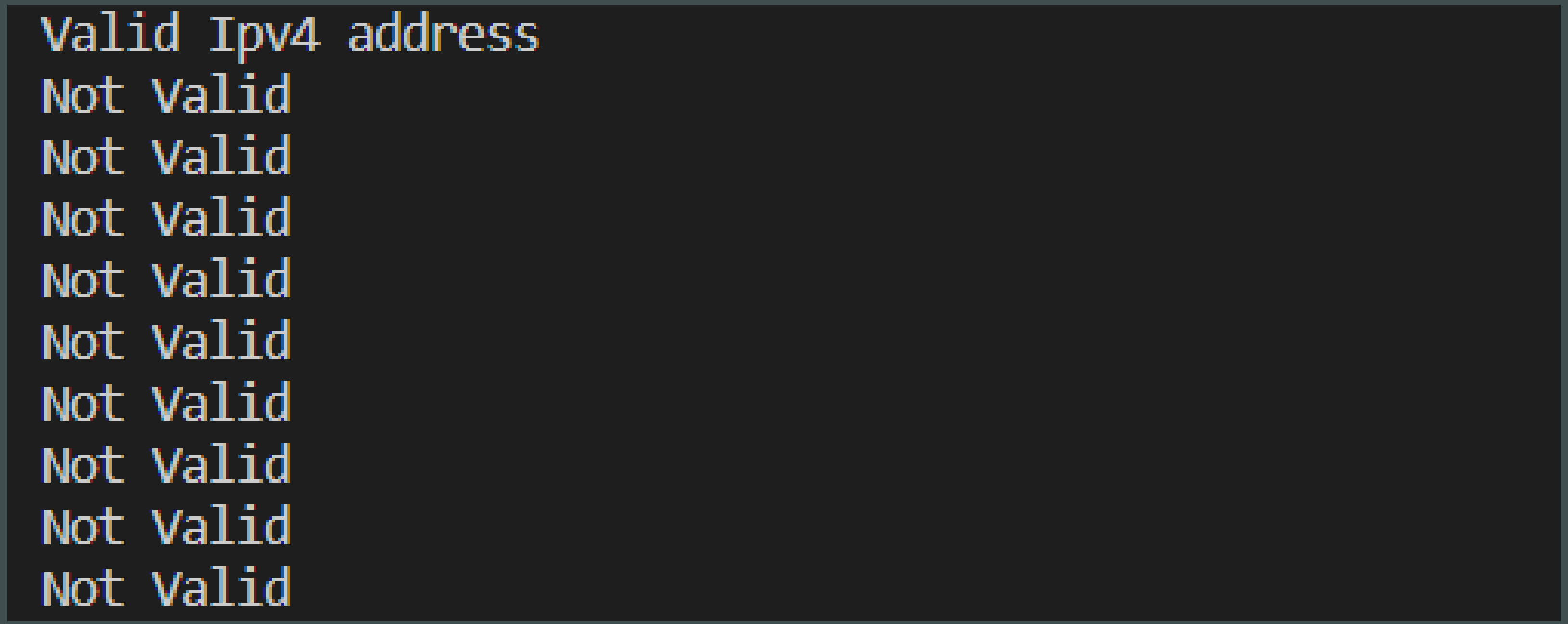
OUTPUT

TASK 1: inputs



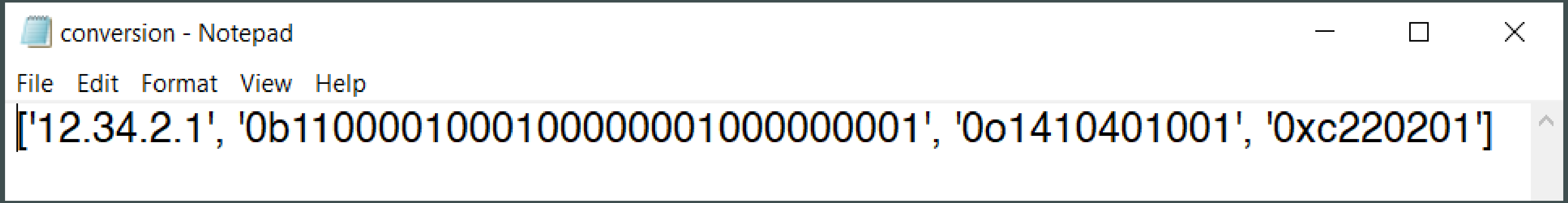
```
Enter the IP address 1: 12.34.2.1
Enter the IP address 2: 00.23.1.3
Enter the IP address 3: etcetcetc
Enter the IP address 4: justastring
Enter the IP address 5: 12341234
Enter the IP address 6: 999.23.1.2
Enter the IP address 7: dummystringcheck
Enter the IP address 8: 4590.2.3
Enter the IP address 9: 0001.2.1.1
Enter the IP address 10: 10thelement
```

TASK 2: validation



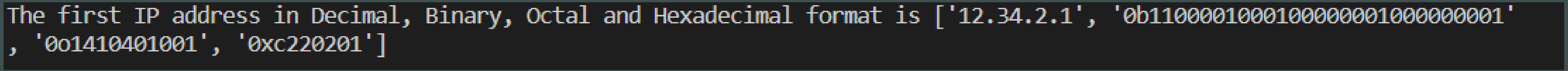
```
Valid Ipv4 address
Not Valid
Not Valid
Not Valid
Not Valid
Not Valid
Not Valid
Not Valid
Not Valid
Not Valid
```

TASK 5: conversion.txt



```
['12.34.2.1', '0b1100001000100000001000000001', '0o1410401001', '0xc220201']
```

TASK 6: output



```
The first IP address in Decimal, Binary, Octal and Hexadecimal format is ['12.34.2.1', '0b1100001000100000001000000001'
, '0o1410401001', '0xc220201']
```

General Comments

- I have utilized built-in packages like Regex and libraries like ipaddress for conversions
- Python data types such as lists, dictionaries, integers and strings were used.
- Methods like .append() were used
- I have also utilized python's file handling for storing the outputs in a text file

Thank You