Object-Oriented Design Lab Report (Python 1)

Bisakh Mondal 001810501079

Format: Question | Approach(if any) | Code | Output

Python Assignment

(Q1)

Write a prime generator program using only primes and using python loops.

Approach: To generate a random prime, First created a pool of primes using **Sieve of Eratosthenes** then randomly picking a prime from that pool.

```
# Prime genrator using Sieve of Eratosthenes

import math, random

MaxSize= int(1e4)
prime = [True]*MaxSize


#Sieve
def MakePrime():
    prime[0]=prime[1]=False
    for i in range(2, int(math.sqrt(MaxSize)) ,1):
        if prime[i]==True:
            for j in range(i*i,MaxSize,i):
                prime[j]=False

if __name__=="__main__":

    MakePrime()

    k= random.randint(0,MaxSize)
    for i in range(k,MaxSize):
        if prime[i]==True:
            print("Prime ", i)
            break
        #continue until a prime is found
        if i==MaxSize-1:
            i=0
```

```
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  31
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  37
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  79
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  23
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  13
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  89
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  627721
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  500107
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/prime.py"
Prime  9239
```

(Q2)

Write a discount coupon code using dictionary in Python with different rate coupons for each day of the week

Approach: Just basic handling with python dictionary.

```python
discountRate = {
   "sunday": "HappYSunDay",
   "monday": "WelComeMondAy",
   "tuesday": "BusyTuesdAY",
   "wednesday":"CooLWEDnesDaY",
   "thursday" : "SuPeRThursday",
   "friday": "BlackFridaY",
   "saturday": "EndSaturday"
}

if __name__=="__main__":
   inp = input("Enter the weekday: ")

   print("Coupon Code: ",discountRate.get(inp.lower(),"Invalid Weekday"))
```

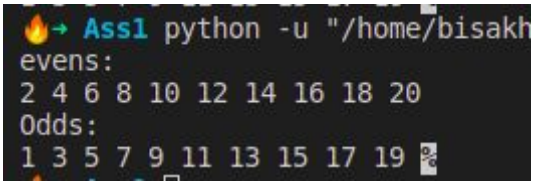
```
🔥→ Ass1 python -u "/home/bisakh/
Enter the weekday: sunday
Coupon Code:  HappYSunDay
🔥→ Ass1 python -u "/home/bisakh/
Enter the weekday: ss
Coupon Code:  Invalid Weekday
🔥→ Ass1 python -u "/home/bisakh/
Enter the weekday: monday
Coupon Code:  WelComeMondAy
```

(Q3)

Print the first 10 odd and even numbers using iterators and compress.

Approach: Since Compress takes an iterator and selector list as arguments, by passing a list of even index 1, with an generator to yield all-natural numbers will give the desired result.

```
from itertools import compress

def numbers():
    i=1
    while True:
        yield i
        i+=1

evensIdx = [0,1]*10
oddsIdx = [1,0]*10

evenItr = compress(numbers(),evensIdx)
oddsItr = compress(numbers(),oddsIdx)

print("evens: ")
for i in evenItr:
    print(i,end=' ')

print("\nOdds: ")
for i in oddsItr:
    print(i,end=' ')
```



(Q4)

Print the permutations of ABCDE using iterators.

Approach: Using some basic backtracking with recursion it can be easily formulated.

```python
def Permutation(s, start, end):
    if start==end:
        print(''.join(s))
        return
    for i in range(start,end):
        s[i],s[start]= s[start], s[i]
        Permutation(s,i+1,end)
        #discarding changes
        s[i],s[start]= s[start], s[i]

if __name__=="__main__":
    word="ABCD"
    Permutation(list(word),0,len(word))
```

```
EDCBA
🔥→ Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/4_permutations.py"
ABCDE | ABCED | ABDCE | ABDEC | ABECD | ABEDC | ACBDE | ACBED | ACDBE | ACDEB | ACEBD | ACEDB | ADBCE | ADBEC | ADCBE | ADCEB | ADEBC | AD
ECB | AEBCD | AEBDC | AECBD | AECDB | AEDBC | AEDCB | BACDE | BACED | BADCE | BADEC | BAECD | BAEDC | BCADE | BCAED | BCDAE | BCDEA | BCEA
D | BCEDA | BDACE | BDAEC | BDCAE | BDCEA | BDEAC | BDECA | BEACD | BEADC | BECAD | BECDA | BEDAC | BEDCA | CABDE | CABED | CADBE | CADEB
| CAEBD | CAEDB | CBADE | CBAED | CBDAE | CBDEA | CBEAD | CBEDA | CDABE | CDAEB | CDBAE | CDBEA | CDEAB | CDEBA | CEABD | CEADB | CEBAD |
CEBDA | CEDAB | CEDBA | DABCE | DABEC | DACBE | DACEB | DAEBC | DAECB | DBACE | DBAEC | DBCAE | DBCEA | DBEAC | DBECA | DCABE | DCAEB | DC
BAE | DCBEA | DCEAB | DCEBA | DEABC | DEACB | DEBAC | DEBCA | DECAB | DECBA | EABCD | EABDC | EACBD | EACDB | EADBC | EADCB | EBACD | EBAD
C | EBCAD | EBCDA | EBDAC | EBDCA | ECABD | ECADB | ECBAD | ECBDA | ECDAB | ECDBA | EDABC | EDACB | EDBAC | EDBCA | EDCAB | EDCBA |
```

(Q5)

Write a matrix multiplication function to compute.

Approach: if col of matA != row of matB, throwing ValueError, else a simple matrix multiplication code using O(N^3) complexity.

```python
#return mata*matb
def multiply(mata, matb):
    ra, ca = len(mata), len(mata[0])
    rb, cb = len(matb), len(matb[0])

    if ca!=rb:
        raise ValueError("Invalid Shape ")
    result = [[0]*cb for i in range(ra)]

    for i in range(ra):
        for j in range(cb):
            for k in range(ca):
                result[i][j]+= mata[i][k]*matb[k][j]
```

```
    return result

matA= [[1,2,3],
        [3,4,5]]

matB= [ [1,6],
        [4,5],
        [8,9]]

print(multiply(matA,matB))
```

```
Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/5_matMul.py"
[[9, 16], [19, 38]]
Ass1 python -u "/home/bisakh/Desktop/Assignments/python/Ass1/5_matMul.py"
Traceback (most recent call last):
  File "/home/bisakh/Desktop/Assignments/python/Ass1/5_matMul.py", line 21, in <module
    print(multiply(matA,matB))
  File "/home/bisakh/Desktop/Assignments/python/Ass1/5_matMul.py", line 8, in multiply
    raise ValueError("Invalid Shape ")
ValueError: Invalid Shape
```

(Q6)

Create list of servers, IP addresses and ports using variable positional and keyword arguments.

Python supports variable-length arguments using args and kwargs argurments.

```
def ServerDesc(*args,**kwargs):
    print(args[0]['name'])
    for key, val in kwargs.items():
        print("\nServer",key)
        print("Name: ", val['name'])
        print("IP: ",val['IP'])
        print("Open port: ", val['port'])
    print('-'*30)

serverList={
    's1':{
        'name':'APACHE2',
        'IP':'32.40.1.12',
        'port':'443'
    },
    's2':{
        'name':'NGINX',
```

```
        'IP':'132.45.1.12',
        'port':'80'
    }
}
if __name__=="__main__":
    ServerDesc(serverList['s1'],**serverList)
```

```
Server s1
Name:  APACHE2
IP:  32.40.1.12
Open port:  443

Server s2
Name:  NGINX
IP:  132.45.1.12
Open port:  80
----------------------------
```

(Q7)

Compute sorted five numbers using keyword-only arguments.

Approach: if we require too pass the arguments of a function using only the variable name then we can put a asterics before the keyword only arguements

```
def MySort(*, lis=[], ascending=True):
    l = len(lis)

    for i in range(0,l):
        for j in range(i+1,l):
            if lis[i] >lis[j]:
                lis[i],lis[j] = lis[j], lis[i]

    if not ascending:
        lis.reverse()

    return lis

if __name__=="__main__":
    inp = input("enter space seperated inputs ").split()
    #convert to int
    intlist = list(map(lambda x : int(x), inp))
    print("Sorted: ")
    print(MySort(lis=intlist))
```

```
(base) 🔥→ python python -u "/home/bisakh/Desktop/
enter space seperated inputs 5 9 8 7 45 2 0 5
Sorted:
[0, 2, 5, 5, 7, 8, 9, 45]
```

(Q8)

Create a list of all the numbers up to N=50 which are multiples of five using anonymous function.

Pretty simple approach: creating an anonymous function would help so.

```
MulOfFive = lambda N: [x for x in range(1,N+1) if x%5==0]

print(MulOfFive(50))
```

```
(base) 💥→ python python -u "/home/bisakh/D
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
```

(Q9)

Use map and zip to find the element-wise maximum amongst sequences of student list,subject list and marks list.

Here zip provides an iterator of tuples and by traversing the iterator tuples and mapping to max function we can find the index wise max element out of the three iterators

```
student=[10,12,45,60]
subjects=[15,10,5,6]
marks=[8,102,55,10]

if __name__=='__main__':
    lis = list(map(max, zip(student,subjects,marks )))
    print(lis)
```

```
[15, 102, 55, 60]
```

(Q10)

Filter out the odd squares using map, filter, list.

Approach: filter method return an iterator of a sequence after checking the odd condition, then map applies square function to each of the element of the iterable and return

```
listOfN = lambda N: list(range(N))

if __name__ =='__main__':
```

```
    inp = int(input("enter N: "))
    listN= listOfN(inp)

    lis= list(map(lambda z: z*z, filter(lambda x: x&1 ==1, listN)))

    print(lis)
```

```
(base) 🔥→ python python -u "/home/bisakh/Desktop/Assignmen
enter N: 25
[1, 9, 25, 49, 81, 121, 169, 225, 289, 361, 441, 529]
```

(Q11)

Let's find all Pythagorean triples whose short sides are numbers smaller than 10. Use filter and comprehension.

Using list comprehension will provide a list of triplets and applying filter to each triplets, will results in the required triplets.

```
N = int(input("Enter N: "))

triplets = [(a,b,c) for a in range(1,N+1) for b in range(a,N+1) \
            for c in range(b,N+1) if a**2 + b**2==c**2 ]

reqTriplets= list(filter(lambda x: not (x[0]<10 or x[1]<10), triplets))

print("Triplets with len greater than 10")
print(reqTriplets)
```

```
(base) 🔥→ python python -u "/home/bisakh/
Enter N: 25
Triplets with len greater than 10
[(12, 16, 20), (15, 20, 25)]
(base) 🔥→ python
```

(Q12)

Enumerate the sequence of all lowercase ASCII letters, starting from 1, using enumerate.

Using enumerate, it returns counter and adding it to ascii of 'a' will result in a sequence of all lowercase ascii letters.

```
lowercaseAsciiStart=97
numofAlphabets=26
```

```
for tup in enumerate(range(lowercaseAsciiStart,
lowercaseAsciiStart+numofAlphabets),start=1):
    print(tup[0], chr(tup[1]),end=' | ')
```

```
(base) 🔥→ python python -u "/home/bisakh/Desktop/Assignments/python/Ass1/tempCodeRunnerFile.py"
1 a | 2 b | 3 c | 4 d | 5 e | 6 f | 7 g | 8 h | 9 i | 10 j | 11 k | 12 l | 13 m | 14 n | 15 o | 16 p | 17 q | 18 r |
19 s | 20 t | 21 u | 22 v | 23 w | 24 x | 25 y | 26 z |
```

(Q13) Create a dictionary with comprehension with keys = the letters in the string of your name, and values of the same letters, but with the case swapped.

Using comprehension in dictionary, converting lowercase to upper and vice versa will do so.

```
name= input("Enter your name: ")

dicti = { ch:ch.lower() if ch.isupper() else ch.upper() for ch in name}

print(dicti)
```

```
(base) 🔥→ python python -u "/home/bisakh/Desktop/Assignments/
Enter your name: Bisakh
{'B': 'b', 'i': 'I', 's': 'S', 'a': 'A', 'k': 'K', 'h': 'H'}
(base) 🔥→ python
```

(Q14)

Write a python program to

1. read lines from a file, break into tokens and convert the tokens to unique numerical values using python dictionary.
2. Convert lines of different lengths into lines of same length (maximum length). Use padding if and when required.

To create an unique numeric for each tokens/words of the text, we can use the 'id' method returns an unique big integer for arguments, but **python due to garbage collected language, sometimes assign same id to different variables, to avoid such scenario id was 1 bit left shifted until the id doesn't appear into the dict.**

To manage line length of particular filestream it can be easily done using the help of a cache variable which will hold the excess length and concat if with the beginning of the next line.

```python
filename='Ass1/readme.txt'
f= open(filename, 'r')

def uniqueNumeric(fileHeader):
    fileHeader.seek(0)
    maps= {}
    for line in fileHeader:

        tokens= line.strip().split()
        for token in tokens:
            idt= id(token)
            while idt in maps:
                idt<<=1
            maps[idt]=token

    return maps

def LineLenManager(fileHeader,maxlen=80):
    fileHeader.seek(0)
    extras=''
    lines=[]
    for line in fileHeader:
        line= extras+' ' +line.strip()
        if len(line)<80:
            extras = line
        elif len(line)>80:
            extras=line[80:]
            lines.append(line[:80])
            print(line[:80])
    if extras!='':
        print(extras)
        lines.append(extras)
    return lines



print(uniqueNumeric(f))
LineLenManager(f)
```

```
(base) ➜ python python -u "/home/bisakh/Desktop/Assignments/python/Ass1/14_FILEtokens.py"
{140485566482864: 'Developing', 140485566452264: 'an', 140485566482928: 'intelligent', 140485566482992: 'dialogue', 1
40485566452320: 'system', 140485566452376: '1', 140485566452432: 'that', 140485566452488: 'not', 140485566452544: 'on
ly', 140485566483056: 'emulates', 140485566452600: 'human', 140485566483120: 'conversation,', 140485566452656: 'but',
 140485566452712: 'also', 140485566452768: 'answers', 140485566483184: 'questions', 140485566452824: 'on', 1404855664
52880: 'topics', 140485566452936: 'ranging', 140485566452992: 'from', 140485566453048: 'latest', 140485566453104: 'ne
ws', 140485566453160: 'about', 140485567330040: 'a', 140485566453216: 'movie', 140485566453272: 'star', 1404855664533
28: 'to', 140485566895280: 'Einstein's', 140485566453384: 'theory', 140485566453440: 'of', 140485566483248: 'relativi
ty,', 140485566453496: 'and', 140485566483312: 'fulfills', 140485566453552: 'complex', 140485566453608: 'tasks', 1404
85566453664: 'such', 140485566453720: 'as', 140485566453776: 'travel', 140485566483376: 'planning,', 140485566453832:
 'has', 140485566453888: 'been', 140485566453944: 'one', 140485566454000: 'of', 140485566454056: 'the', 1404855664541
12: 'longest', 140485566454168: 'running', 140485566454224: 'goals', 140485566454280: 'in', 140485566454336: 'AI.', 1
40485566454392: 'The', 140485566454448: 'goal', 140485566454504: 'has', 140485566483440: 'remained', 140485566454560:
 'elusive', 140485566454616: 'until', 140485566483504: 'recently.', 140485566454672: 'We', 140485566454728: 'are', 14
0485566454784: 'now', 140485566483568: 'observing', 140485566483632: 'promising', 140485566454840: 'results', 1404855
66454896: 'both', 140485566454952: 'in', 140485566483696: 'academia', 140485566483760: 'sindustry,', 140485566455008:
 'as', 140485566455064: 'large', 140485566455120: 'amounts', 140485566455176: 'of', 140485566483824: 'conversational'
, 140485566455232: 'data', 140485566455288: 'become', 140485566483888: 'available', 140485566455344: 'for', 140485566
483952: 'training,', 140485566455400: 'and', 140485566455456: 'the', 140485566484016: 'breakthroughs', 14048556645551
2: 'in', 140485566455568: 'deep', 140485566484080: 'learning', 140485566455624: '(DL)', 140485566455680: 'and', 14048
5566484144: 'reinforcement', 140485566484208: 'learning', 140485566455736: '(RL)', 140485566464048: 'are', 1404855664
64104: 'applied', 140485566464160: 'to', 140485566484272: 'conversational', 140485566464216: 'AI.'}

-----------------------------

 Developing an intelligent dialogue system 1 that not only emulates human conver
sation, but also answers questions on topics ranging from latest news about a mo
vie star to Einstein's theory of relativity, and fulfills complex tasks such as
travel planning, has been one of the longest running goals in AI. The goal has r
emained elusive until recently. We are now observing promising results both in a
cademia sindustry, as large amounts of conversational data become available for
training, and the breakthroughs in deep learning (DL) and reinforcement learning
 (RL) are applied to conversational AI.
```

(Q15)

Write a python program to identify and extract numerical chunks from a text file and convert them into words;

Nothing to be mentioned of, just reading the input from file and extracting the hundreds, thousand digits and converting them into words will results in the desired ans.

```
def decode(num):
    if(num>=100000):
        return
    single_digits = ["zero", "one", "two", "three",
                     "four", "five", "six", "seven",
                     "eight", "nine"]

    two_digits = ["ten", "eleven", "twelve",
                  "thirteen", "fourteen", "fifteen",
                  "sixteen", "seventeen", "eighteen",
                  "nineteen"]

    tens_multiple = ["", "", "twenty", "thirty", "forty",
                     "fifty", "sixty", "seventy", "eighty",
                     "ninety"]

    tens_power = ["hundred", "thousand"]


    thouDigits = int(num/1000)
```

```
    hunDigit = int(num/100)-10*thouDigits
    deciDigits=  num- 100*hunDigit - 1000*thouDigits


    phrase, two='',False

    if thouDigits>=9 and thouDigits<20:
        phrase+= two_digits[thouDigits%10]+' '
        two=True
    elif thouDigits>19:
        phrase += tens_multiple[int(thouDigits/10)] +' '

    if not two and thouDigits!=0:
        phrase+= single_digits[thouDigits%10]+ ' '
    if len(phrase.strip())!=0:
        phrase+=tens_power[1]+' '

    if not hunDigit==0:
        phrase+=single_digits[hunDigit]+' '+tens_power[0]+' '

    if deciDigits>=10 and deciDigits<20:
        phrase+=two_digits[deciDigits%10]+' '
    else:
        phrase+=tens_multiple[int(deciDigits/10)]+' '
        if deciDigits%10 !=0:
            phrase+=single_digits[deciDigits%10]

    return phrase.strip()

f= open('a.txt','r')
data = f.read().strip()
n=int(data)

print(decode(n))
```

```
Enter the Number: 40650
forty zero thousand six hundred fifty
(base) 🔥→ python python -u "/home/bisakh/De
Enter the Number: 25
twenty five
(base) 🔥→ python python -u "/home/bisakh/De
Enter the Number: 9801
nineteen thousand eight hundred  one
(base) 🔥→ python python -u "/home/bisakh/De
Enter the Number: 456
four hundred fifty six
```