

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Departamentul Informatică și Ingineria Sistemelor**

**Disciplina: Programarea intereactivă**

**Lucrare de laborator №2**

**Tema: Operaţii asupra Listelor și Seturi.**

A efectuat:

Pleșca Virgiliu,

*student*

IA-201

A controlat:

Strună Vadim,

*lector univ.*

**Chișinău 2021**

Cerințe:

* Rezolvați sarcina personală de la punctul I (nr. problemei conform registrului)

I. Sarcini personale

1. Să se calculeze suma numerelor naturale impare mai mici sau egale decât o valoare **n** dată.

Exemplu: Pentru *n=10* se obţine suma *25*.

1. Să se afişeze numerele prime de 3 cifre care au produsul cifrelor egal cu o valoare **p** dată.

Exemplu: Pentru *p=9* numerele *191, 313, 331, 911* îndeplinesc condiţiile problemei.

1. Două numere întregi **x** şi **y** sunt prietene dacă suma divizorilor numărului **x** este egală cu suma divizorilor numărului **y**. Să se găsească toate numerele prietene din intervalul **[a, b].**

Exemplu: Pentru *a=10* şi *b=25* se vor afişa perechile *(10, 17), (14, 15), (14, 23), (15, 23), (16, 25)*.

1. Să se afişeze numărul de perechi (şi perechile) de numere naturale care nu depăşesc un număr natural dat şi care au cel mai mare divizor comun un număr dat **d**.

Exemplu: Pentru *n=20* există *6* perechi *(5, 5), (5, 10), (5, 15), (5, 20), (10, 15), (15, 20)* care au cel mai mare divizor comun egal cu *5*.

1. Se cere să se afizeşe toate numerele de 3 cifre având cifrele în ordine crescătoare şi suma lor este egală cu 18.

Exemplu: Un astfel de număr este *369*.

1. Fiind dat un număr natural **x**, să se afişeze factorul prim care apare la puterea cea mai mare în descompunerea lui **x** în factori primi.

Exemplu: Pentru *x=1620*, descompunerea este *22 \* 34 \* 5*, iar factorul cerut este *3*.

1. Să se găsească numerele de 2 cifre care au următoarea proprietate: răsturnatul pătratului numărului este egal cu pătratul numărului răsturnat.

Exemplu: Un număr care îndeplineşte condiţiile problemei este *13* (*132’=169*, *312=961*).

1. Determinaţi cel mai mic număr mai mic sau egal cu **n** care are numărul maxim de divizori proprii (divizorul propriu este diferit de 1 şi de numărul însuşi).

Exemplu: Pentru *n=20* se obţine numărul *12* ( numerele *18* şi *20* au acelaţi număr de divizori proprii).

1. Determinaţi cel mai mic număr care care exact **k** divizori.

Exemplu: Pentru *k=4* se obţine numărul *6*.

1. Să se afişeze toate numerele prime de 3 cifre care citite invers sunt tot numere prime.

Exemplu: Un astfel de număr este *761* (*761* este prim şi *167* este prim).

1. Un număr natural se numeşte perfect daca el este egal cu suma divizorilor săi mai mici decât el. Să se verifice dacă un număr **n** este perfect sau nu.

Exemplu: Petnru *n=28* este numărul perfect deoarece *28=1+2+4+7+14*.

1. Să se afişeze toate perechile de numere palindromice din intervalul **[a, b]**. O pereche de numere se numeşte palindromică dacă al doilea este răsturnatul primului.

Exemplu: Pentru *a=10* şi *b=40* se obţin perechile *(11, 11), ( 12, 21), (13, 31), (22, 22), (23, 32), (33, 33)*.

1. Să se afişeze numerele naturale mai mici sau egale cu o valoare **n** dată cu proprietatea că suma cifrelor lor este un număr prim.

Exemplu: Pentru *n=25* se vor afişa valorile *2, 3, 5, 7, 11, 12, 14, 16, 20, 21, 23.*

1. Pentru un număr **n** (între 1 şi 1000) dat să se afişeze mulţimea divizorilor săi naturali (inclusiv 1 şi **n**), numărul divizorilor lui **n** şi suma divizorilor lui **n**.

Exemplu: Divizorii lui *n=20* sunt *1, 2, 4, 5, 10, 20*, suma lor este *42*, iar numărul lor este *6*.

1. Să se afişeze primele **n** numere care au suma cifrelor mai mică sau egală cu **m**.

Exemplu: Pentru *n=10* şi *m=4* se vor afişa numerele *1, 2, 3, 4, 10, 11, 12, 13, 20, 21*.

1. Dintre numerele naturale mai mici sau egale cu o valoare dată **n**, să se afişeze acelea care sunt divizibile cu suma cifrelor lor.

Exemplu: Pentru *n=25* se vor afişa valorile *1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 18, 20, 21, 24*.

1. Să se afizeşe descompunerea în factori primi ai unui număr natural nenul dat. Afişarea se va face sub forma:

3268 | 2

1634 | 2

817 | 19

43 | 43

1 |

1. Pentru un întreg **n** dat să se afişeze toate numerele naturale mai mici sau egale cu **n** a căror sumă a cifrelor este impară.

Exemplu: Pentru *n=15* se vor afişa numerele *1, 3, 5, 7, 9, 10, 12, 14*.

1. Să se afişeze toate numerele de 2 cifre care împărţite la 15 dau un rest egal cu pătratul câtului.

Exemplu, Un astfel de număr este *34* (*34:15=2 rest 4* şi *22=4*).

1. Să se afişeze toate numerele prime situate în intervalul **[p, q]**, precum şi numărul acestora, unde **p** şi **q** sunt două numere naturale date.

Exemplu: Pentru p=10 şi q=25 se vor afişa numerele 11, 13, 17, 19, 23.

# Mersul lucrarii

## Codul programului

import math

# 1.----------------------------------------------------------------

# Suma numerelor impare pana la n

def suma\_impare(n):

sum = 0

for i in range(1, n, 2): # Range 1 : \_n with step 2

sum = sum + i

return sum

print(" 1. ", suma\_impare(10))

# 2.---------------------------------------------------------------

# Verifies if the prod if digits equals to p

def prod\_verify(n, p):

a = n % 10

n = n // 10

b = n % 10

n = n // 10

c = n % 10

if a \* b \* c == p:

return True

else:

return False

def prime\_nums\_prod(\_p):

prime = []

res = []

# Find the all prime numbers which are of 3 digits

for num in range(100, 999):

# all prime numbers are greater than 1

for i in range(2, num):

if (num % i) == 0:

break

else:

prime.append(num)

# Find all the prime numbers whose digit product equals to \_p

for n in prime:

if prod\_verify(n, \_p):

res.append(n)

return res

print(" 2. ", prime\_nums\_prod(9))

# 3.--------------------------------------------------------------------

# Suma divizorilor lui n

def sum\_divisors(\_n):

s = 0

for i in range(1, \_n + 1):

if \_n % i == 0:

s = s + i

return s

def friend\_nums(a, b):

b = b + 1

d = {}

res = []

for i in range(a, b):

d[i] = sum\_divisors(i) # Dictionarul cu suma divizorilor pentru fiecare numar din intervalul [a, b]

for i in range(a, b):

for j in range(a, b):

if i == j: # Excludem elementele cu valorile tuplului identice

continue

elif res.\_\_contains\_\_((i, j)) or res.\_\_contains\_\_((j, i)): # Excludem tuplurile cu elementele asemanatoare

continue

elif d[i] == d[j]:

res.append((i, j))

return res

print(" 3. ", friend\_nums(10, 25))

# 4.-------------------------------------------------------------------------------------------------------

def div\_com(nr, div):

n = 0

print(" 4. ")

for i in range(1, nr + 1):

for j in range(i + 1, nr + 1):

if math.gcd(i, j) == div:

n = n + 1

print((i, j))

print(n, " perechi")

div\_com(20, 6)

# 5.-------------------------------------------------------------------------------------------------------

def is\_consec(n):

a = n % 10

n = n // 10

b = n % 10

n = n // 10

c = n

if a > b > c:

return True

else:

return False

def sum\_digits(n):

# Calculeaza suma cifrelor unui numar

a = n % 10

n = n // 10

b = n % 10

n = n // 10

c = n

return a + b + c

def sum(s):

arr = []

for i in range(100, 1000):

if is\_consec(i) and sum\_digits(i) == s:

arr.append(i)

print(" 5. ", arr)

sum(18)

# 6.-------------------------------------------------------------------------------

def desc(x):

arr = []

prim = []

p = 0

i = 0

# Numerele prime pana la x

for num in range(2, x + 1):

# all prime numbers are greater than 1

if num > 1:

for i in range(2, num):

if (num % i) == 0:

break

else:

prim.append(num)

i = 0

p = 0

while x > 1:

while x % prim[i] == 0:

p = p + 1

x = x / prim[i]

if p == 0:

i = i + 1

continue

arr.append((prim[i], p))

p = 0

i = i + 1

print(" 6. ", arr)

desc(512)

# 7.---------------------------------------------------------------------

def reverse(x):

return int(str(x)[::-1])

i = 0

a = []

def \_7():

for i in range(10, 100):

if i\*\*2 == reverse(reverse(i)\*\*2):

a.append(i)

print(" 7. ", a)

\_7()

# 8.------------------------------------------------------------------

def \_8(n):

d = []

r = []

# Gasim divizorii proprii

for i in range(1, n):

k = 0

for j in range(1, i):

if(i % j == 0 and j != 1 and j != i):

k = k + 1

d.append((i, k))

# Gasim numerele cu cei mai multi divizori proprii

max = -1

ind = 0

for i in range(len(d)):

if(d[i][1] >= max):

max = d[i][1]

ind = i

# Numaram cate numere cu nr maxim de div proprii

a = []

for i in range(len(d)):

if(d[i][1] == max):

a.append(i)

for i in a:

r.append(d[i])

print(" 8. ", r)

\_8(20)

# 9.------------------------------------------------------

def \_9(k):

i = 1

while(True):

count = 0

for j in range(1, i + 1):

if(i % j == 0):

count = count + 1

if(count == k):

print(" 9. ", i)

break

i = i + 1

\_9(4)

# 10.---------------------------------------------------

k = 0

def is\_prime(x):

global k

if x == 2:

return True

for i in range(2, x):

if(x % i == 0):

return False

elif(i == x - 1):

return True

r = []

def \_10():

prime = []

r = []

# generam numere prime

for num in range(100, 999):

for i in range(2, num):

if(num % i == 0):

break

elif(i == num - 1):

prime.append(num)

# verificam care numere sunt inverse si prime

for num in prime:

if(is\_prime(reverse(num))):

if(not r.\_\_contains\_\_((num, reverse(num))) and not r.\_\_contains\_\_((reverse(num), num)) and reverse(num) != num):

r.append((num, reverse(num)))

print(" 10. ", r)

\_10()

# 11.--------------------------------------------------------------------------------------------------------------------

# Numere perfecte

def \_11(x):

s = 0

for i in range(1, x):

if(x % i == 0):

s = s + i

if(s == x):

print(" 11. Numarul", x, "este pefect. Suma divizorilor este:", s)

else:

print(" 11. Numarul", x, "nu este pefect. Suma divizorilor este:", s)

\_11(30)

# 12.------------------------------------------------------------------------------

def \_12(a, b):

r = []

for i in range(a, b + 1):

for j in range(i, b + 1):

if i == reverse(j):

r.append((i, j))

print(" 12. ", r)

\_12(10, 40)

# 13.----------------------------------------------------------------------------

def sum\_digits(x):

s = 0

for digit in str(x):

s = s + int(digit)

return s

def \_13(n):

r = []

for i in range(n):

if(is\_prime(sum\_digits(i))):

r.append(i)

print(" 13. ", r)

\_13(25)

# 14.-------------------------------------------------------

def \_14(n):

if(n < 1 and n > 1000):

print("Erorr: Ati introdus o valoare care nu se incadreaza in interval")

return -1

d = []

s = 0

for i in range(1, n + 1):

if(n % i == 0):

d.append(i)

s = s + i

print(" 14. ", d)

print(" Suma divizorilor:", s)

\_14(20)

# 15.-------------------------------------------------------------------

def \_15(n, m):

r = []

i = 1

while(len(r) < n):

if(sum\_digits(i) <= m):

r.append(i)

i = i + 1

print(" 15. ", r)

\_15(10, 4)

# 16.----------------------------------------------------------------

def \_16(n):

r = []

for i in range(1, n):

if(i % sum\_digits(i) == 0):

r.append(i)

print(" 16. ", r)

\_16(25)

# 17.------------------------------------------------------------

# Returneaza

def closest\_num\_to\_divide(n):

i = 2

while(True):

if(n % i == 0):

return i

i = i + 1

def \_17(n):

r = []

print(" 17. ")

while(True):

if(n == 1):

break

print(" ", n, " | ", closest\_num\_to\_divide(n))

n = n / closest\_num\_to\_divide(n)

print(" ", 1, " |")

\_17(3268)

# 18.--------------------------------------------------------------

def is\_odd(n):

if(n % 2 != 0):

return True

else:

return False

def \_18(n):

r = []

for i in range(n + 1):

if(is\_odd(sum\_digits(i))):

r.append(i)

print(" 18. ", r)

\_18(15)

# 19.-------------------------------------------------

def \_19():

r = []

for i in range(10, 100):

cat = i // 15

rest = i % 15

if cat \*\* 2 == rest:

r.append(i)

print(" 19. ", r)

\_19()

# 20.---------------------------------------------------------

def \_20(p, q):

r = []

k = 0

for i in range(p, q + 1):

if(is\_prime(i)):

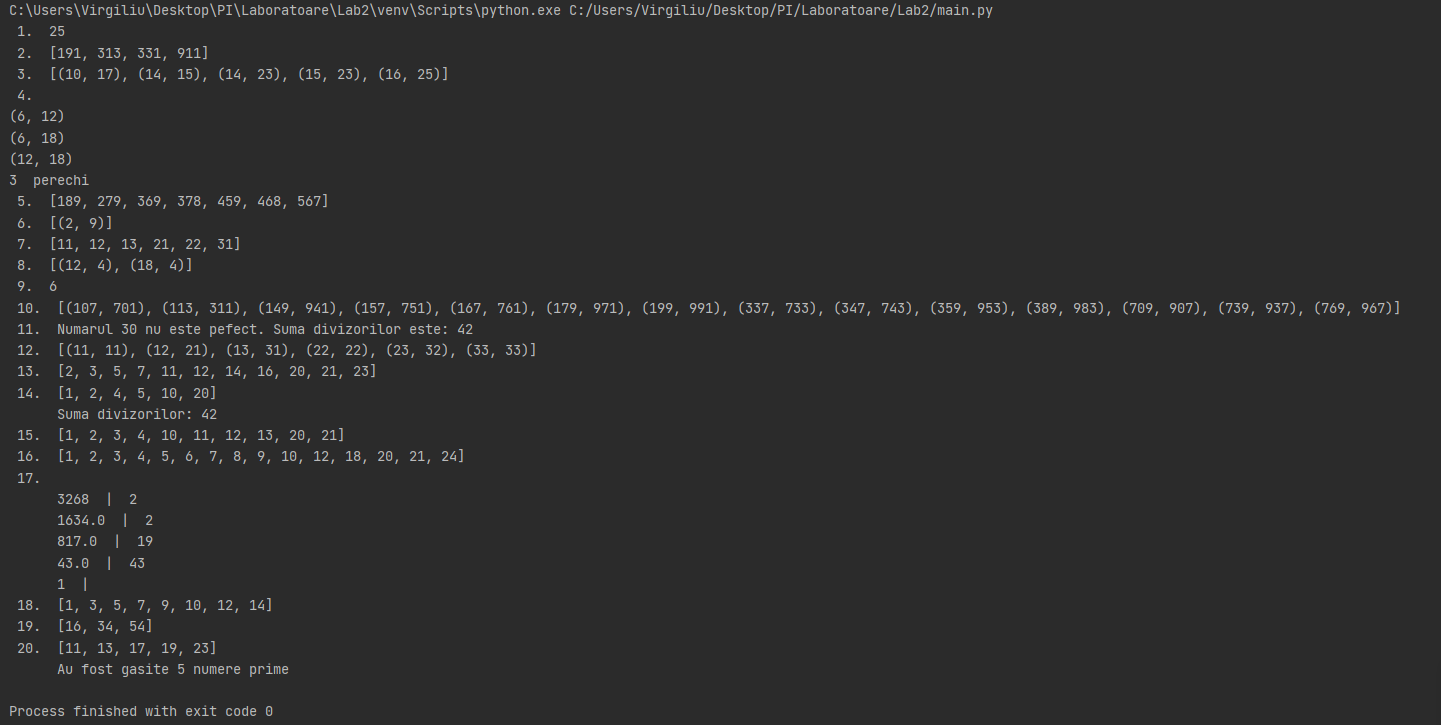
r.append(i)

k = k + 1

print(" 20. ", r, "\n Au fost gasite", k, "numere prime")

\_20(10, 25)

## Output



**Figura 1.1** – Output-ul programului

# Conclusion

În urma elaborării acestei lucrări de laborator am aplicat cunoștințele acumulate la orele de curs și anume mi-am întărit cunoștințele ce țin de seturi, tupluri, liste. Am înțeles că o listă in python este un tip de date mutabil, care poate conține în el mai multe elemente de tipuri diferite. Tuplul deja este ca o lista doar ca pentru a declara un tuplu se folosesc paranteze patrate și în afară de asta obiectele de tipul tuple sunt imutabile. Iar seturile sunt un tip de date care la fel poate conține în el mai multe elemente, la fel ca și în cazul listelor și tuplurilor. Diferența este că seturile nu pot conține elemente dublicate, toate elementele trbuie sa fie unice și totodată seturile sunt neordonate. Am folosit metodele predefinite pentru aceste tipuri de date, de ex. Push(), pop(), \_\_contains\_\_(), metodele care implementeaza functii matematice ca reuniunea, intersecția etc.