MINISTERUL EDUCAȚIEI, CULTURII şi CERCETĂRII al Rep. MOLDOVA
UNIVERSITATEA TEHNICĂ a MOLDOVEI
FACULTATEA CALCULATOARE, INFORMATICĂ şi MICROELECTRONICĂ
Departamentul „Tehnologia Informaţiei”

LUCRARE INDIVIDUALĂ
GRAFICA PE CALCULATOR

Student: Popa Cătălin
gr. TI-211, FCIM
Conducător:
Conf. univ. MALCOCI Iulian

CHIȘINĂU 2022

CUPRINS

CUPRINS 2
Tema 4. Operatori logici si de comparatie. Structuri repetitive.....3-18
Tema 5. Funcții 19-29
Sarcină suplimentară-Turtle.....30-33
Yin Yang.....34-35
Aplicatia calculator.....36-40

Mo	Coal	Nr.	Semn.	Data	GC Nr. 21-509 – Popa Catalin		
Elaborat		Popa Catalin			Lucrare independentă la disciplina: Grafica pe calculator	Litera	Coala 1 / Coli 60
Verificat		Malcoci Iulian				UTM FCIM	Gr. TI-211

Ex.4.1 **TEMA 4**

Ce răspuns vom obține după rularea următorului cod?

```
print ("Python" == "PYTHON")
print ("Python" != "PYTHON")
print ("PYTHON" != "PYTHON")
print (15 <= 14)
print (15 <= 15)
```

Răspuns:

```
False
True
False
False
True
```

Ex.4.2

Ce răspuns vom obține după rularea următorului cod?

Ce răspuns vom obține după rularea următorului cod?

```
print('Welcome' != 'WELCOME' and (7>5) and (4==4))
print('Welcome' != 'WELCOME' or (7>5) or (4==4))
print('Welcome' == 'WELCOME' and (7<5) and (4==4))
print('Welcome' == 'WELCOME' or (7<5) or (4==4))
print('Welcome' == 'WELCOME' or (7<5) and (4==4))
print('Welcome' == 'WELCOME' and (7<5) or (4==4))
```

Răspuns:

```
True
True
False
True
True
True
```

Ex.4.3

Ce răspuns vom obține după rularea următorului cod?

```
x = 15
y = 7

print(x == y)
print(x != y)
print(x > y)
print(x >= y)
print(x == y and x != y)
print(x == y or x != y)
```

Răspuns:

```
False
True
True
True
False
True
```

Ex.4.4

Să presupunem că avem următorul dicționar my_dict ={'key 1': 1, 'key 2': 7, 'key 3': 9}, trebuie să creăm un cod care la răspuns să ne dea o listă cu valorile cheilor din dicționar care sunt mai mari decât 5.?

Răspuns:

```
my_dict ={'key 1': 1, 'key 2': 7, 'key 3': 9}
del[my_dict['key 1']]
print(my_dict.values())
```

```
dict_values([7, 9])
```

Ex.4.5

În acest exemplu trebuie să generaţi un număr aliator a între 1 şi 100 (sugestie: importam modulul random şi folosim .randint(1,100)), să cereţi de la utilizator să introducă un număr b între 1 şi 100, după care să comparăm aceste două numere.

Răspuns:

```
import random
a = random.randint(1,100)
print(a)
b = int(input("Introduceti un numar, intre 1 si 100: "))
if(a > b):
    print('Numarul mai mare este: ', a)
elif(a < b) :
     print('Numarul mai mare este: ',b)
else:
    print('Numerele sunt egale')
```

```
69
Introduceti un numar, intre 1 si 100: 50
Numarul mai mare este:  69
```

Ex.4.6

În acestexemplu trebuie să scrieţi un cod în care să cereţi de la utilizator să introducă orice număr doreşte, iar la răspuns să obţinem dacă numărul ales este pozitiv, negativ sau egal cu zero.

Răspuns:

```
b = int(input("Introduceti un numar: "))
if(b > 0):
    print('Numarul este pozitiv')
elif(b < 0) :
     print('Numarul este negativ')
else:
    print('Numarul este egal cu zero')
```

```
Introduceti un numar: -9
Numarul este negativ
```

```
Introduceti un numar: 0
Numarul este egal cu zero
```

```
Introduceti un numar: 5
Numarul este pozitiv
```

Ex.4.7

În acest exemplu trebuie să scrieţi un cod în care să cereţi de la utilizator să introducă un număr între 1 şi 10000, după care să verificăm dacă acest număr este multiplul lui 5 dar nu şi multiplul lui 3.

Răspuns:

```
def isMultipleof5(n):

    if n%5 == 0 and n%3 != 0:
        return 1

    return 0

i = int(input("Dati un numar de la 1 la 10000: "))
if ( isMultipleof5(i) == 1 ):
    print (i, ": este multimplu lui 5 dar nu este multiplu lui 3")
else:
    print (i, ": nu verifica conditiile")
```

```
Dati un numar de la 1 la 10000: 15
15 : nu verifica conditiile

=============== RESTART: C:\Users\Catalin\Desktop\GUI\exemplu.py
Dati un numar de la 1 la 10000: 5
5 : este multimplu lui 5 dar nu este multiplu lui 3
```

Ex.4.8

În acest exemplu trebuie să scrieţi un cod în care să cereţi de la utilizator să introducă un număr pozitiv, iar la răspuns să obţinem din câte cifre este format numărul ales de utilizator. De exemplu daca utilizatorul a ales numărul 77la răspuns trebuie să obţinem: Nr ales de utilizator are 2 cifre.

Răspuns:

```
Dati un numar pozitiv: 154313
Number of digits: 6
```

```
Dati un numar pozitiv: 21
Number of digits: 2
```

```
Dati un numar pozitiv: 2
Number of digits: 1
```

Ex.4.9

În acest exemplu trebuie să scrieţi un cod în care să cereţi de la utilizator să introducă un număr oarecare, iar la răspuns să obţinem răspunsul dacă acest număr este par sau impar.

Răspuns:

```
i = int(input("Dati un numar: "))
if ( i%2 == 0 ):
    print (i, ": este par")
else:
    print (i, ": este impar")
```

```
Dati un numar: 2
2 : este par

=============== R
Dati un numar: 5
5 : este impar
```

Ex.4.10

Pentru dicţionarul my_dict = {'Ana' : 11, 'Mariana': 40, 'Iulian': 41} scrieţi un cod care la răspuns să nea dea o listă care să conţină doar valorile cheilor din dicţionar care sunt mai mari de 25.

Răspuns:

```
my_dict = {'Ana' : 11, 'Mariana' : 40, 'Iulian' : 41}
del[my_dict['Ana']]
print(my_dict.values())
```

```
dict_values([40, 41])
```

Ex.4.11

Care va fi răspunsul după rularea următoarelor coduri:

a)
```
lista_mea = ['Ion', 'Ana', 'Maria']
for x in lista_mea:
    print(x)
    print('Salut')
```
b)
```
lista_mea = ['Ion', 'Ana', 'Maria']
for x in lista_mea[0:2]:
    print(x)
    print('Salut')
```
c)
```
lista_mea = ['Ion', 'Ana', 'Maria']
for x in lista_mea:
    print(x)
print('Salut')
```

Răspuns:

a)

```
Ion
Salut
Ana
Salut
Maria
Salut
```

b)

```
Ion
Salut
Ana
Salut
```

c)

```
Ion
Ana
Maria
Salut
```

Ex.4.12

Care va fi răspunsul după rularea următoarelor coduri:

a)
```
lista_mea = ['Ion', 'Ana', 'Maria']
for x in 'Ana':
    print(x)
```
b)
```
lista_mea = ['Ion', 'Ana', 'Maria']
for x in 'Ana':
    print(x, end = '')
```

Răspuns:

a)

```
A
n
a
```

b)

```
A n a
```

Ex.4.13

Care va fi răspunsul după rularea următoarelor coduri:

```
lista_mea = ['nokia', 'samsung', 'google', 'iphone']

i=0
while i < len(lista_mea):
    telefoane = lista_mea[i]
    i += 1

    if telefoane == 'google':
        continue

    print(telefoane)
```

Răspuns:

```
nokia
samsung
iphone
```

Ex.4.14

Scrieţi un cod folosind bucla for şi al doilea cod folosind bucla while astfel încât la răspuns să obţinem:

```
1
2
3
4
5
6
7
```

Răspuns:

while

```
i = 1
while(i<=7):
    print(i)
    i += 1
```

for

```
n = 7
for i in range(1, n + 1):
    print (i)
```

Ex.4.15

Scrieţi un cod folosind bucla for astfel încât la răspuns să obţine 3 coloane, in prima coloană să fie numerele de la 1 la 10, în a doua coloană să fie pătratul acestor numere, iar a treia coloană să fie cubul acestor numere ca în exemplu:

a)

```
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
```

b)

```
 1   1    1
 2   4    8
 3   9   27
 4  16   64
 5  25  125
 6  36  216
 7  49  343
 8  64  512
 9  81  729
10 100 1000
```

Răspuns:

```
for n in [1,2,3,4,5,6,7,8,9,10]:
    square = n**2
    square2 = n**3
    print(n,square,square2)
```

```
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
```

Ex.4.16

Scrieţi un cod care trebuie să simuleze aruncarea zarurilor cu ajutorul buclei while şi modulului random. Răspunsul trebuie să arate în felul următor:

```
***** ARUNCĂ ZARURILE *****
Zarurile sunt aruncate asteapta ...
Zarul 1 indică: 4
Zarul 2 indică: 2
Apasa tasta y pentru a arunca din nou zarurile: y
Zarurile sunt aruncate asteapta ...
Zarul 1 indică: 1
Zarul 2 indică: 4
Apasa tasta y pentru a arunca din nou zarurile:
```

Răspuns:

```
import random
print('******ARUNCA ZARURILE******')
min = 1
max = 6
roll_again = "y"
while roll_again == "yes" or roll_again == "y":
    print ("Zarurile sunt aruncate, asteapta...",)
    print ("Zarul 1 indica: ",random.randint(min, max))
    print ("Zarul 2 indica: ",random.randint(min, max))
    roll_again = input("Apasati tasta y pentru a arunca din nou zarurile:")
```

```
******ARUNCA ZARURILE******
Zarurile sunt aruncate, asteapta...
Zarul 1 indica:  4
Zarul 2 indica:  6
Apasati tasta y pentru a arunca din nou zarurile:y
Zarurile sunt aruncate, asteapta...
Zarul 1 indica:  2
Zarul 2 indica:  2
```

Ex.4.17

Trebuie să scriem un cod care să ceară de la utilizator să introducă un număr de la 1 până la 9, după care folosind bucla for să obținem tabla înmulțirii cu acel număr ales de utilizator. Răspunsul trebuie să arate în felul următor:

```
Alege un numar cuprins între 1 si 9: 7
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

Răspuns:

```
import random

number = int(input("Alege un numar cuprins intre 1 si 9: "))
for count in range(1, 11):
   print (number, 'x', count, '=', number * count)
```

```
Alege un numar cuprins intre 1 si 9: 8
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

Ex.4.18

Trebuie să scriem un cad care să ceară de la utilizator o valoare minimă şi una maximă, iar după rulare la răspuns să apară doar numerele pare dintre aceste două valori alese de utilizator. Răspunsul trebuie să arate în felul următor:

```
Introdu limita de jos: 45
Introdu limita de sus: 63
46
48
50
52
54
56
58
60
62
```

Răspuns:

```
num1 = int(input("Introdu limita de jos: "))
num2 = int(input("Introdu limita de sus: "))
for count in range(num1, num2):
    if count%2 == 0:
        print (count)
```

```
Introdu limita de jos: 45
Introdu limita de sus: 63
46
48
50
52
54
56
58
60
62
```

Ex.4.19

Ce răspuns vom obţine dacă vom rula următorul cod:

```
for i in 'Dacia':
    if i == 'a':
        continue
    print(i)
```

Răspuns:

```
for i in 'Dacia':
    if i == 'a':
        continue
    print(i)
```

```
D
c
i
```

Ex.4.20

Trebuie să scriem un cod în care să cerem de la utilizator să introducă câteva valori separate prin spaţiu pentru temperatura în grade C sub formă de listă, după care această listă să o transformăm într-o listă nouă care să ne arate aceste valori în grade F cu ajutorul buclei for. Răspunsul trebuie să arate în felul următor:

```
Introdu 4 valori pentru temp in grade C separate prin spatiu
0 20 30 45
Temperatura în grade C si grade F
 [0, 20, 30, 45] [32.0, 68.0, 86.0, 113.0]
```

Răspuns:

```
print("Introdu 4 valori pentru temp in grade C separate prin spatiu: ")
m, n, x, c = list(map(float,input().split()))
print('Temperatura in grade C si grade F')
ml =(m * (9/5) + 32)
nl =(n * (9/5) + 32)
xl =(x * (9/5) + 32)
cl =(c * (9/5) + 32)
print(m,n,x,c)
print(ml,nl,xl,cl)
```

```
Introdu 4 valori pentru temp in grade C separate prin spatiu:
3 4 5 6
Temperatura in grade C si grade F
3.0 4.0 5.0 6.0
37.4 39.2 41.0 42.8
```

Ex.4.21

Trebuie să scriem un cod în care utilizatorul să specifice lungimea parolei pe care calculatorul trebuie s-o genereze în mod aleatoriu cu ajutorul modulului random şi folosirea buclei for, iar caracterele să conţină litere, numere şi caractere speciale din modulul string.

```
import string
print('Numere:', string.digits)
print('Litere:', string.ascii_letters)
print('Simboluri:', string.punctuation)
```

După rulare vedem ce conține fiecare instrucțiune din modulul `string`:

```
Numere: 0123456789
Litere: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
Simboluri: !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

După ce terminăm de scris codul care va genera o parolă în mod aleatoriu vom obține un răspuns care arată în felul următor:

```
Introdu lungimea parolei: 5
sdAsI
```

Răspuns:

```
l = int(input('\nIntrodu lungimea parolei:'))
lower = string.ascii_lowercase
upper = string.ascii_uppercase
num = string.digits
symbols = string.punctuation
all = lower + upper + num + symbols
t = random.sample(all,l)
password = "".join(t)
print(password)
```

```
Introdu lungimea parolei:7
6HD]ySe
```

Ex.4.22

Trebuie să scriem un cod care să îl întrebe pe utilizator cum se numeşte şi câţi ani are,după care utilizând comenzile .split(), .sort() şi bucla for să sortăm cuvintele din propoziţia introdusă de utilizator. După rularea codului răspunsul trebuie să arate în felul următor:

```
Cum te numesti si ce vârstă ai?: Mă numesc Iulian Malcoci si am 41 de ani.
41
Iulian
Malcoci
Mă
am
ani.
de
numesc
si
```

Răspuns:

```
def sortare(string):
    lista = string.split()
    lista.sort()
    for word in lista:
        print(word," ")
lista = input("Cum te numesti si ce varsta ai: ")
sortare(lista)
```

```
Cum te numesti si ce varsta ai: Ma numesc Catalin Popa si am 20 de ani.
20
Catalin
Ma
Popa
am
ani.
de
numesc
si
```

Ex.4.23

Având două liste culori = ['Alb', 'Negru', 'Oranj'] şi auto = ['Duster', 'Logan', 'Dokker']trebuie să scriem două coduri în care să folosim bucle imbricate pentru a obţine următoarele răspunsuri:

```
[('Duster', 'Alb'), ('Logan', 'Alb'), ('Dokker', 'Alb'), ('Duster', 'Negru'),
('Logan', 'Negru'), ('Dokker', 'Negru'), ('Duster', 'Oranj'), ('Logan', 'Oran
j'), ('Dokker', 'Oranj')]
```

Răspuns:

```
culori = ['Alb', 'Negru', 'Oranj']
auto = ['Duster', 'Logan', 'Dokker']
final = [[x, y] for x in auto for y in culori if x != y]
print(final)
```

```
[['Duster', 'Alb'], ['Duster', 'Negru'], ['Duster', 'Oranj'], ['Logan', 'Alb'], ['Logan', 'Negru'],
['Logan', 'Oranj'], ['Dokker', 'Alb'], ['Dokker', 'Negru'], ['Dokker', 'Oranj']]
```

Ex.4.24

Trebuie să scriem un cod care să ceară de la utilizator să introducă o propoziţie, după care cu ajutorul buclei for să obţinem la răspuns nr. de

litere mici şi nr. de litere mari. Răspunsul trebuie să arate în felul următor:

```
Introdu limita de jos începând cu 100: 123
Introdu limita de sus care nu depăseste 999: 217
['200', '202', '204', '206', '208']
```

Răspuns:

```
string = input("Introdu o propozitie:")
x1 = 0
x2 = 0
for i in string:
    if(i.islower()):
        x1 = x1 + 1
    elif(i.isupper()):
        x2 = x2 + 1
print('Numarul de litere mari este:',x2)
print('Numarul de litere mici este:',x1)
```

```
Introdu o propozitie:Chisinau Moldova
Numarul de litere mari este: 2
Numarul de litere mici este: 13
```

Ex.4.25

Trebuie să scriem un cod în care să cerem de la utilizator să introducă limita de jos şi limita de sus şi limita de jos intre 100 şi 999. După care la răspuns să obţinem o listă care să cuprindă o listă doar din numerele care au toate cele trei cifre numere pare.Răspunsul trebuie să arate în felul următor:

```
Introdu limita de jos începând cu 100: 123
Introdu limita de sus care nu depăseste 999: 217
['200', '202', '204', '206', '208']
```

Răspuns:

```
num1 = int(input("Introdu limita de jos(intre 100 si 999): "))
num2 = int(input("Introdu limita de sus(intre 100 si 999): "))
a = set('02468')
lista=[res for res in range(num1,num2)if set(str(res)) <= a]
print(lista)
```

```
Introdu limita de jos(intre 100 si 999): 120
Introdu limita de sus(intre 100 si 999): 215
[200, 202, 204, 206, 208]
```

Ex.4.26

În acest exerciţiu vom folosi modulul time şi comanda .sleep() pentru ca de exemplu răspunsul dorit de noi să apară pe ecran să apară la fiecare 5 secunde. Vom întreba care este filmul preferat al utilizatorului. După rulare la fiecare 5 secunde va apărea mesajul: Acest film este SUPER!!!După rulare vom obţine:

```
s = input('Care este filmul tau preferat?: ')
count = 0
while count<5:

  print(f"{s} este un film SUPER!!!")
  time.sleep(5)
  count+=1
```

Răspuns:

```
Care este filmul tau preferat?: Puak cic pak pac
Puak cic pak pac este un film SUPER!!!
Puak cic pak pac este un film SUPER!!!
Puak cic pak pac este un film SUPER!!!
Puak cic pak pac este un film SUPER!!!
Puak cic pak pac este un film SUPER!!!
```

Ex.4.27

În acestexerciţiu îi vom cere utilizatorului să introducă o parolă cu condiţiile ca această parolă să fie formată din 10 caractere, să conţină cel puţin o literă mare şi cel puţin o cifră. Aici vom folosi cam tot ce am învăţat până acum şi bucla while şi bucla for şi operatorul logic andşi instrucţiunile condiţionate if şi else. După rulare răspunsul poare arăta în felul următor:

```
Introdu parola: qwertya555
Parola nu are 10 caractere, o cifra sau majusculă!!!
Introdu parola: Qwertya555
Parola nouă a fost salvată cu succes
```

Răspuns:

```python
import re

validation = False
while validation == False:
    password = input("Introdu parola: ")
    if len(password) < 10:
        print("Verifica daca parola ta are 10 caractere.")
    elif re.search('[0-9]',password) is None:
        print("Verifica daca parola ta are o cifra.")
    elif re.search('[A-Z]',password) is None:
        print("Verifica daca parola ta are o litera capitala.")
    else:
        print("Parola noua a fost setata cu succes")
        validation = True
```

```
Introdu parola: alionaa777
Verifica daca parola ta are o litera capitala.
Introdu parola: Alionaa777
Parola noua a fost setata cu succes
```

Ex.5.1 **TEMA 5**

Trebuie să creăm o funcţie care să redea pătratul oricărui număr, iar numărul respectiv să fie introdus de către utilizator cu funcţia input(). După rularea codului răspunsul trebuie să arate în felul următor:

```
Introdu un număr: 4
Pătratul numărului 4 este 16
```

Răspuns:

```
def up(x):
    return x**2;
num = int(input("Introdu un numar: "));
print(f"Patratul numarului {num} este : {up(num)}")
```

```
Introdu un numar: 5
Patratul numarului 5 este : 25
```

Ex.5.2

Trebuie să creăm o funcţie care să redea suma şi produsul a două numere. Numerele trebuie să fie introduse de utilizator. După rularea codului răspunsul trebuie să arate în felul următor:

```
Introdu primul număr: a=5
Introdu al doilea număr: b=3
Suma a+b=8, Produsul a*b=15
Suma 5+3=8, Produsul 5*3=15
```

Răspuns:

```
def up(a,b):
    print(f"Suma a+b = {a+b}, Produsul a*b = {a*b}")
    print(f"Suma {a}+{b} = {a+b}, Produsul {a}*{b} = {a*b}")
a = int(input("Introduceti primul numar: "))
b = int(input("Introduceti primul numar: "))
up(a,b);
```

Ex.5.3

Trebuie să realizăm un cod care să redea suma şi produsul a două numere folosindu-ne de expresia lambda. Numerele trebuie să fie introduse de utilizator. După rularea codului răspunsul trebuie să arate în felul următor:

```
Introdu primul număr: a=5
Introdu al doilea număr: b=3
Suma a+b=8, Produsul a*b=15
Suma 5+3=8, Produsul 5*3=15
```

Răspuns:

```
a = int(input("Introduceti primul numar: "))
b = int(input("Introduceti primul numar: "))
```

```
    suma = lambda x,y:x+y
    produs = lambda x,y:x*y
    print(f"Suma {a}+{b} = {suma(a,b)}, Produsul {a}+{b} =
{produs(a,b)}")
```

Ex.5.4

Trebuie să realizăm o funcţie care să redea după rulare factorialul numărului 4, iar numărul 4 să fie introdus de utilizator (4x3x2x1=24). După rularea codului răspunsul trebuie să arate în felul următor:

```
Doriti să determinati factorialul numărului: 4
Factorialul lui 4 este 24
```

Răspuns:

```
def fac(num):
    prod = 1
    for i in range(1,num+1):
        prod = prod*i
    print(f"Factorialul lui {num} este : {prod}")

a = int(input("Doriti sa determinati factorialul numarului : "))
fac(a)
```

Ex.5.5

Presupunând că avem creată o listă cu funcţia lista_mea = range(-7,8), trebuie să creăm un cod care după rulare să redea o nouă listă în care toate elementele din lista iniţială să fie ridicate la cub.

Răspuns:

```
def fun(num):
    return num**3
my_list = range(-7,8)
c = list(map(fun,my_list))
print(c)
```

```
[-343, -216, -125, -64, -27, -8, -1, 0, 1, 8, 27, 64, 125, 216, 343]
```

Ex.5.6

Presupunând că avem deja scris următorul cod:

```
def patrat(x):
    return x*x

numere = range(-5,6)
```

folosind expresia lambda trebuie să continuăm acest cod astfel ca la răspuns să obţinem o listă care să conţină toate elementele din variabila numereridicate la pătrat. După rulare trebuie să obţinem următorul răspuns:

```
Pătratul listei
 [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5] este
 [25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25]
```

Răspuns:

```
my_list = range(-5,6)
result  = lambda x : x**2
print("Patratul listei")
for i in my_list:
    print(f" {i} ", end=" ")
print("este")
for i in my_list:
    print(f" {result(i)} ", end=" ")
```

Ex.5.7

Trebuie să creăm 3 funcţii care să redea diametrul, lungimea şi aria unui cerc la răspuns. Raza cercului trebuie să fie introdusă de utilizator. După rulare trebuie să obţinem următorul răspuns:

```
Care este raza cercului : 5
Diametrul : 10
Lungimea : 31.41592653589793
Aria : 78.53981633974483
```

Răspuns:

```
def diam(raza):
print(f"Diametrul : {raza*2}")
```

```
def lung(raza):
print(f"Lungimea : {math.pi*2*raza}")
def aria(raza):
print(f"Aria : {math.pi*(raza**2)}")
r = int(input("Care este raza cercului : "))
diam(r)
lung(r)
aria(r)
```

Ex.5.8

Trebuie să creăm un cod în care să cerem de la utilizat limita minimă (număr negativ) şi limita maximă, din şirul de numere obţinut să obţinem la răspuns o listă care să cuprindă doar numerele pozitive şi pare (trebuie să folosim funcţiile range(), filter()şi expresia lambda. După rulare trebuie să obţinem următorul răspuns:

```
Introdu limita minimă (nr. negativ): -7
Introdu limita maximă (nr. pozitiv): 27
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26]
```

Răspuns:

```
min = int(input("Introdu limita minima(nr.negativ) : "))
max = int(input("Introdu limita maxima(nr.pozitiv) : "))
my_list = []
for i in range(min,max+1):
    my_list.append(i);

my_list2 = list(filter(lambda x : (x%2==0 and x>0),my_list))
print(my_list2)
```

Ex.5.9

Un exemplu asemănător cu cel precedent. Dacă avem lista numere = [-1, 2, 45, 20, -23, 17, -3, 9, - 5, 6, 32, -14, 14, 17, 12, -20, 11, 8] trebuie să scriem un cod folosind funcţia filter()şi expresia lambda astfel încât la răspuns să obţinem două liste noi, prima listă să conţină doar numerele negative şi cele impare, iar a doua listă să conşină numerele mai mari decât -20 şi mai mici decât 20. După rulare trebuie să obţinem următorul răspuns:

```
Nr negative / impare:  [-1, -23, -3, -5]
Nr între -20 si 20:  [-1, 2, 17, -3, 9, -5, 6, -14, 14, 17, 12, 11, 8]
```

Răspuns:

```
min = int(input("Introdu limita minima(nr.negativ) : "))
max = int(input("Introdu limita maxima(nr.pozitiv) : "))
my_list = []
for i in range(min,max+1):
    my_list.append(i);

my_list1 = list(filter(lambda x : (x%2!=0 and x<0),my_list))
my_list2 = list(filter(lambda x : (x>-20 and x<20),my_list))
print(my_list1)
print(my_list2)
```

Ex.5.10

Să presupunem că avem două matrice *xx*=�3985�şi *yy*=�2317�. Trebuie să creăm un cod în care să definim funcţii pentru adunarea (np.add(x,y)), scăderea(np.subtract(x,y)), înmulţirea (np.multiply(x,y)), împărţirea (np. divide(x,y)), înmulţirea scalară (np.dot(x,y)) a matricelor x şi y şi funcţia pentru determinarea transpusei matricei x (x.T) şi inversei matricei y (np.linalg.inv(y)). După rulare trebuie să obţinem următorul răspuns:

```
Suma matricelor x+y=
 [[ 5 12]
 [ 9 12]]
Diferenta matricelor x-y=
 [[ 1  6]
 [ 7 -2]]
Produsul matricelor x*y=
 [[ 6 27]
 [ 8 35]]
Împărtirea matricelor x/y=
 [[1.5        3.        ]
 [8.         0.71428571]]
Transpusa matricei x=
 [[3 8]
 [9 5]]
Inversa matriecei y=
 [[ 0.63636364 -0.27272727]
 [-0.09090909  0.18181818]]
```

Răspuns:

```
x= np.array([[3,9],[8,5]])
y= np.array([[2,3],[1,7]])

print(f"Suma: \n{np.add(x,y)}")
print(f"Diferenta: \n{np.subtract(x,y)}")
print(f"Produsul: \n{np.multiply(x,y)}")
print(f"Impartirea: \n{np.divide(x,y)}")
print(f"Transpusa: \n{np.dot(x,y)}")
print(f"Inversarea matricii y: \n{np.linalg.inv(y)}")
```

Ex.5.11

În acest exerciţiu trebuie să analizaţi 2 coduri şi fără a rula codurile trebuie să vă daţi seama care va fi răspunsul:

a)
```
def suma (x=4, y=-2):
    return x+y

print(suma())
```

b)
```
def suma (x=4, y=-2):
    return x+y

print(suma(5,-5))
```

Răspuns:

a) 2

b) 0

Ex.5.12

În acestexerciţiu trebuie să creăm un cod în care să cerem de la utilizator să introducă limita de jos şi limita de sus. Din acest şir de numere trebuie să cream o listă nouă lista_mea = []în care să se regăsească doar numerele multele cu 4 şi non multiple cu 3.Deoarece suntem la tema despre funcţii codul creat trebuie să se bazeze pe o funcţie creată de noi. La soluţii vor fi date două răspunsuri a) cod care nu se bazează pe funcţie creată de programist b) cod care se bazează pe o funcţie creată de programist . În ambele cazuri după rularea codului trebuie să obţinem următorul răspuns:

```
Introdu limita de jos: -30
Introdu limita de sus: 23
['-28', '-20', '-16', '-8', '-4', '4', '8', '16', '20']
```

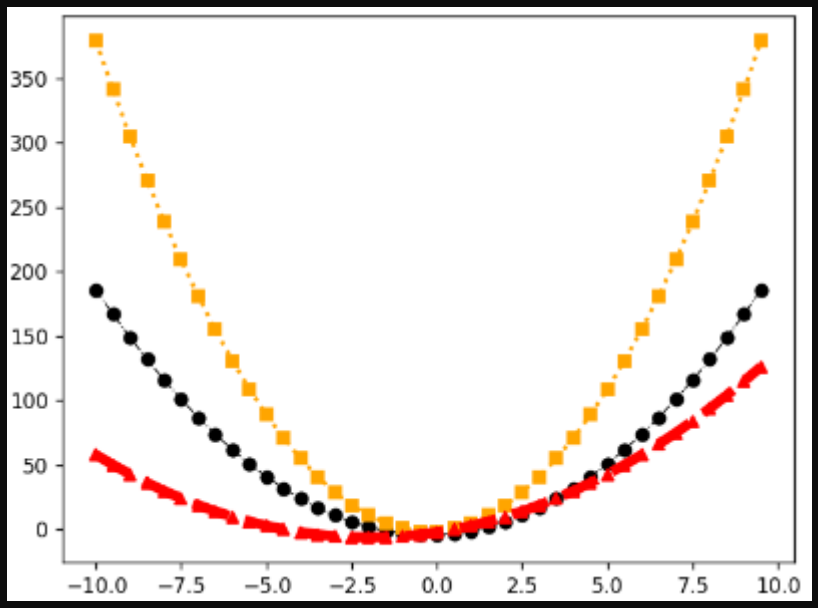
Răspuns:

```
a) def fun(list):
    new_list = []
    for i in list:
        if i%4==0 and i%3!=0:
            new_list.append(i)
    return new_list
min = int(input("Introdu limita minima: "))
max = int(input("Introdu limita maxima: "))
my_list = []
for i in range(min,max+1):
    my_list.append(i)
print(fun(my_list))

b)

min = int(input("Introdu limita minima: "))

max = int(input("Introdu limita maxima: "))

my_list = []

for i in range(min,max+1):

    my_list.append(i)
```

```
result = list(filter(lambda x: (x%4 == 0 and x%3!=0) ,
my_list))print(result)
```

Ex.5.13

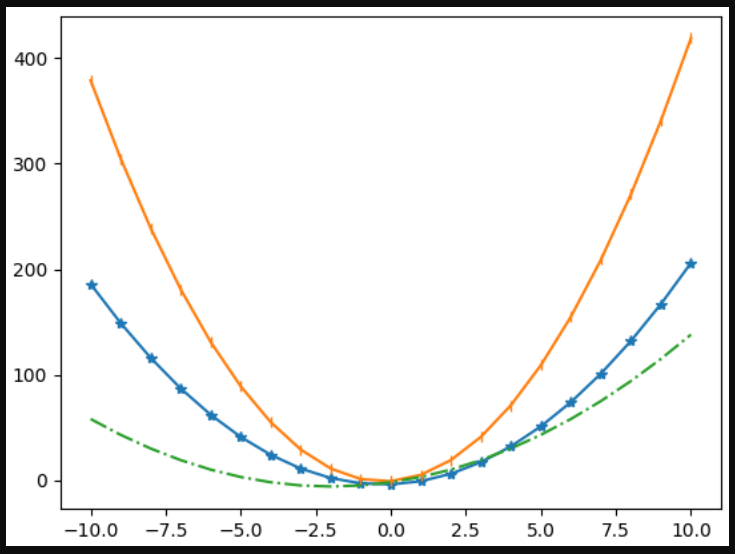
Folosind bibliotecile NumPy și Matplotlib creați în aceeași figură graficele următoarelor funcții $yy=2xx2+xx-4$, $yy=4xx2+2xx-1$ și $yy=xx2+4xx-2$. Liniile celor trei grafice trebuie să fie diferite: culoare, tip linie, grosime linie și marker linie. Dupărularea codului un răspuns care putem să îl obținem poate arăta în felul următor:



Răspuns:

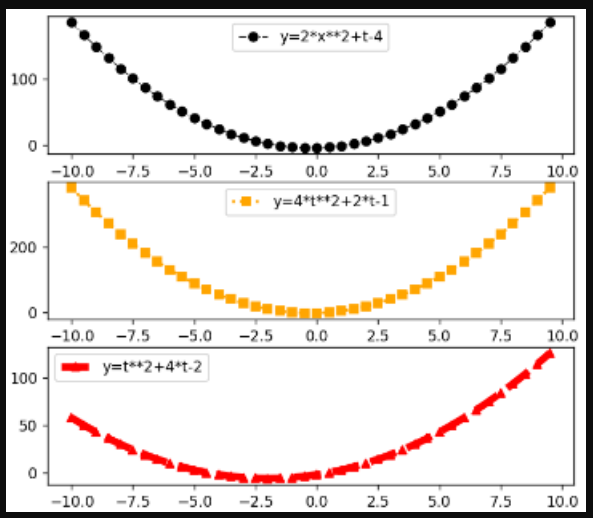
```
import matplotlib.pyplot as plt
import numpy as np

x = np.arange(-10,11)
y_1 = 2*x**2+x-4
y_2 = 4*x**2+2*x-1
y_3 = x**2+4*x-2

plt.plot(x,y_1, '-*')
```

```
plt.plot(x,y_2, '-|')
plt.plot(x,y_3, '-.')
plt.show()
```



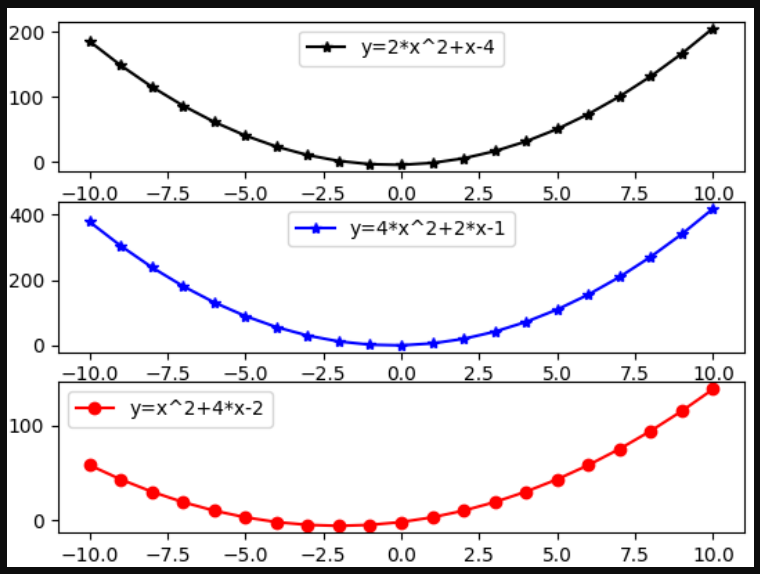
Ex.5.14

Modificaţi codul din exemplul 5_13 astfel încât să obţinem cele trei grafice ale ecuaţiilor *yy=2xx2+xx−4*, *yy=4xx2+2xx−1*şi *yy=xx2+4xx−2*în aceeaşi figură dar ca 3 sub figuri distincte. După rulare trebuie să obţinem următoarea reprezentare grafică:



Răspuns:

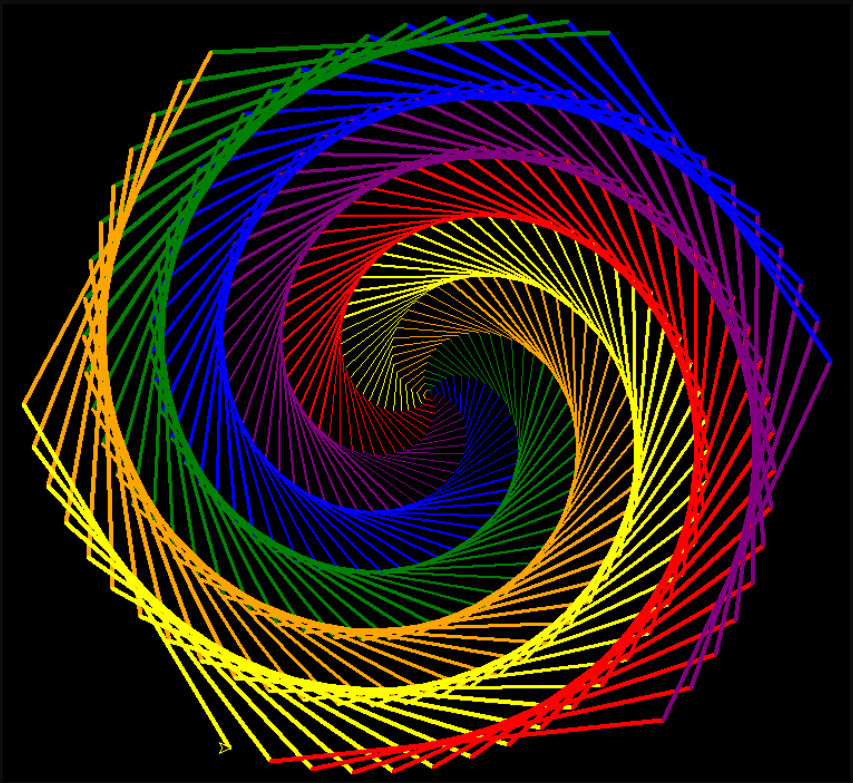
```
import matplotlib.pyplot as plt

import numpy as np

x = np.arange(-10,11)

y_1 = 2*x**2+x-4

y_2 = 4*x**2+2*x-1

y_3 = x**2+4*x-2

plt.subplot(3,1,1)

plt.plot(x,y_1, 'k-*')

plt.legend(['y=2*x^2+x-4'])

plt.subplot(3,1,2)

plt.plot(x,y_2, 'b-*')

plt.legend(['y=4*x^2+2*x-1'])

plt.subplot(3,1,3)

plt.plot(x,y_3, 'r-o')

plt.legend(['y=x^2+4*x-2'])

plt.show()
```



Turtle

Ex.1

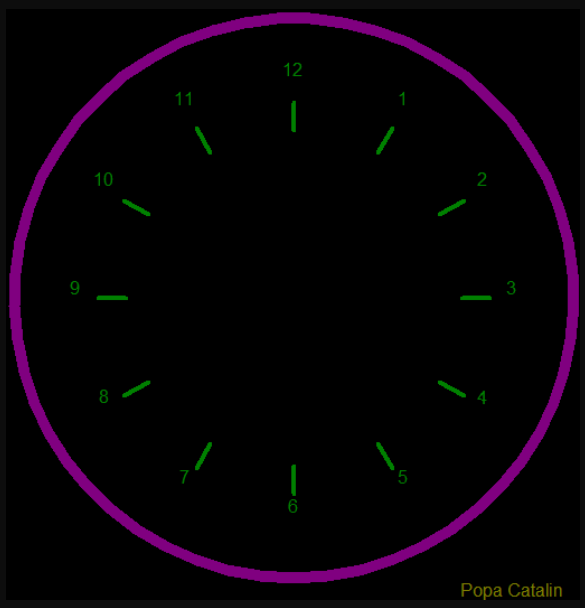
```
import turtle

colors = ['red', 'purple', 'blue', 'green', 'orange', 'yellow']

t = turtle.Pen()

turtle.bgcolor('black')

for x in range(360):

    t.pencolor(colors[x%6])

    t.width(x//100 + 1)

    t.forward(x)

    t.left(59)
```



Ex.2

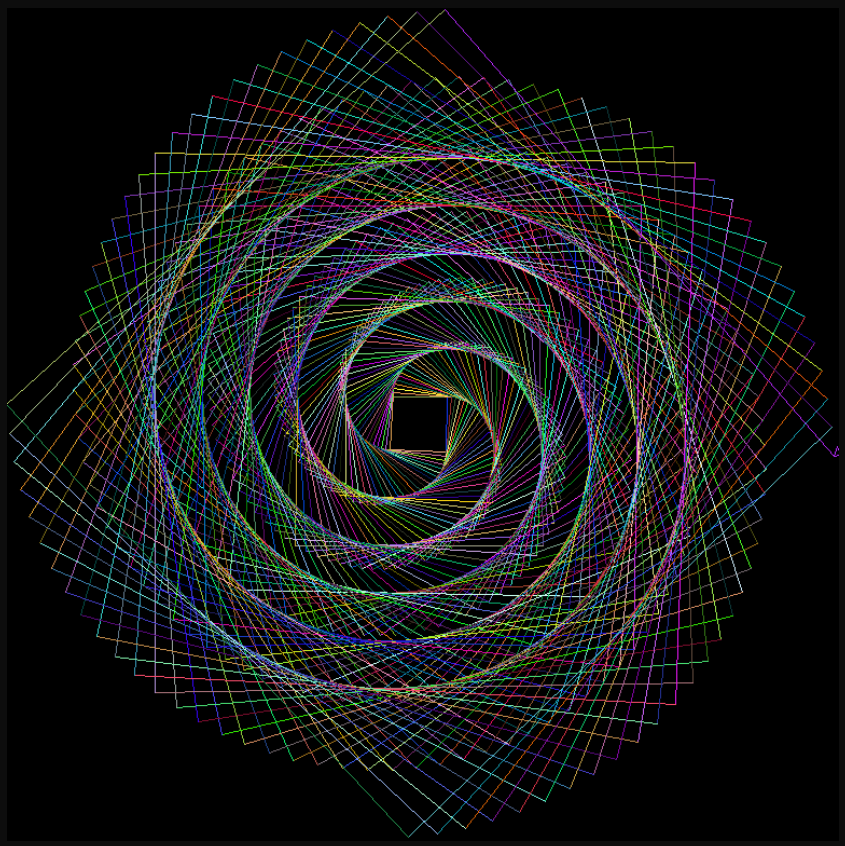
```
from turtle import *
import turtle as tur
import turtle
screen=turtle.Screen()
trtl=turtle.Turtle()
screen.setup(620,620)
screen.bgcolor('black')
clr=['red','green','blue','yellow','purple']
trtl.pensize(4)
trtl.shape('turtle')
trtl.penup()
trtl.pencolor('green')
m=0
for i in range(12):
      m=m+1
      trtl.penup()
      trtl.setheading(-30*i+60)
      trtl.forward(150)
      trtl.pendown()
      trtl.forward(25)
      trtl.penup()
      trtl.forward(20)
      trtl.write(str(m),align="center",font=("Arial", 12, "normal"))
```

```
        if m==12:
          m=0
        trtl.home()
    trtl.home()
    trtl.setpos(0,-250)
    trtl.pendown()
    trtl.pensize(10)
    trtl.pencolor('purple')
    trtl.circle(250)
    trtl.penup()
    trtl.setpos(150,-270)
    trtl.pendown()
    trtl.pencolor('olive')
    trtl.write('Popa Catalin',font=("Arial", 12, "normal"))
    trtl.ht()
    tur.mainloop()
```



Ex.3

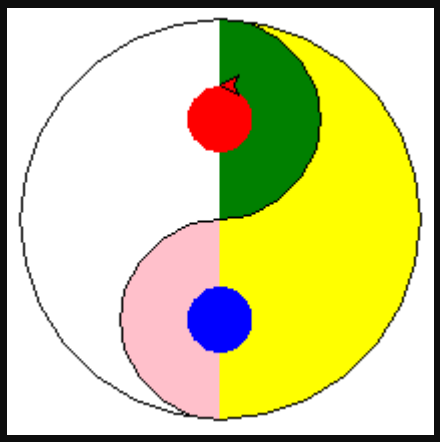
```
import turtle

import random

turtle.bgcolor('black')

turtle.colormode(255)

x=0

turtle.speed(0)

for x in range(500):

    r,b,g=random.randint(0,255),random.randint(0,255),random.randint(0,255)

    turtle.pencolor(r,g,b)

    turtle.fd(x+50)

    turtle.rt(91)

turtle.exitonclick()
```



yin yang

```
from turtle import *

RAD = 100

RAD2 = RAD / 2

RAD6 = RAD / 6

degrees()

fillcolor('yellow')

begin_fill()

circle(RAD, 180)

end_fill()

circle(RAD, 180)

left(180)

penup()

goto(0, RAD)

pendown()

fillcolor('pink')

begin_fill()

circle(RAD2, 180)

end_fill()

penup()

goto(0, RAD)

pendown()

fillcolor('green')
```

```
begin_fill()

circle(RAD2, 180)

end_fill()

penup()

goto(0, RAD2 + RAD6)

fillcolor('blue')

begin_fill()

circle(RAD6)

end_fill()

fillcolor('red')

goto(0, 2 * (RAD - RAD6))

begin_fill()

circle(RAD6)

end_fill()
```



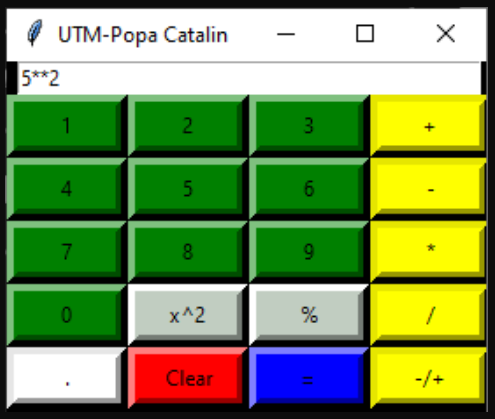
Calculatorul

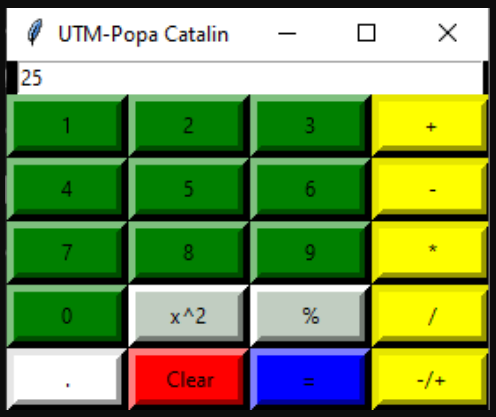
```
from tkinter import *

expression = ""

def apasa(num):

    global expression

    expression = expression + str(num)

    equation.set(expression)



def equalpress():

    try:



        global expression

        total = str(eval(expression))

        equation.set(total)

        expression = ""

    except:



        equation.set(" error ")

        expression = ""



def minus_plus():

    expression = (-abs(expression))



def clear():

    global expression
```

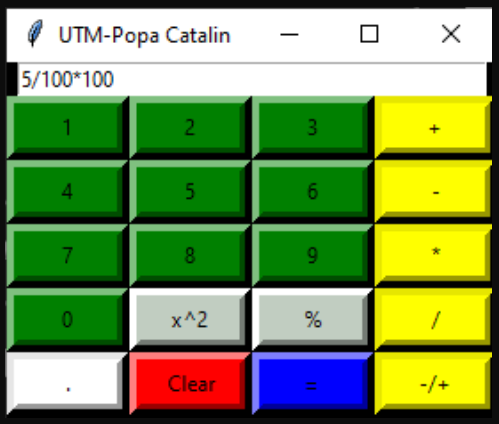
```
        expression = ""
        equation.set("")


    if __name__ == "__main__":
        gui = Tk()
        gui.configure(background="black")
        gui.title("UTM-Popa Catalin")
        gui.geometry("273x200")
        equation = StringVar()
        expression_field = Entry(gui, textvariable=equation)
        expression_field.grid(columnspan=4, ipadx=70)


        button1 = Button(gui, text=' 1 ', fg='black',bd =7, bg='green',
                         command=lambda: apasa(1), height=1, width=7)
        button1.grid(row=2, column=0)
        button2 = Button(gui, text=' 2 ', fg='black',bd =7, bg='green',
                         command=lambda: apasa(2), height=1, width=7)
        button2.grid(row=2, column=1)
        button3 = Button(gui, text=' 3 ', fg='black',bd =7, bg='green',
                         command=lambda: apasa(3), height=1, width=7)
        button3.grid(row=2, column=2)
        button4 = Button(gui, text=' 4 ', fg='black',bd =7, bg='green',
                         command=lambda: apasa(4), height=1, width=7)
        button4.grid(row=3, column=0)
        button5 = Button(gui, text=' 5 ', fg='black',bd =7, bg='green',
                         command=lambda: apasa(5), height=1, width=7)
```

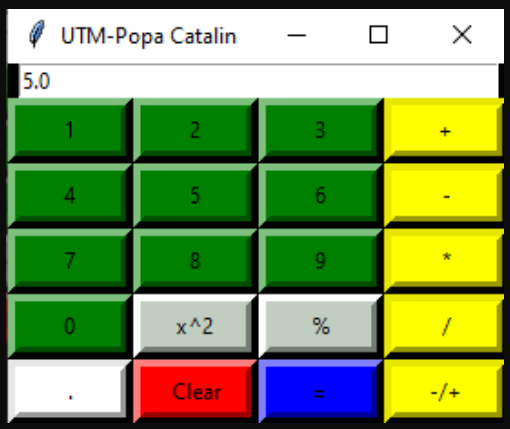
```
button5.grid(row=3, column=1)

button6 = Button(gui, text=' 6 ', fg='black',bd =7, bg='green',

                 command=lambda: apasa(6), height=1, width=7)

button6.grid(row=3, column=2)

button7 = Button(gui, text=' 7 ', fg='black',bd =7, bg='green',

                 command=lambda: apasa(7), height=1, width=7)

button7.grid(row=4, column=0)

button8 = Button(gui, text=' 8 ', fg='black',bd =7, bg='green',

                 command=lambda: apasa(8), height=1, width=7)

button8.grid(row=4, column=1)

button9 = Button(gui, text=' 9 ', fg='black',bd =7, bg='green',

                 command=lambda: apasa(9), height=1, width=7)

button9.grid(row=4, column=2)

button0 = Button(gui, text=' 0 ', fg='black',bd =7, bg='green',

                 command=lambda: apasa(0), height=1, width=7)

button0.grid(row=5, column=0)

plus = Button(gui, text=' + ', fg='black',bd =7, bg='yellow',

            command=lambda: apasa("+"), height=1, width=7)

plus.grid(row=2, column=3)

minus = Button(gui, text=' - ', fg='black',bd =7, bg='yellow',

            command=lambda: apasa("-"), height=1, width=7)

minus.grid(row=3, column=3)

multiply = Button(gui, text=' * ', fg='black',bd =7, bg='yellow',

                 command=lambda: apasa("*"), height=1, width=7)

multiply.grid(row=4, column=3)

divide = Button(gui, text=' / ', fg='black',bd =7, bg='yellow',
```

```
                command=lambda: apasa("/"), height=1, width=7)
divide.grid(row=5, column=3)
btnPercentage=Button(gui, text="%", fg='black',bd =7,bg="Honeydew3",
                command=lambda: apasa("/100*"),height=1, width=7)
btnPercentage.grid(row=5,column=2)
btnpower=Button(gui, text="x^2", fg='black',bd =7,bg="Honeydew3",
           command=lambda: apasa("**2"),height=1, width=7)
btnpower.grid(row=5,column=1)
equal = Button(gui, text=' -/+ ', fg='black',bd =7, bg='yellow',
            command=minus_plus, height=1, width=7)
equal.grid(row=6, column=3)
equal = Button(gui, text=' = ', fg='black',bd =7, bg='blue',
            command=equalpress, height=1, width=7)
equal.grid(row=6, column=2)
clear = Button(gui, text='Clear', fg='black',bd =7, bg='red',
            command=clear, height=1, width=7)
clear.grid(row=6, column='1')
Decimal= Button(gui, text='.', fg='black',bd =7, bg='white',
                command=lambda: apasa('.'), height=1, width=7)
Decimal.grid(row=6, column=0)
gui.mainloop()
```


UTM-Popa Catalin
5**2
1
2
3
+
4
5
6
-
7
8
9
*
0
x^2
%
/
.
Clear
=
-/+


UTM-Popa Catalin
25
1
2
3
+
4
5
6
-
7
8
9
*
0
x^2
%
/
.
Clear
=
-/+


UTM-Popa Catalin
5/100*100
1
2
3
+
4
5
6
-
7
8
9
*
0
x^2
%
/
.
Clear
=
-/+


UTM-Popa Catalin
5.0
1
2
3
+
4
5
6
-
7
8
9
*
0
x^2
%
/
.
Clear
=
-/+