MINISTERUL EDUCAȚIEI, CULTURII și CERCETĂRII al Rep. MOLDOVA
UNIVERSITATEA TEHNICĂ a MOLDOVEI
FACULTATEA CALCULATOARE, INFORMATICĂ și MICROELECTRONICĂ
Departamentul „Ingineria Software si Automatica”

LUCRARE INDIVIDUALĂ
GRAFICA PE CALCULATOR

Student(ă): Cojocari Dragos
gr.TI-214, FCIM
Conducător:
Conf. univ. MALCOCI Iulian

CHIȘINĂU 2022

CUPRINS

Tema 1. Operații matematice de bază. Variabile

Ex1_1

a) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
y = 20
y = 15
y = 30
print (y)
print (type(y))
```

b) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
y = 20
y = 15
y = 30
# print (y)
# print (type(y))
x = 5
z = y / x
print (z)
print (type(z))
```

Răspuns **Ex1_1**:

a)

```
30
<class 'int'>
```

b)

```
6.0
<class 'float'>
```

Ex1_2

a) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
print ("Vasile", "are", 30, "de ani")

# print ("Vasile", "are", 30, "de ani", end = " ... ")

#print ("Vasile", "Ion", "Ana", sep = " || ", end = " : Cine e fata?")

#print ()

#print ("Fata este Ana!!!")
```

b) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
print ("Vasile", "are", 30, "de ani")

print ("Vasile", "are", 30, "de ani", end = " ... ")

#print ("Vasile", "Ion", "Ana", sep = " || ", end = " : Cine e fata?")

#print ()

#print ("Fata este Ana!!!")
```

c) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
print ("Vasile", "are", 30, "de ani")

print ("Vasile", "are", 30, "de ani", end = " ... ")

print ("Vasile", "Ion", "Ana", sep = " || ", end = " : Cine e fata?")

#print ()

#print ("Fata este Ana!!!")
```

d) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
print ("Vasile", "are", 30, "de ani")

print ("Vasile", "are", 30, "de ani", end = " ... ")

print ("Vasile", "Ion", "Ana", sep = " || ", end = " : Cine e fata?")

print ()

#print ("Fata este Ana!!!")
```

e) Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
print ("Vasile", "are", 30, "de ani")

print ("Vasile", "are", 30, "de ani", end = " ... ")

print ("Vasile", "Ion", "Ana", sep = " || ", end = " : Cine e fata?")

print ()

print ("Fata este Ana!!!")
```

Raspuns ex 1_2

A)
```
Vasile are 30 de ani
```

B)
```
Vasile are 30 de ani
Vasile are 30 de ani ...
```

C)
```
Vasile are 30 de ani
Vasile are 30 de ani ... Vasile || Ion || Ana : Cine e fata?
```

D)
```
Vasile are 30 de ani
Vasile are 30 de ani ... Vasile || Ion || Ana : Cine e fata?
```

E)
```
Vasile are 30 de ani
Vasile are 30 de ani ... Vasile || Ion || Ana : Cine e fata?

Fata este Ana!!!
```

Ex1_3

Scrieți codul pentru a determina aria și perimetrul unui dreptunghi dacă știm că lungimea este a = 10 cm, iar lățimea b = 4 cm. Aria: $A = a \times b$ iar Perimetrul: $P = 2(a + b)$.

Raspuns ex 1_3

```
a = 10
b = 4

print("Aria dreptunghiului este:",a*b)
print("Perimetrul dreptunghiului este:",2*(a+b))
```

```
Aria dreptunghiului este: 40
Perimetrul dreptunghiului este: 28
```

Ex1_4

Care va fi răspunsul obținut după rularea codului în fereastra de program IDLE:

```
x = 25
y = 20
z = 3
f = 2

a = (x - y) * z / f

b = ((x - y) * z) / f

c = (x - y) * (z / f)

print ("a = ", a)
print ("b = ", b)
print ("c = ", c)
```

Raspuns Ex1_4

```
a =  7.5
b =  7.5
c =  7.5
```

Ex1_5

Scrieți un cod în care să verificați daca un număr este par sau impar.

```
a = int(input("Dati un numar:"))

print("Numarul este par") if a % 2 == 0 else print("Numarul este impar")
```

Raspuns Ex1_5

```
Dati un numar:3
Numarul este impar

C:\Users\xccelerato
Dati un numar:2
Numarul este par
```

Ex1_6

Scrieți un cod în care în primă fază să ridicați un număr la pătrar (exponentul 2) după care să extrageți rădăcina pătrată din el ca să obțineți la răspuns numărul inițial.

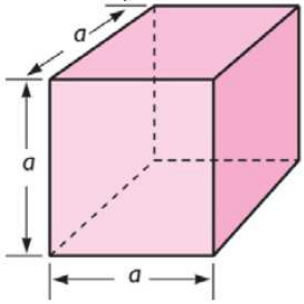
Raspuns Ex1_6

```
import math

a = 3

a = a**2

print("Numarul ridicat la puterea a 2-a",a)

print("Radicalul din numarul ridicat la puterea a 2-a",int(math.sqrt(a)))
```

```
Numarul ridicat la puterea a 2-a 9
Radicalul din numarul ridicat la puterea a 2-a 3
```

Ex1_7

Scrieți codul pentru a determina Aria: $A = 6a^2$ și Volumul: $V=a^3$ unui cub cu latura $a=20$ cm.



Raspuns Ex1_7

```
a = 20

print("Aria cubului este:",6*a**2)
print("Volumul cubului este:",a**3)
```

```
Aria cubului este: 2400
Volumul cubului este: 8000
```

Ex1_8

Scrieți un cod în 3 moduri diferite care să ne dea rezultatul când baza este 4 și exponentul este 3, 4^3=?

Raspuns Ex1_8

```
print("4**3 = ",4**3)
print("4*4*4 = ",4*4*4)
print("pow(4,3) = ",pow(4,3))
```

```
4**3 =  64
4*4*4 =  64
pow(4,3) =  64
```

Ex1_9

Scrieți un cod care să rezolve următoarea ecuație: z = |x – y| * (x + y) dacă știm că x = 4, iar y = 6.

```
x = 4
y = 6

z = abs(x - y) * (x + y)
print(z)
```

```
20
```

Ex1_10

Numele variabilelor. În acest exercițiu vom învăța să dăm nume corecte variabilelor.
Reguli de bază pentru numele variabilelor în Python:
- numele unei variabile poate să înceapă cu literă sau simbolul „_”. Exemple:
continent_ sau _continent
- numele unei variabile poate să conțină litere, cifre și simbolul „_”. Exemple:
martie_13 sau martie13
- numele unei variabile trebuie să conțină aceleași litere pe parcursul întregului cod:
Martie2020 și martie2020 vor fi două variabile diferite în Python
- nu este permis de folosit „cuvinte-cheie” în numele unei variabile. Exemple de „cuvinte-cheie”:
`False`, `def`, `if`, `raise`, `None`, `del`, `import`, `return`, `True`, `elif`, `in`, `try`, `and`, `else`, `is`, `while`, `as`, `except`, `lambda`, `with`, `assert`, `finally`, `nonlocal`, `yield`, `break`, `for`, `not`.

Analizați codul de mai jos, copiați-l în fereastra de program IDLE și încercați să rulați acest cod. O să apară multe erori, sarcina dumneavoastră este să eliminați din cod și din funcția `print ( )` toate variabilele care sunt greșite astfel încât în final codul să poată fi rulat.

```
salariu = 20 # $ pe oră
Salariu = 22
for = 7
salariu = 17
SALARIU = 14
s_a_l_a_r_i_u = 15
2salariu = 10
_salariu = 30
False = 21
salariu_ = 8
_SALARIU_ = 27

print (salariu, Salariu, for, salariu, SALARIU, s_a_l_a_r_i_u, 2salariu, False, salariu_, _SALARIU_)
```

Raspuns Ex1_10

```
salariu = 20 # $ pe oră
Salariu = 22
salariu = 17
SALARIU = 14
s_a_l_a_r_i_u = 15
_salariu = 30
salariu = 8
_SALARIU_ = 27

print(salariu, Salariu, salariu, Salariu, s_a_l_a_r_i_u, _SALARIU_)
```

```
8 22 8 22 15 27
```

Ex1_11

Scrieți un cod care să transforme gradele Fahrenheit în grade Celsius și invers. Formula de transformare este: F = 9/5*C+32

Raspuns Ex1_11

```
c = int(input("Dati gradele in Celsius:"))

print(c,"grade Celsius in Fahrenheit:", 9/5*c+32)

f = int(input("Dati gradele in Fahrenheit:"))
print(f,"grade Fahrenhei in Celsius:", (f-32)*5/9)
```

```
Dati gradele in Celsius:100
100 grade Celsius in Fahrenheit: 212.0
Dati gradele in Fahrenheit:212
212 grade Fahrenhei in Celsius: 100.0
```

Ex1_12

În ultimul exercițiu trebuie să transformați MPH (mile pe oră) în KMH (kilometri pe oră) și invers. Formulele de calcul vor fi: MPH = 0.6214 * KMH și KMH = 1.6093 * MPH

Raspuns Ex1_12

```
mph = float(input("Dati viteza in MPH:"))
print(mph,"MPH in KMH este:",1.6093*mph)

kmh = float(input("Dati viteza in kmh:"))
print(kmh,"KMH in MPH este:",0.6214*kmh)
```

```
Dati viteza in MPH:100
100.0 MPH in KMH este: 160.93
Dati viteza in kmh:160.93
160.93 KMH in MPH este: 100.001902
```

Ex2_1

Acest exercițiu l-am rezolvat și în capitolul 1 la Ex1_12 când am transformat MPH în KMH. Acum ne propunem același lucru doar că valoare care vrem s-o transformăm să fie introdusă de utilizator, prin urmare în codul nostru trebuie să fie prezentă și funcția `input ( )`. Scrieți codul astfel încât răspunsul obținut să arate în felul următor:

```
Care este viteza în km/h?: 30
Viteza exprimată în km/h este egală cu 30 KM/H
Viteza exprimată în mph este egală cu? 18.642 MPH
```

Raspuns Ex2_1

```
kmh = int(input("Care este viteza in km/h?:"))

print("Viteza exprimata in km/h este egala cu",int(kmh),"KM/H")
print("Viteza exptimata in mph este egala cu?",0.6214*int(kmh),"MPH")
```

```
Care este viteza in km/h?:30
Viteza exprimata in km/h este egala cu 30 KM/H
Viteza exptimata in mph este egala cu? 18.642 MPH
```

Ex2_2

Studiu individual. În acest exercițiu la răspuns trebuie să obținem calendarul pentru această lună. Drept exemplu vom lua luna septembrie din anul 2021. Pentru a obține acest rezultat aveți nevoie de următorul modul `calendar`, care trebuie importat în `Python` (Sugestie: `calendar.month(anul, luna)`. Cum trebuie să arate codul?, pentru ca la rezultat să obținem:

```
    September 2021
Mo Tu We Th Fr Sa Su
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

Raspuns Ex2_2

```
import calendar

print(calendar.month(2022,3))
```

```
     March 2022
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

Ex2_3

Exercițiul curent este continuarea celui precedent, trebuie să modificăm codul astfel încât anul și luna care ne interesează să fie introduse de către utilizator. Să presupunem că dorim să aflăm care a fost ziua săptămânii în care ne-am născut. Eu am ales în acest exemplu anul și luna mea de naștere și din calendarul apărut pot să aflu această informație.

Răspunsul trebuie să arate în felul următor:

```
Care este anul care vă interesează? [ex. 2007]: 1980
Care este luna care vă interesează? [ex. 04]: 04
     April 1980
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Raspuns Ex2_3

```
import calendar

yy = int(input("Care este anul care va intereseaza? [ex. 2007]:"))
mm = int(input("Care este luna care va intereseaza? [ex. 04]:"))

print(calendar.month(yy,mm))
```

```
Care este anul care va intereseaza? [ex. 2007]:1980
Care este luna care va intereseaza? [ex. 04]:04
     April 1980
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

Ex2_4

În acest exercițiu simplu se cere ca utilizatorul să introducă variabilele **a** și **b**, iar la răspuns să obținem rezultatul de la operațiile de adunare, scădere, înmulțire și împărțire. Rezultatul trebuie să arate în felul următor (încercați să folosiți metode diferite de formatare a șirurilor):

```
Care este valoarea pentru nr. a? a = 4
Care este valoarea pentru nr. b? b = 6

a + b = 10, a - b = -2

a * b = 24, a : b = 0.6666666666666666
```

Raspuns Ex2_4

```
a = int(input("Care este valoare pentru nr. a? a = "))
b = int(input("Care este valoare pentru nr. b? b = "))

print("a + b =",a+b,end=",")
print(" a - b =",a-b)
print("a * b =",a*b,end=",")
print(" a : b =",a/b)
```

```
Care este valoare pentru nr. a? a = 4
Care este valoare pentru nr. b? b = 6
a + b = 10, a - b = -2
a * b = 24, a : b = 0.6666666666666666
```

Ex2_5

În acest exercițiu trebuie să cerem utilizatorului un număr oarecare, iar la răspuns să obținem doar numărul introdus de utilizator, pătratul acestui număr și cubul acestui număr. Răspunsul trebuie să arate în felul următor:

```
Care este valoarea pentru nr. a? a = 4

4 16 64
```

Raspuns Ex2_5

```
a = int(input("Care este valoare pentru nr. a? a = "))

print(a,a**2,a**3)
```

```
Care este valoare pentru nr. a? a = 4
4 16 64
```

Ex2_6

În acest exercițiu trebuie să cerem de la utilizator care este temperatura în grade Celsius, iar la răspuns să obținem cât este temperatura în grade Celsius și cât este această temperatură în grade Fahrenheit. Dacă ați uitat relațiile de transformare revedeți Ex1_11 din Capitolul 1. Răspunsul trebuie să arate în felul următor:

```
Care este temperatura în grade Celsius?: 27
Temperatura în grade Celsius este 27 grade C
Temperatura în grade Fahrenheit este 80.6 grade F
```

Raspuns Ex 2_6

```
c = int(input("Care este temperatura in grade Celsius?: "))

print("Temperatura in grade Celsius este",c,"grade C")
print("Temperatura in grade Fahrenheit este",9/5*c+32,"grade F")
```

```
Care este temperatura in grade Celsius?: 27
Temperatura in grade Celsius este 27 grade C
Temperatura in grade Fahrenheit este 80.6 grade F
```

Ex2_7

În acest exercițiu trebuie să scriem un cod care să ceară de la utilizator lungimea și lățimea unui dreptunghi, și programul să calculeze aria și perimetrul dreptunghiului cu aceste valori. Răspunsul trebuie să arate în felul următor:

```
************ PROGRAM ************
        ARIA & PERIMETRUL
          DREPTUNGHI
Care este lungimea dreptungiului?: 6
Care este lățimea dreptunghiului?: 4

ARIA = 24, PERIMETRUL = 20
```

Raspuns Ex2_7

```
print("*"*12," PROGRAM ","*"*12)
print("\tARIA & PERIMETRUL")
print("\t  DREPTUNGHI")

lung = int(input("Care este lungimea dreptunghiului?: "))
lat = int(input("Care este latimea dreptunghiului?: "))

print("ARIA = ",lung*lat,end=",")
print(" PERIMETRUL = ",2*(lung+lat))
```

```
************  PROGRAM  ************
        ARIA & PERIMETRUL
          DREPTUNGHI
Care este lungimea dreptunghiului?: 6
Care este latimea dreptunghiului?: 4
ARIA =  24, PERIMETRUL =  20
```

Ex2_8

În acest exercițiu va trebui să scriem un cod care să ceară de la utilizator raza unui cerc și programul să calculeze diametrul, lungimea circumferinței și aria cercului: D = 2R; L = 2πR; A = πR^2.

Ca să nu ne complicăm putem aproxima $\pi \cong 3.14$. Dar in Python avem la dispoziție modulul math de care trebuie să vă folosiți atunci când veți scrie codul pentru acest exercițiu.

Răspunsul trebuie să arate în felul următor:

```
|||||||||| PROGRAM ||||||||||
           CERCUL
       LUNGIME & ARIA
Care este raza cercului?: 5

Diametrul cercului este:  10
Lungimea cercului este:  31.41592653589793
Aria cercului este:  78.53981633974483
```

Raspuns Ex2_8

```
import math

print("|"*9,"PROGRAM","|"*9)
print("\t  CERCUL")
print("\tLUNGIME & ARIA")

raza = int(input("Care este raza cercului?: "))

print("Diametrul cercului este:",2*raza)
print("Lungimea cercului este:",2*math.pi*raza)
print("Aria cercului este:",math.pi*raza**2)
```

```
||||||||| PROGRAM |||||||||
          CERCUL
        LUNGIME & ARIA
Care este raza cercului?: 5
Diametrul cercului este: 10
Lungimea cercului este: 31.415926535897932
Aria cercului este: 78.53981633974483
```

Ex2_9

Analizați cu atenție codul și încercați să depistați care sunt erorile comise. La rularea acestui cod vor apărea mesaje de eroare. Sarcina dumneavoastră este să înlăturați greșelile depistate astfel încât codul să ruleze fără erori.

```
nume = "Andrei"
Vârsta = 37
Angajat = True
print(Nume, vârsta, angajat)
```

Raspuns Ex2_9

```
nume = "Andrei"
Varsta = 37
Angajat = True
print(nume, Varsta, Angajat)
```

```
Andrei 37 True
```

Ex2_10

În acest exercițiu se cere să verificați corectitudinea codului și dacă depistați erori să le eliminați astfel încât codul să ruleze fără erori.

```
x = '20'
y = 4
print (x - y)
```

Raspuns Ex2_10

```
x = '20'
y = 4
print(int(x) - y)
```

```
16
```

Ex2_11

În acest exercițiu trebuie să îl întrebați pe utilizator `Ce planuri ai pentru diseară?` cu ajutorul funcției `input ( )`. Răspunsul tastat de utilizator trebuie să îl prezentați sub formă de listă cu ajutorul comenzii `split ( )`. Un exemplu de răspuns:

```
Ce planuri ai pentru diseară?: Diseară mă duc la cinema
Utilizatorul a introdus cuvintele: ['Diseară', 'mă', 'duc', 'la', 'cinema']
```

Rapsuns Ex2_11

```
text = input("Ce planuri ai pentru diseara?:")

print("Utilizatorul a introdus cuvintele:",text.split())
```

```
Ce planuri ai pentru diseara?:Diseara ma duc la cinema
Utilizatorul a introdus cuvintele: ['Diseara', 'ma', 'duc', 'la', 'cinema']
```

Ex2_12

În acest exercițiu trebuie să îl întrebați pe utilizator `Care este mâncarea ta preferată?` cu ajutorul funcției `input ( )`. Răspunsul tastat de utilizator trebuie să îl prezentați doar cu litere majuscule cu ajutorul comenzii `upper ( )`. Un exemplu de răspuns:

```
Care este mâncarea ta preferată?: Pizza
Mâncarea ta preferată este: PIZZA
```

Raspuns Ex2_12

```
text = input("Care este mancarea ta preferata?:")

print("Mancarea ta preferata este:",text.upper())
```

```
Care este mancarea ta preferata?:Pizza
Mancarea ta preferata este: PIZZA
```

Ex2_13

În acest exercițiu trebuie să îi cereți utilizatorului să introducă adresa sa de e-mail (în format prenume.nume@utm.md), iar la răspuns să fie afișat doar numele său (sau ce este scris înaintea simbolului @). Un exemplu de răspuns:

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
iulian.malcoci
```

Raspuns Ex2_13

```
email = input("Introdu adresa de e-mail?: ")
impartit = email.split("@")
print(impartit[0])
```

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
iulian.malcoci
```

Ex2_14

Vom complica exemplul precedent Ex2_13. La fel vom cere utilizatorului să introducă adresa sa de email (în format prenume.nume@utm.md) dar la răspuns să obținem mesajul `Salut Prenume Nume`. După cum observați prenumele și numele sunt tastate cu litere mici, noi trebuie să obținem răspunsul cu prima literă majusculă de la prenume și nume. Un exemplu de răspuns:

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
Salut Iulian Malcoci
```

Raspuns Ex2_14

```
email = input("Introdu adresa de e-mail?: ")
impartit = email.split("@")
numeprenume = impartit[0].split(".")
print("Salut",numeprenume[0].capitalize(),numeprenume[1].capitalize())
```

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
Salut Iulian Malcoci
```

Ex2_15

În acest exercițiu cerem de la utilizator să introducă prenumele, numele și vârsta cu ajutorul funcției `input ( )`. Răspunsul trebuie să arate în felul următor:

```
Introdu prenumele: Iulian
Introdu nume: Malcoci
Introdu vârsta: 41
Salut Iulian Malcoci  ai împlinit 41 ani!
```

Raspuns Ex2_15

```
prenume = input("Introdu prenumele: ")
nume = input("Introdu nume: ")
age = int(input("Introdu varsta: "))

print("Salut",prenume,nume,"ai implinit",age,"ani!")
```

```
Introdu prenumele: Iulian
Introdu nume: Malcoci
Introdu varsta: 41
Salut Iulian Malcoci ai implinit 41 ani!
```

Ex2_16

În acest exercițiu trebuie să îi cereți utilizatorului să introducă adresa sa de e-mail (în format prenume.nume@utm.md), iar la răspuns trebuie să obținem din câte caractere este format numele și prenumele din adresa de e-mail. Sugestie: trebuie să folosiți comanda `len ( )`. Un exemplu de răspuns:

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
Prenumele si numele din adresa contin 14 caractere
```

Raspuns Ex2_16

```
email = input("Introdu adresa de e-mail?: ")

impartit = email.split("@")

print("Prenumele si numele din adresa contin",len(impartit[0]),"caractere")
```

```
Introdu adresa de e-mail?: iulian.malcoci@bpm.utm.md
Prenumele si numele din adresa contin 14 caractere
```

Ex2_17

În acest exemplu să presupunem că avem definită variabila `scriitor = 'Liviu Rebreanu'`. La răspuns vrem să apară doar prenumele `Liviu`. Din cele studiate până în prezent putem obține acest răspuns în două moduri. Un exemplu de răspuns:

```
Liviu
Liviu
```

Raspuns Ex2_17

```
scriitor = 'Liviu Rebreanu'
impartit = scriitor.split()
print(impartit[0])

pos = scriitor.find(" ")
print(scriitor[0:pos])
```

```
Liviu
Liviu
```

Ex3_1

Dacă avem lista `nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']`. Scrieți codul care după rulare să ne dea următorul răspuns:

```
Ion
Olga
Stas
```

```
nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']
print(nume[0])
print(nume[1])
print(nume[-1])
```

Ex3_2

Dacă avem lista `nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']`. Scrieți codul care după rulare să ne dea următorul răspuns:

```
['Ion', 'Olga', 'Stas']
```

```
nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']
nume2 = []

nume2.append(nume[0])
nume2.append(nume[1])
nume2.append(nume[-1])

print(nume2)
```

Ex3_3

Dacă avem lista `nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']`. Scrieți codul care după rulare să ne dea următorul răspuns:

a)
```
['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']
```
b)
```
['Olga', 'Vicu', 'Ana']
```
c)
```
['Ion', 'Olga', 'Vicu']
```
d)
```
['Eva', 'Vlad', 'Stas']
```
e)
```
['Ion', 'Vicu', 'Eva', 'Stas']
```

```
nume = ['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']
print(nume)
print(nume[1:4])
print(nume[0:3])
print(nume[0:3])
```

```
print(nume[4:7])
print(nume[0:8:2])
```

```
['Ion', 'Olga', 'Vicu', 'Ana', 'Eva', 'Vlad', 'Stas']
['Olga', 'Vicu', 'Ana']
['Ion', 'Olga', 'Vicu']
['Ion', 'Olga', 'Vicu']
['Eva', 'Vlad', 'Stas']
['Ion', 'Vicu', 'Eva', 'Stas']
```

Ex3_4

Dacă avem lista `lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']`. Scrieți codul care după rulare să ne dea următorul răspuns:

a)

```
Petru
```

b)

```
['Ion', 'Vlad', 'Petru']
```

c)

```
Ala
```

```
lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
print(lista_mea[1][2])
print(lista_mea[1])
print(lista_mea[-1])
```

```
Petru
['Ion', 'Vlad', 'Petru']
Ala
```

Ex3_5

Dacă avem lista `lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']`. Scrieți codul care după rulare să ne dea următorul răspuns:

a) Să obținem lungimea listei (numărul de elemente din listă) cu ajutorul, cu ajutorul funcției `len()`;

b) Să adăugăm elementul `'Iulian'` la sfârșitul listei. Răspunsul va arăta în felul următor:

```
['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala', 'Iulian']
```

c) Să adăugăm elementul `'Iulian'` la începutul listei. Răspunsul va arăta în felul următor:

```
['Iulian', 'Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
```

```
lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']

for i in lista_mea:
   print(len(i), end=" ")

lista_mea.append("Iulian")
print("\n",lista_mea)
lista_mea.pop(4); lista_mea.insert(0,"Iulian")
print(lista_mea)
```

```
4 3 3 3
 ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala', 'Iulian']
['Iulian', 'Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
```

Ex3_6

Dacă avem lista `lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']`. Scrieți codul în 3 moduri diferite care după rulare să elimine elementul `'Ana'` din listă. După rulare trebuie să obținem următorul răspuns:

`['Vera', ['Ion', 'Vlad', 'Petru'], 'Ala']`

```
lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
lista_mea.pop(2)
print(lista_mea)

lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
lista_mea.remove('Ana')
print(lista_mea)

lista_mea = ['Vera', ['Ion', 'Vlad', 'Petru'], 'Ana', 'Ala']
del lista_mea[2]
print(lista_mea)
```

```
['Vera', ['Ion', 'Vlad', 'Petru'], 'Ala']
['Vera', ['Ion', 'Vlad', 'Petru'], 'Ala']
['Vera', ['Ion', 'Vlad', 'Petru'], 'Ala']
```

Ex3_7

Dacă avem lista my_list = `[77, 11, 22, 99, 33, 55]`. Scrieți un cod care la răspuns să ne dea această listă aranjată în ordine crescătoare, descrescătoare și inversată. Folosiți `.sort()` `.reverse()` și `.sort(reverse = True)`.

```
my_list = [77,11,22,99,33,55]
my_list.sort()
print(my_list)

my_list = [77,11,22,99,33,55]
my_list.sort(reverse = True)
print(my_list)

my_list = [77,11,22,99,33,55]
my_list.reverse()
print(my_list)
```

```
[11, 22, 33, 55, 77, 99]
[99, 77, 55, 33, 22, 11]
[55, 33, 99, 22, 11, 77]
```

Ex3_8

Dacă avem lista `old_list = [[22, 44, 66], [11, 33, 55], [20, 30, 40]]`. Scrieți un cod care să creeze o nouă listă `new_list` care să cuprindă doar ultimul element din fiecare sub-listă, în cazul nostru `[66, 55, 40]`

```
old_list = [[22, 44, 66], [11, 33, 55], [20, 30, 40]]
new_list = []

for i in range(len(old_list)):
    new_list.append(old_list[i][-1])
```

```
print(new_list)
```

```
[66, 55, 40]
```

Ex3_9

Dacă avem `lista_mea = [1,2,3,4,5,6,7,8,9,10]`. Scrieți un cod care să aranjeze într-un mod aleatoriu elementele din listă. Studiu individual, ca sugestie trebuie să folosim (importăm) modulul `random` și să folosim una din comenzile `.randrange()` `.randint()` sau `.shuffle()`

```
import random
lista_mea = [1,2,3,4,5,6,7,8,9,10]
random.shuffle(lista_mea)
print(lista_mea)
```

```
[7, 5, 6, 9, 8, 4, 3, 10, 2, 1]
```

Ex3_10

Trebuie să creați o listă care să conțină numele celor mai buni 5 prieteni, după care să aranjăm aceste nume în ordine alfabetică.

```
lista = ["Dragos","Dragos3","Dragos2","Dragos5","Dragos4"]
lista.sort()
print(lista)
```

```
['Dragos', 'Dragos2', 'Dragos3', 'Dragos4', 'Dragos5']
```

Ex3_11

Trebuie să creați un dicționar care să cuprindă numele a trei prieteni (la cheie) și vârsta acestora (la valoare). La răspuns trebuie să obținem doar vârsta prietenului celui de al doilea prieten din dicționar.

```
dic = {"Dragos1" : 12, "Dragos2" : 13, "Dragos3" : 14}
print(dic["Dragos2"])
```

Ex3_12

Folosind dicționarul de la Ex3_11 trebuie să obținem la răspuns media vârstei celor trei prieteni.

```
dic = {"Dragos1" : 12, "Dragos2" : 13, "Dragos3" : 14}

medie = dic.values()
print(sum(medie) / len(medie))
```

Ex3_13

Folosind dicționarul de la Ex_12 trebuie să mai adăugați 2 prieteni (la cheie) și vârsta lor (la valoare) după care să determinați media vârstei celor 5 prieteni.

```
dic = {"Dragos1" : 12, "Dragos2" : 13, "Dragos3" : 14}

dic["Dragos4"] = 15

medie = dic.values()
print(sum(medie) / len(medie))
```

Ex3_14

Să presupunem că avem următorul dicționar. Continuați codul astfel încât la răspuns să obținem produsul celor trei valori ale cheilor din dicționar.

```
culori ={'Verde': 3,
         'Alb': 4,
         'Maro': 5}
```

```
import math
culori = {'Verde' : 3, 'Alb' : 4, 'Maro' : 5}
```

```
num = culori.values()
print(math.prod(num))
```

Ex3_15

Să presupunem că avem următoarea listă:

```
student_note = [
    {'student_nr': 1,'fizica': 5, 'mate': 7, 'chimia': 6},
    {'student_nr': 2,'fizica': 8, 'mate': 10, 'chimia': 6},
    {'student_nr': 3,'fizica': 6, 'mate': 7, 'chimia': 8}
    ]
```

După cum se observă lista noastră are 3 elemente care sunt dicționare. Trebuie să continuăm codul astfel ca la răspuns să obținem o nouă listă în care elementele sunt dicționare care ne arată media la cele trei discipline pentru fiecare student.

```
student_note = [
    {'student_nr' : 1, 'fizica' : 5, 'mate' : 7, 'chimia' : 6},
    {'student_nr' : 2, 'fizica' : 8, 'mate' : 10, 'chimia' : 6},
    {'student_nr' : 3, 'fizica' : 6, 'mate' : 7, 'chimia' : 8},
    ]
medie = []
for i in range(len(student_note)):
  medie.append((sum(student_note[i].values()) - i)/(len(student_note[i]) - 1))
print(medie)
```

```
[6.333333333333333, 8.333333333333334, 7.333333333333333]
```

Ex3_16

Să presupunem că avem următoarea listă:

```
student_note = [
    {'student_nr': 1,'fizica': 5, 'mate': 7},
    {'student_nr': 2,'fizica': 8, 'mate': 10},
    {'student_nr': 3,'fizica': 6, 'mate': 7,}
    ]
```

Trebuie să scriem un cod astfel încât în fiecare element din listă (fiecare dicționar) să mai apară un element (cheie: valoare) care să conțină media și valoarea numerică a acestei medii.

```
student_note = [
    {'student_nr' : 1, 'fizica' : 5, 'mate' : 7, 'chimia' : 6},
    {'student_nr' : 2, 'fizica' : 8, 'mate' : 10, 'chimia' : 6},
    {'student_nr' : 3, 'fizica' : 6, 'mate' : 7, 'chimia' : 8},
    ]
for i in range(len(student_note)):
  student_note[i]['informatica'] = 5

print(student_note)
```

```
[{'student_nr': 1, 'fizica': 5, 'mate': 7, 'chimia': 6, 'informatica': 5}, {'s
tudent_nr': 2, 'fizica': 8, 'mate': 10, 'chimia': 6, 'informatica': 5}, {'stud
ent_nr': 3, 'fizica': 6, 'mate': 7, 'chimia': 8, 'informatica': 5}]
```

Ex3_17

Să presupunem că avem următorul dicționar:

```
my_dict = {'C1': [2,4,6],
           'C2': [3,5,7]}
```

Trebuie să modificăm astfel acest dicționar astfel încât toate cifrele de la valorile cheilor să fie ridicate la pătrat.

```
my_dict = {'C1': [2,4,6], 'C2' : [3,5,7]}
```

```
keys = my_dict.keys()
for i in keys:
   for j in range(len(my_dict[i])):
      my_dict[i][j] = my_dict[i][j]**2
print(my_dict)
```

```
{'C1': [4, 16, 36], 'C2': [9, 25, 49]}
```

Ex3_18

Să presupunem că avem următorul dicționar:

```
salariu_ora = {'Alex': 27, 'Petru': 29, 'Ion': 30}
```

Trebuie să scriem codul astfel încât după rulare să obținem cheile și valorile din dicționar sub formă de listă și de asemenea să obținem și suma valorilor. Răspunsul trebuie să arate în felul următor:

```
['Alex', 'Petru', 'Ion']

[27, 29, 30]

86
```

```
salariu_ora = {'Alex' : 27, 'Petru' : 29, 'Ion' : 30}
print(list(salariu_ora.keys()))
print(list(salariu_ora.values()))
print(sum(salariu_ora.values()))
```

Ex3_19

Să presupunem că avem următorul dicționar:

```
marfa = {'flori': 50, 'fructe': 40, 'legume': 42}
```

Scrieți codul care la răspuns să ne dea numărul elementelor din dicționar și să aranjeze elementele cheilor în ordine crescătoare.

Răspunsul trebuie să arate în felul următor:

```
3

[40, 42, 50]
```

```
marfa = {'flori' : 50, 'fructe' : 40, 'legume' : 42}
print(len(list(marfa.keys())))
m = list(marfa.values())
m.sort()
print(m)
```

Ex3_20

Pentru dicționarul din exercițiul precedent scrieți un cod care la răspuns să ne dea doar elementele din dicționar care au valoarea mai mare de 40. Acest exercițiu se poate ușor rezolva cu ajutorul buclei `for` și instrucțiunii de comparare `if` pe care noi le vom studia în capitolul următor. Acest exercițiu îl puteți trece cu vederea dacă nu reușiți să îl rezolvați.

```
marfa = {'flori' : 50, 'fructe' : 40, 'legume' : 42}

keys = list(marfa.keys())
for i in keys:
   if marfa[i] > 40:
      print(marfa[i], end = " ")
```

Ex3_21

În acest exercițiu trebuie să vă dați seama ce răspuns vom obține după rularea codului următor?

```
x = 22
y = 14

print (x < y)
print (x > y)
print (x != y)
print (x == y)
```

```
False
True
True
False
```

Ex3_22

Corectați greșelile din cod astfel ca după rulare să obținem True.

```
d1 = {'a': 100. 'b': 200. 'c': 300}
d2 = {a: 300, b: 200, c: 100}

print(d1['b'] == d2['b'])

# Răspunsul va fi:
# True
```

```
d1 = {'a' : 100, 'b' : 200, 'c' : 300}
d2 = {'a' : 100, 'b' : 200, 'c' : 300}

print(d1['b'] == d2['b'])
```

Ex3_23

Pentru tupla `t = ('Iulian', 41)` greșelile scrieți un cod care după rulare să afișeze următorul răspuns: `Salut Iulian, ai împlinit 41 de ani!!!`

```
t = ('Iulian', 41)
print("Salut",t[0],', ai implinit',t[1],'de ani!!!')
```

Ex3_24

Creați o tuplă care să conțină numele a cinci prieteni. În continuare scrieți codul care după rulare să ne dea următoarele răspunsuri:

a) Să fie afișate toate elementele;

b) Să fie afișate doar primele 2 elemente din tuplă;

c) Să fie afișate doar primele 3 elemente din tuplă.

```
t = ('Dragos1', 'Dragos2','Dragos3','Dragos4','Dragos5')
print(t)
print(t[0:2])
print(t[0:3])
```

```
('Dragos1', 'Dragos2', 'Dragos3', 'Dragos4', 'Dragos5')
('Dragos1', 'Dragos2')
('Dragos1', 'Dragos2', 'Dragos3')
```

Ex3_25

Având un tip de date care conține numele, prenumele și vârsta, exemplu `t = ('Malcoci', 'Iulian', 41)` determinați nr. de elemente și tipul.

```
t = ('Dragos1', 'Dragos2','Dragos3','Dragos4','Dragos5')
print(len(t), ' ', type(t))
```

Ex3_26

Având următoarea tuplă `t = ('Mai', [1,9,20], 'Iunie', (7,11,23))` scrieți codul care după rulare să ne dea următorul răspuns:

```
20
Mai
11
[1, 9, 20]
(7, 11, 23)
```

```
t = ('Mai', [1,9,20], 'Iunie', (7,11,23))
print(t[1][2])
print(t[0])
print(t[3][1])
print(t[1])
print(t[3])
```

Ex3_27

În acest exercițiu avem 2 coduri dintre care unul după rulare va afișa o eroare, iar al doilea va rula. Trebuie să examinați ambele coduri și să înțelegeți de ce apare eroarea.

a)

```
t = (27, 7, 9, [11, 20])
t[2] = 10
print(t)
```

b)

```
t = (27, 7, 9, [11, 20])
t[3][0] = 10
print(t)
```

a) Nu se poate de atribuit la tuple
b) Se atribuie la lista

Ex3_28

Ca și în exemplul precedent avem 2 coduri care după rulare vor da eroare. Trebuie să examinați ambele variante și să înțelegeți de ce apar aceste erori.

a)

```
t = (27, 7, 9, [11, 20])
del t[1]
print(t)
```

b)

```
t = (27, 7, 9, [11, 20])
del t
print(t)
```

a) Nu se poate de modificat tuple
b) Variablia t nu mai exista

Ex3_29

Având următoarea tuplă `discipline = ('fizica', 'chimia', 'geografia')` trebuie să verificăm dacă elementele `'fizica'` și `'Fizica'` se află în tupla `discipline`.

```
discipline = ("fizica", "chimia", "geografia")
print('fizica' in discipline)
print('Fizica' in discipline)
```

Ex3_30

Având următoarea listă `lista_1 = [3, '4', 7, 4, 3, '4', 11]` scrieți un cod care după rulare să creeze o listă `lista_2` din care să fie omise elementele duplicate.

```
lista_1 = [3,'4',7,4,3,'4',11]
lista_2 = list(set(lista_1))
print(lista_2)
```

```
[3, 4, '4', 7, 11]
```

Ex3_31

Având următorul set `setul_meu = {'ion', 'ana', 'ion', 'vera'}` scrieți un cod care să adauge elementul `'vlad'` în set, după care să transformați setul în listă, iar după rulare să apară lista.

```
setul_meu = {'ion','ana','ion','vera'}
setul_meu.add('vlad')
setul_meu = list(setul_meu)
print(setul_meu)
```

Ex3_32

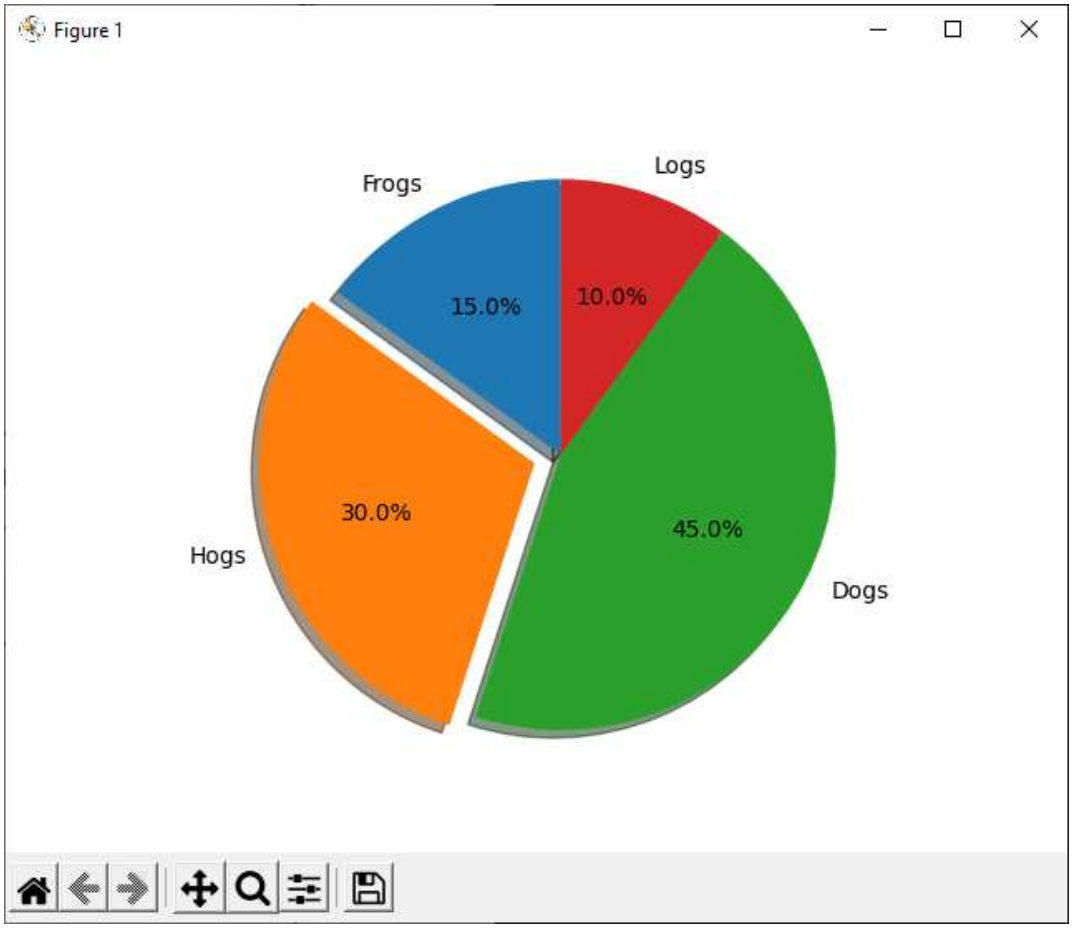
În acest exercițiu sunt prezentate instrucțiunile condiționate if și elif. Trebuie să copiați codul să îl rulați și să încercați să înțelegeți de ce am obținut unul din acele două răspunsuri.

```
set_1 = {'mere', 'pere'}
if('banane' in set_1):
    print('DA, bananele sunt în lista de cumpărături!!!')
elif('banane' not in set_1):
    print('NU, bananele nu sunt în lista de cumpărături!!!')
```

Se duce la elif din cauza ca 'banane' nu sunt in setul set_1

Problema 1

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

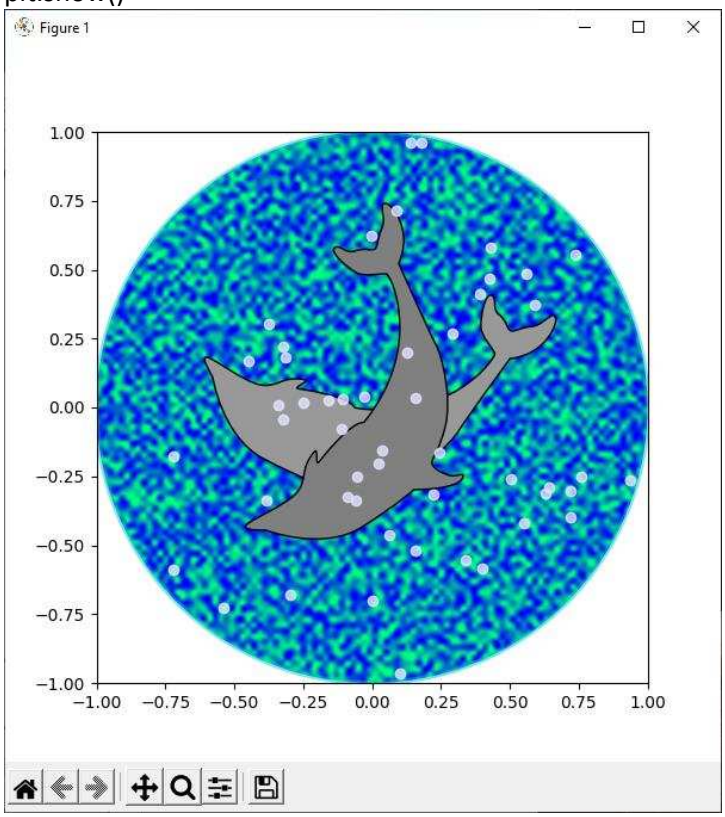


Problema 2

```
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from matplotlib.patches import Circle, PathPatch
from matplotlib.path import Path
from matplotlib.transforms import Affine2D
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)
```

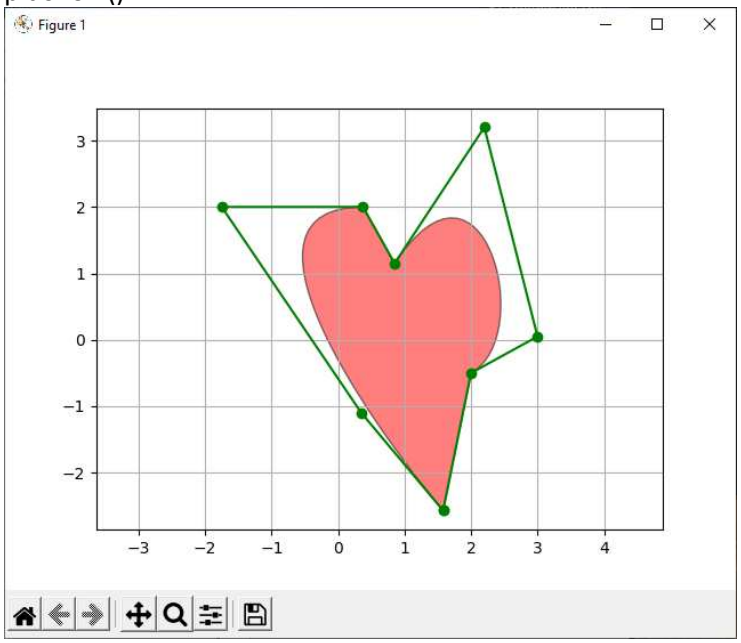
```
r = np.random.rand(50)
t = np.random.rand(50) * np.pi * 2.0
x = r * np.cos(t)
y = r * np.sin(t)

fig, ax = plt.subplots(figsize=(6, 6))
circle = Circle((0, 0), 1, facecolor='none',
        edgecolor=(0, 0.8, 0.8), linewidth=3, alpha=0.5)
ax.add_patch(circle)

im = plt.imshow(np.random.random((100, 100)),
        origin='lower', cmap=cm.winter,
        interpolation='spline36',
        extent=([-1, 1, -1, 1]))
im.set_clip_path(circle)

plt.plot(x, y, 'o', color=(0.9, 0.9, 1.0), alpha=0.8)

# Dolphin from OpenClipart library by Andy Fitzsimon
#   <cc:License rdf:about="http://web.resource.org/cc/PublicDomain">
#     <cc:permits rdf:resource="http://web.resource.org/cc/Reproduction"/>
#     <cc:permits rdf:resource="http://web.resource.org/cc/Distribution"/>
#     <cc:permits rdf:resource="http://web.resource.org/cc/DerivativeWorks"/>
#   </cc:License>

dolphin = """
M -0.59739425,160.18173 C -0.62740401,160.18885 -0.57867129,160.11183
-0.57867129,160.11183 C -0.57867129,160.11183 -0.5438361,159.89315
-0.39514638,159.81496 C -0.24645668,159.73678 -0.18316813,159.71981
-0.18316813,159.71981 C -0.18316813,159.71981 -0.10322971,159.58124
-0.057804323,159.58725 C -0.029723983,159.58913 -0.061841603,159.60356
-0.071265813,159.62815 C -0.080250183,159.65325 -0.082918513,159.70554
-0.061841203,159.71248 C -0.040763903,159.7194 -0.0066711426,159.71091
0.077336307,159.73612 C 0.16879567,159.76377 0.28380306,159.86448
0.31516668,159.91533 C 0.3465303,159.96618 0.5011127,160.1771
0.5011127,160.1771 C 0.63668998,160.19238 0.67763022,160.31259
0.66556395,160.32668 C 0.65339985,160.34212 0.66350443,160.33642
0.64907098,160.33088 C 0.63463742,160.32533 0.61309688,160.297
0.5789627,160.29339 C 0.54348657,160.28968 0.52329693,160.27674
0.50728856,160.27737 C 0.49060916,160.27795 0.48965803,160.31565
0.46114204,160.33673 C 0.43329696,160.35786 0.4570711,160.39871
0.43309565,160.40685 C 0.4105108,160.41442 0.39416631,160.33027
0.3954995,160.2935 C 0.39683269,160.25672 0.43807996,160.21522
0.44567915,160.19734 C 0.45327833,160.17946 0.27946869,159.9424
-0.061852613,159.99845 C -0.083965233,160.0427 -0.26176109,160.06683
-0.26176109,160.06683 C -0.30127962,160.07028 -0.21167141,160.09731
-0.24649368,160.1011 C -0.32642366,160.11569 -0.34521187,160.06895
-0.40622293,160.0819 C -0.467234,160.09485 -0.56738444,160.17461
-0.59739425,160.18173
"""

vertices = []
codes = []
```

```
parts = dolphin.split()
i = 0
code_map = {
   'M': Path.MOVETO,
   'C': Path.CURVE4,
   'L': Path.LINETO,
}

while i < len(parts):
   path_code = code_map[parts[i]]
   npoints = Path.NUM_VERTICES_FOR_CODE[path_code]
   codes.extend([path_code] * npoints)
   vertices.extend([[*map(float, y.split(','))]
                    for y in parts[i + 1:][:npoints]])
   i += npoints + 1
vertices = np.array(vertices)
vertices[:, 1] -= 160

dolphin_path = Path(vertices, codes)
dolphin_patch = PathPatch(dolphin_path, facecolor=(0.6, 0.6, 0.6),
                          edgecolor=(0.0, 0.0, 0.0))
ax.add_patch(dolphin_patch)

vertices = Affine2D().rotate_deg(60).transform(vertices)
dolphin_path2 = Path(vertices, codes)
dolphin_patch2 = PathPatch(dolphin_path2, facecolor=(0.5, 0.5, 0.5),
                           edgecolor=(0.0, 0.0, 0.0))
ax.add_patch(dolphin_patch2)

plt.show()
```



Problema 3

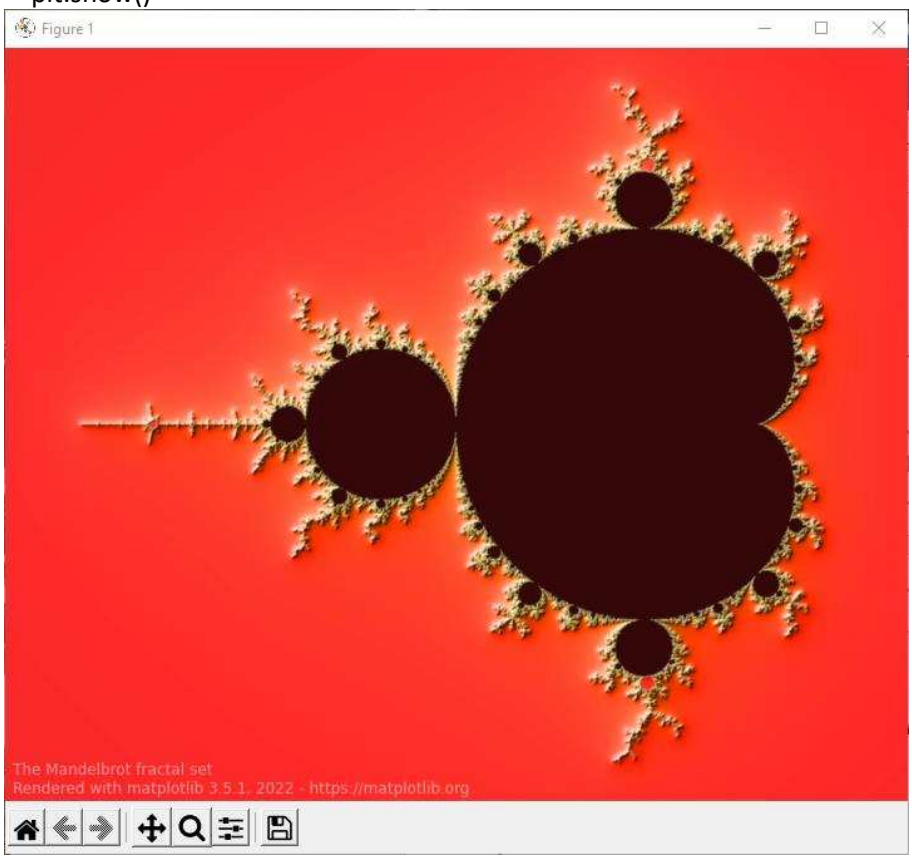
```
import matplotlib.path as mpath
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

Path = mpath.Path
path_data = [
   (Path.MOVETO, (1.58, -2.57)),
   (Path.CURVE4, (0.35, -1.1)),
   (Path.CURVE4, (-1.75, 2.0)),
   (Path.CURVE4, (0.375, 2.0)),
   (Path.LINETO, (0.85, 1.15)),
   (Path.CURVE4, (2.2, 3.2)),
   (Path.CURVE4, (3, 0.05)),
   (Path.CURVE4, (2.0, -0.5)),
   (Path.CLOSEPOLY, (1.58, -2.57)),
   ]
codes, verts = zip(*path_data)
path = mpath.Path(verts, codes)
patch = mpatches.PathPatch(path, facecolor='r', alpha=0.5)
ax.add_patch(patch)

# plot control points and connecting lines
x, y = zip(*path.vertices)
line, = ax.plot(x, y, 'go-')

ax.grid()
ax.axis('equal')
plt.show()
```



Problema 4

```
import numpy as np


def mandelbrot_set(xmin, xmax, ymin, ymax, xn, yn, maxiter, horizon=2.0):
    X = np.linspace(xmin, xmax, xn).astype(np.float32)
    Y = np.linspace(ymin, ymax, yn).astype(np.float32)
    C = X + Y[:, None] * 1j
    N = np.zeros_like(C, dtype=int)
    Z = np.zeros_like(C)
    for n in range(maxiter):
        I = abs(Z) < horizon
        N[I] = n
        Z[I] = Z[I]**2 + C[I]
    N[N == maxiter-1] = 0
    return Z, N


if __name__ == '__main__':
    import time
    import matplotlib
    from matplotlib import colors
    import matplotlib.pyplot as plt

    xmin, xmax, xn = -2.25, +0.75, 3000 // 2
    ymin, ymax, yn = -1.25, +1.25, 2500 // 2
    maxiter = 200
    horizon = 2.0 ** 40
    log_horizon = np.log2(np.log(horizon))
    Z, N = mandelbrot_set(xmin, xmax, ymin, ymax, xn, yn, maxiter, horizon)

    with np.errstate(invalid='ignore'):
        M = np.nan_to_num(N + 1 - np.log2(np.log(abs(Z))) + log_horizon)

    dpi = 72
    width = 10
    height = 10*yn/xn
    fig = plt.figure(figsize=(width, height), dpi=dpi)
    ax = fig.add_axes([0, 0, 1, 1], frameon=False, aspect=1)

    # Shaded rendering
    light = colors.LightSource(azdeg=315, altdeg=10)
    M = light.shade(M, cmap=plt.cm.hot, vert_exag=1.5,
                    norm=colors.PowerNorm(0.3), blend_mode='hsv')
    ax.imshow(M, extent=[xmin, xmax, ymin, ymax], interpolation="bicubic")
    ax.set_xticks([])
    ax.set_yticks([])

    # Some advertisement for matplotlib
    year = time.strftime("%Y")
    text = ("The Mandelbrot fractal set\n"
            "Rendered with matplotlib %s, %s - https://matplotlib.org"
            % (matplotlib.__version__, year))
```

```
ax.text(xmin+.025, ymin+.025, text, color="white", fontsize=12, alpha=0.5)

plt.show()
```



Problema 5

```
import re
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.path import Path
import matplotlib.patches as patches

firefox = "M28.4,22.469c0.479-0.964,0.851-1.991,1.095-3.066c0.953-3.661,0.666-6.854,0.666-6.854l-
0.327,2.104c0,0-0.469-3.896-1.044-5.353c-0.881-2.231-1.273-2.214-1.274-
2.21c0.542,1.379,0.494,2.169,0.483,2.288c-0.01-0.016-0.019-0.032-0.027-0.047c-0.131-0.324-0.797-1.819-
2.225-2.878c-2.502-2.481-5.943-4.014-9.745-4.015c-4.056,0-7.705,1.745-
10.238,4.525C5.444,6.5,5.183,5.938,5.159,5.317c0,0-0.002,0.002-0.006,0.005c0-0.011-0.003-0.021-0.003-
0.031c0,0-1.61,1.247-1.436,4.612c-0.299,0.574-0.56,1.172-0.777,1.791c-0.375,0.817-0.75,2.004-
1.059,3.746c0,0,0.133-0.422,0.399-0.988c-0.064,0.482-0.103,0.971-0.116,1.467c-0.09,0.845-0.118,1.865-
0.039,3.088c0,0,0.032-0.406,0.136-1.021c0.834,6.854,6.667,12.165,13.743,12.165l0,0c1.86,0,3.636-
0.37,5.256-1.036C24.938,27.771,27.116,25.196,28.4,22.469zM16.002,3.356c2.446,0,4.73,0.68,6.68,1.86c-
2.274-0.528-3.433-0.261-3.423-0.248c0.013,0.015,3.384,0.589,3.981,1.411c0,0-1.431,0-2.856,0.41c-
0.065,0.019,5.242,0.663,6.327,5.966c0,0-0.582-1.213-1.301-1.42c0.473,1.439,0.351,4.17-0.1,5.528c-
0.058,0.174-0.118-0.755-1.004-1.155c0.284,2.037-0.018,5.268-1.432,6.158c-0.109,0.07,0.887-3.189,0.201-
1.93c-4.093,6.276-8.959,2.539-10.934,1.208c1.585,0.388,3.267,0.108,4.242-0.559c0.982-0.672,1.564-
1.162,2.087-1.047c0.522,0.117,0.87-0.407,0.464-0.872c-0.405-0.466-1.392-1.105-2.725-0.757c-0.94,0.247-
2.107,1.287-3.886,0.233c-1.518-0.899-1.507-1.63-1.507-2.095c0-0.366,0.257-0.88,0.734-
1.028c0.58,0.062,1.044,0.214,1.537,0.466c0.005-0.135,0.006-0.315-0.001-0.519c0.039-0.077,0.015-0.311-
0.047-0.596c-0.036-0.287-0.097-0.582-0.19-0.851c0.01-0.002,0.017-0.007,0.021-0.021c0.076-0.344,2.147-
1.544,2.299-1.659c0.153-0.114,0.55-0.378,0.506-1.183c-0.015-0.265-0.058-0.294-2.232-0.286c-
```

```
0.917,0.003-1.425-0.894-1.589-1.245c0.222-1.231,0.863-2.11,1.919-2.704c0.02-0.011,0.015-0.021-0.008-
0.027c0.219-0.127-2.524-0.006-3.76,1.604C9.674,8.045,9.219,7.95,8.71,7.95c-0.638,0-1.139,0.07-
1.603,0.187c-0.05,0.013-0.122,0.011-0.208-0.001C6.769,8.04,6.575,7.88,6.365,7.672c0.161-0.18,0.324-
0.356,0.495-0.526C9.201,4.804,12.43,3.357,16.002,3.356z"  # noqa


def svg_parse(path):
    commands = {'M': (Path.MOVETO,),
                'L': (Path.LINETO,),
                'Q': (Path.CURVE3,)*2,
                'C': (Path.CURVE4,)*3,
                'Z': (Path.CLOSEPOLY,)}
    vertices = []
    codes = []
    cmd_values = re.split("([A-Za-z])", path)[1:]  # Split over commands.
    for cmd, values in zip(cmd_values[::2], cmd_values[1::2]):
        # Numbers are separated either by commas, or by +/- signs (but not at
        # the beginning of the string).
        points = ([*map(float, re.split(",|(?<!^)(?=[+-])", values))] if values
                  else [(0., 0.)])  # Only for "z/Z" (CLOSEPOLY).
        points = np.reshape(points, (-1, 2))
        if cmd.islower():
            points += vertices[-1][-1]
        codes.extend(commands[cmd.upper()])
        vertices.append(points)
    return np.array(codes), np.concatenate(vertices)


# SVG to Matplotlib
codes, verts = svg_parse(firefox)
path = Path(verts, codes)

xmin, ymin = verts.min(axis=0) - 1
xmax, ymax = verts.max(axis=0) + 1

fig = plt.figure(figsize=(5, 5), facecolor="0.75")  # gray background
ax = fig.add_axes([0, 0, 1, 1], frameon=False, aspect=1,
                  xlim=(xmin, xmax),  # centering
                  ylim=(ymax, ymin),  # centering, upside down
                  xticks=[], yticks=[])  # no ticks

# White outline (width = 6)
ax.add_patch(patches.PathPatch(path, facecolor='none', edgecolor='w', lw=6))
# Actual shape with black outline
ax.add_patch(patches.PathPatch(path, facecolor='orange', edgecolor='k', lw=2))

plt.show()  # Display
```

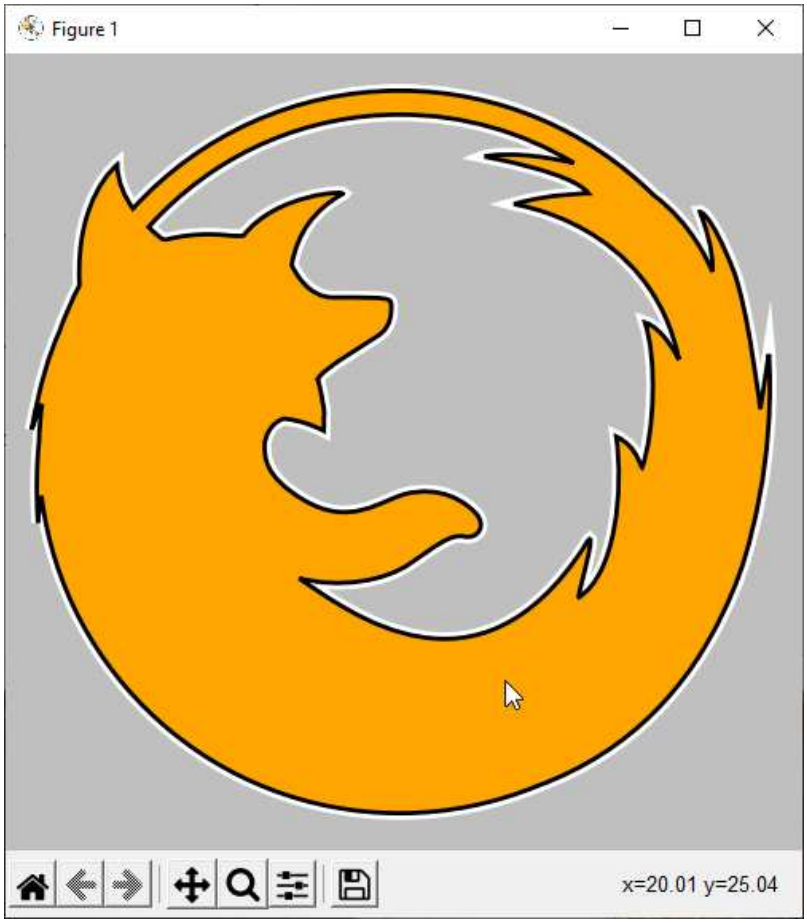


Figure 1
x=20.01 y=25.04

Noutati Python 3.10

Python 3.10 include multe îmbunătățiri ale verificării tipurilor, inclusiv operatorul de unire a tipurilor, care are o sintaxă mai curată.

```
# Function that accepts either `int` or `float`
# Old:
def func(value: Union[int, float]) -> Union[int, float]:
    return value

# New:
def func(value: int | float) -> int | float:
    return value
```

De asemenea, această îmbunătățire simplă nu se limitează la adnotări de tip, ea poate fi aplicată cu funcțiile isinstance() și issubclass():

```
isinstance("hello", int | str)
# True
```

Introduceți modificări de sintaxă a aliasului:

Era:

```
FileName = str

def parse(file: FileName) -> None:
    ...
```

A devenit:

```
FileName: TypeAlias = str

def parse(file: FileName) -> None:
    ...
```

Începând cu Python 3.10, puteți apela int.bit_count() pentru a număra numărul de biți din reprezentarea binară a unui număr întreg.

```
value = 42
print(bin(value))
# '0b101010'
print(value.bit_count())
# 3
```

Managerii de context sunt grozavi pentru deschiderea și închiderea fișierelor, lucrul cu conexiuni la baze de date și multe altele, iar în Python 3.10 devin puțin mai convenabile. Modificarea vă permite să specificați mai mulți manageri de context în paranteze, ceea ce este util dacă doriți să creați mai mulți manageri într-o singură instrucțiune:

```
with (
    open("somefile.txt") as some_file,
    open("otherfile.txt") as other_file,
):
    ...

from contextlib import redirect_stdout

with (open("somefile.txt", "w") as some_file,
      redirect_stdout(some_file)):
    ...
```

Ca și în cazul tuturor versiunilor recente de Python, îmbunătățirile de performanță vor veni cu Python 3.10. Prima este optimizarea constructorilor str(), bytes() și bytearray(), care ar trebui să fie cu aproximativ 30% mai rapid

```
~ $ ./python3.10 -m pyperf timeit -q --compare-to=python "str()"
Mean +- std dev: [python] 81.9 ns +- 4.5 ns -> [python3.10] 60.0 ns +- 1.9 ns:
1.36x faster (-27%)
~ $ ./python3.10 -m pyperf timeit -q --compare-to=python "bytes()"
Mean +- std dev: [python] 85.1 ns +- 2.2 ns -> [python3.10] 60.2 ns +- 2.3 ns:
1.41x faster (-29%)
~ $ ./python3.10 -m pyperf timeit -q --compare-to=python "bytearray()"
Mean +- std dev: [python] 93.5 ns +- 2.1 ns -> [python3.10] 73.1 ns +- 1.8 ns:
1.28x faster (-22%)
```

A fost adaugat match:

```
def func(day):
    match day:
        case "Monday":
            return "Here we go again..."
        case "Friday":
            return "Happy Friday!"
        case "Saturday" | "Sunday":  # Multiple literals can be combined with `|`
            return "Yay, weekend!"
        case _:
            return "Just another day..."
```

In Python 3.10 sa facut un lucru enorm asupra afisarea erorilor pentru a usura mai tare viata programistilor:

Exemple:

```
expected = {9: 1, 18: 2, 19: 2, 27: 3, 28: 3, 29: 3, 36: 4, 37: 4,
            38: 4, 39: 4, 45: 5, 46: 5, 47: 5, 48: 5, 49: 5, 54: 6,
some_other_code = foo()
```

Versiunea trecuta:

```
File "example.py", line 3
    some_other_code = foo()
                    ^
SyntaxError: invalid syntax
```

Python 3.10.

```
File "example.py", line 1
    expected = {9: 1, 18: 2, 19: 2, 27: 3, 28: 3, 29: 3, 36: 4, 37: 4,
               ^
SyntaxError: '{' was never closed
```

Versiunea trecuta:

```
>>> foo(x, z for z in range(10), t, w)
  File "<stdin>", line 1
    foo(x, z for z in range(10), t, w)
           ^
SyntaxError: Generator expression must be parenthesized
```

Python 3.10.

```
>>> foo(x, z for z in range(10), t, w)
  File "<stdin>", line 1
    foo(x, z for z in range(10), t, w)
           ^^^^^^^^^^^^^^^^^^^^
SyntaxError: Generator expression must be parenthesized
```

Versiunea trecuta:

```
>>> if rocket.position > event_horizon
  File "<stdin>", line 1
    if rocket.position > event_horizon
                                      ^
SyntaxError: expected ':'
```

Python 3.10.

```
>>> {x,y for x,y in zip('abcd', '1234')}
  File "<stdin>", line 1
    {x,y for x,y in zip('abcd', '1234')}
     ^
SyntaxError: did you forget parentheses around the comprehension target?
```