



#### Tema 4 Multimi Rough

### Sisteme Semantice

Draghici Andreea-Maria  
Inginerie Software  
IS 2.1

---

## Tema 4 Multimi Rough

---

### Contents

1. Tabel de informații .....	3
1.1 Proiectarea unui Dialog cu Utilizatorul pentru a Selecta Entitățile din Tabel Conform Cerințelor Utilizatorului ( Mulțimea X) .....	3
1.2 Proiectarea Aproximărilor Mulțimii X Folosind Atributele din Tabel .....	4
2.1 Demo aplicativ – solution.py .....	5
Dialogul cu utilizatorul ( se citește de la tastatură ): .....	5
Rezultatele filtrării .....	6
Aplicarea Teoriei Mulțimilor Rough .....	7
2.2 Demo aplicativ – solution2.py .....	8
1. Tabelul inițial de informații .....	10
2. Mulțimea X .....	10
3. Aproximarea Inferioară .....	11
4. Aproximarea Superioară .....	11
5. Regiunea de Frontieră .....	11
Observație .....	11
2.3 Demo aplicativ – solution3.py .....	12
1. Crearea datelor și tabelului .....	14
2. Calcularea deciziei .....	14
3. Gruparea în clase de echivalență .....	14
4. Aproximări și regiunea de frontieră .....	14
5. Generarea regulilor .....	14
6. Reducte și nucleu .....	15
7. Regiunea pozitivă .....	15
8. Funcțiile de apartenență rough .....	15
Tabelul Inițial de Informații .....	18
Clasele de Echivalență .....	18
Aproximări și Regiunea de Frontieră .....	18
Reguli Certe .....	18
Reguli Posibile .....	18
Nucleul .....	18
Regiunea Pozitivă .....	19

## Tema 4 Multimi Rough

---

Funcții de Apartenență Rough .....	19
Concluzie Generală .....	19
3. Concluzii finale .....	19
4. Cod sursă .....	20

## Tema 4 Multimi Rough

### 1. Tabel de informații

ID	Produs	Categorie	Pret (\$)	Stoc	Vandut online	Rating mediu	An lansare
1	Telefon mobil Samsung	Smartphone	300	DA	DA	4.7	2021
2	Tabletă Samsung	Tabletă	450	NU	DA	4.0	2020
3	Laptop Asus	Laptop	800	DA	NU	4.8	2022
4	Smartwatch	Ceas	200	DA	DA	4.0	2023
5	Consolă	Gaming	350	NU	NU	4.5	2023
6	Laptop Dell	Laptop	750	DA	DA	4.3	2022
7	Telefon iPhone	Smartphone	350	DA	DA	5.0	2024
9	Camera Canon	Aparat Compact	400	DA	DA	4.3	2021

### Detalii ale Setului de Date:

#### 1.1 Proiectarea unui Dialog cu Utilizatorul pentru a Selecta Entitățile din Tabel Conform Cerințelor Utilizatorului ( Mulțimea X)

Codul inițiază un dialog cu utilizatorul prin solicitarea inputurilor pentru diferite criterii:

- Categorie (ex. Laptop, Smartphone)
- Prețul maxim dorit
- Necesitatea ca produsul să fie în stoc
- Ratingul minim dorit
- Dacă produsul trebuie să fie vândut online
- Anul minim de lansare

```
criterii = {  
    'categorie' : input("Introduceți categoria produsului (ex. Laptop, Smartphone): "),  
    'pret_max' : int(input("Introduceți prețul maxim dorit: ")),  
    'stoc' : input("Produsul trebuie să fie în stoc? (DA/NU): "),  
    'rating_min' : float(input("Introduceți ratingul minim dorit: ")),  
    'vandut_online' : input("Produsul trebuie să fie vândut online? (DA/NU): "),  
    'an_min' : int(input("Introduceți anul minim de lansare: "))  
}
```

Aceste inputuri sunt folosite pentru a stabili setul de criterii care definește mulțimea X. Astfel, mulțimea X este definită ca fiind ansamblul tuturor produselor din tabel care îndeplinesc criteriile specificate de utilizator.

### 1.2 Proiectarea Aproximărilor Mulțimii X Folosind Atributele din Tabel

Voi defini multimea X ca fiind setul de produse care îndeplinesc anumite criterii specificate de utilizator (e.g., categoria, preț maxim, disponibilitate în stoc, rating minim).

Atributul U reprezintă universul de produse, iar A setul de atribute pentru fiecare produs, cum ar fi preț, stoc, rating, etc.

#### Aproximarea Inferioară $\text{aprox\_inferioara}(X)$

Aproximarea inferioară a mulțimii X este definită ca mulțimea tuturor produselor care îndeplinesc **toate** criteriile specificate:

$$\text{aprox\_inferioara}(X) = \{ x \in U \mid \forall a \in A, \text{ condițiile pe } a \text{ sunt îndeplinite pentru } a \text{ fi în } x \}$$

Aceasta include produsele pentru care avem certitudinea că respectă toate condițiile.

#### Aproximarea Superioară $\text{aprox\_superioara}(X)$

Aproximarea superioară a mulțimii X include produsele care îndeplinesc **cel puțin una** dintre condițiile specificate:

$$\text{aprox\_superioara}(X) = \{ x \in U \mid \exists a \in A, \text{ cel puțin un criteriu pe } a \text{ este îndeplinit pentru } a \text{ fi în } x \}$$

Aceasta reprezintă posibilitatea ca produsele să fie acceptabile conform criteriilor, dar cu o certitudine mai mică.

#### Regiunea de Frontieră BND (X)

Regiunea de frontieră include produsele pentru care există incertitudine între a fi sau nu în X:

$$\text{BND}(X) = \text{aprox\_superioara}(X) - \text{aprox\_inferioara}(X)$$

Aceste produse nu pot fi clasificate cu certitudine folosind doar informațiile disponibile.

#### Exemplu de Aplicare

Dacă utilizatorul caută un "Laptop" cu prețul maxim de 800\$, care este în stoc și are un rating de minim 4.5, setările pentru X ar putea fi:

- Categorie = Laptop
- Preț (\$)  $\leq 800$
- Stoc = DA
- Rating mediu  $\geq 4.5$

## Tema 4 Multimi Rough

În acest caz, utilizând setul de date:

- aprox\_inferioara (X) va conține "Laptop Asus" și "Laptop Dell" dacă ambele îndeplinesc toate condițiile.
- aprox\_superioara (X) va include orice laptop cu oricare dintre aceste condiții îndeplinite parțial sau complet.
- BND (X) va cuprinde laptopurile pentru care informațiile nu sunt suficiente pentru a determina dacă îndeplinesc absolut toate criteriile.

### 2.1 Demo aplicativ – solution.py

Codul inițializează un DataFrame pandas cu un set de date și folosește criterii specificate de utilizator pentru a calcula trei tipuri de rezultate:

1. **Aproximarea inferioară** - care reprezintă produsele ce îndeplinesc toate criteriile specificate. Aceasta este calculată folosind un operator logic "și" între toate condițiile (folosind operatorul & în Python).
2. **Aproximarea superioară** - care include produse ce îndeplinesc oricare dintre condițiile specificate. Acesta este calculat folosind un operator logic "sau" între condiții (folosind operatorul | în Python).
3. **Regiunea de frontieră** - care reprezintă diferența dintre aproximarea superioară și cea inferioară, indicând elementele pentru care există incertitudine în clasificare. Aceasta este calculată folosind diferența între indicii produselor din cele două mulțimi.

```
def filtreaza_produce(df, criterii):  
    """Filtreaza produsele folosind criteriile specificate."""  
    conditii = {  
        'Categorie' : df['Categorie'] == criterii['categorie'],  
        'Pret ($)' : df['Pret ($)'] <= criterii['pret_max'],  
        'Stoc' : df['Stoc'] == criterii['stoc'],  
        'Rating mediu' : df['Rating mediu'] >= criterii['rating_min'],  
        'Vândut online' : df['Vândut online'] == criterii['vândut_online'],  
        'An lansare' : df['An lansare'] >= criterii['an_min']  
    }  
  
    # C filtru pentru aproximarea inferioară si superioara  
    filtru_inferior = pd.Series([True] * len(df))  
    filtru_superior = pd.Series([False] * len(df))  
  
    for key, value in conditii.items():  
        filtru_inferior &= value  
        filtru_superior |= value  
  
    aproximare_inferioara = df[filtru_inferior]  
    aproximare_superioara = df[filtru_superior]  
    regiune_frontiera = aproximare_superioara[~aproximare_superioara.index.isin(aproximare_inferioara.index)]  
  
    return aproximare_inferioara, aproximare_superioara, regiune_frontiera
```

### Dialogul cu utilizatorul ( se citește de la tastatură ):

1. **Categorie:** Laptop - Utilizatorul este interesat de produse din categoria "Laptop".
2. **Preț maxim dorit:** 800 - Produsele trebuie să aibă un preț de până la 800 de dolari.

## Tema 4 Multimi Rough

3. **Stoc:** DA - Produsele trebuie să fie în stoc.
4. **Rating minim dorit:** 4.3 - Ratingul mediu al produselor trebuie să fie cel puțin 4.3.
5. **Vândut online:** DA - Produsele trebuie să fie disponibile pentru vânzare online.
6. **An minim de lansare:** 2020 - Produsele trebuie să fie lansate începând cu anul 2020

### Rezultatele filtrării

- **Aproximare inferioară:** Contine produsele care îndeplinesc toate criteriile specificate, fără ambiguitate. În cazul meu, doar "Laptop Dell" se potrivește perfect tuturor criteriilor.
- **Aproximare superioară:** Include produsele care îndeplinesc cel puțin unul dintre criteriile specificate, indicând o incertitudine mai mare. Practic, fiecare produs din lista inițială apare aici, deoarece toate îndeplinesc cel puțin un criteriu (de exemplu, fiecare are un rating peste 4.0, sunt vândute online, etc.).
- **Regiune de frontieră:** Conține produsele pentru care există incertitudine între a fi sau nu complet adecvate conform tuturor criteriilor. În acest caz, regiunea de frontieră include toate produsele care apar în aproximarea superioară, dar nu sunt prezente în cea inferioară. Practic, orice produs care nu îndeplinește toate criteriile, dar îndeplinește unele, se află în această categorie.

```
D:\Data\_DevTools\Conda\envs\Tema4\python.exe D:\Data\Work\Tema4\src\rezolvare_execitiu1.py
Introduceți categoria produsului (ex. Laptop, Smartphone): laptop
Introduceți prețul maxim dorit: 800
Produsul trebuie să fie în stoc? (DA/NU): DA
Introduceți ratingul minim dorit: 4.3
Produsul trebuie să fie vândut online? (DA/NU): DA
Introduceți anul minim de lansare: 2020

Aproximare inferioară:
  ID      Produs Categoria  ...  Vândut online Rating mediu An lansare
5   6   Laptop Dell      Laptop  ...           DA           4.3      2022

[1 rows x 8 columns]

Aproximare superioară:
  ID      Produs  ... Rating mediu An lansare
0   1  Telefon mobil Samsung  ...      4.7      2021
1   2    Tabletă Samsung  ...      4.0      2020
2   3      Laptop Asus  ...      4.8      2022
3   4      Smartwatch  ...      4.0      2023
4   5    Consolă Gaming  ...      4.5      2023
5   6      Laptop Dell  ...      4.3      2022
6   7    Telefon iPhone  ...      5.0      2024
7   9      Camera Canon  ...      4.3      2021

[8 rows x 8 columns]
```

## Tema 4 Multimi Rough

```
Regiune de frontieră:
  ID      Produs  ... Rating mediu  An lansare
0  1  Telefon mobil Samsung  ...      4.7      2021
1  2      Tabletă Samsung  ...      4.0      2020
2  3      Laptop Asus    ...      4.8      2022
3  4      Smartwatch    ...      4.0      2023
4  5      Consolă Gaming  ...      4.5      2023
6  7      Telefon iPhone  ...      5.0      2024
7  9      Camera Canon   ...      4.3      2021

[7 rows x 8 columns]

Process finished with exit code 0
```

### Aplicarea Teoriei Multimilor Rough

Teoria multimilor Rough în acest context ajută la gestionarea incertitudinilor în setul de date:

- **Aproximarea inferioară** ne dă certitudinea că produsele listate aici îndeplinesc toate condițiile impuse.
- **Aproximarea superioară** ne arată posibilitatea ca produsele să fie suficient de bune, chiar dacă nu îndeplinesc fiecare condiție complet.
- **Regiunea de frontieră** este utilă pentru a identifica produsele despre care nu avem suficiente informații pentru a decide dacă îndeplinesc complet toate cerințele.



### 2.2 Demo aplicativ – solution2.py

```
4 def creeaza_tabel() :
5     # Crearea tabelului de informatii
6     data = {
7         "Nume" : ["Marius", "Petre", "Ana", "Maria", "Nicu"],
8         "Varsta" : [33, 27, 20, 39, 45],
9         "Studii" : ["Postliceale", "Superioare", "Superioare", "Postliceale", "Liceale"],
10        "Vechime" : [8, 5, 0, 14, 19],
11        "Certificat_formator" : ["DA", "NU", "DA", "NU", "NU"]
12    }
13    return pd.DataFrame(data)
14
15
16 def indeplineste_conditii(row) :
17     # Condițiile de angajare
18     return row["Varsta"] < 40 and row["Studii"] in ["Superioare", "Postliceale"] and row["Vechime"] > 0
19
20
21 def aproximare_inferioara(row) :
22     # Apartine sigur dacă îndeplinește condițiile și are Certificat formator = "DA"
23     return row["Conditii"] and row["Certificat_formator"] == "DA"
24
25
26 def aproximare_superioara(row) :
27     # Poate apartine dacă îndeplinește condițiile
28     return row["Conditii"]
29
```

#### Funcția creeaza\_tabel:

- Creează un tabel de informații pentru persoanele analizate.
- Include caracteristici precum vârstă, studii, vechime, și dacă au certificat de formator.

#### Funcția calculeaza\_decizie:

- Calculează dacă o persoană îndeplinește condițiile de angajare.

#### Funcția indiscernabilitate:

- Grupați obiectele în clase de echivalență bazate pe anumite atribute.

#### Funcția aproximare\_inferioara\_superioara:

- Calculează aproximarea inferioară (sigură), superioară (posibilă), și regiunea de frontieră.

#### Funcția genereaza\_reguli:

## Tema 4 Multimi Rough

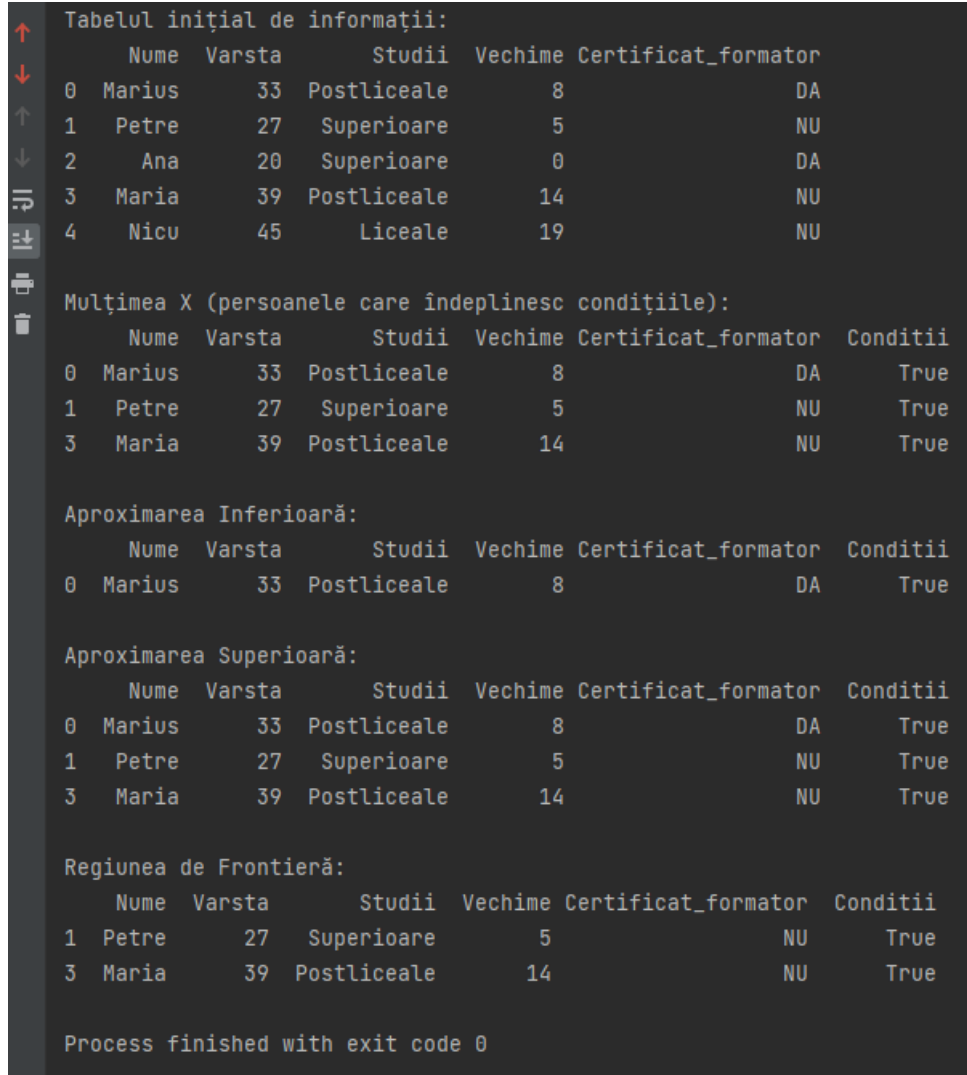
- Generează reguli certe și posibile pe baza claselor de echivalență și deciziilor atribuite.

```
1 def main() :
2     # Crearea tabelului
3     df = creeaza_tabel()
4     print("Tabelul initial de informatii:")
5     print(df)
6
7     # Adăugarea unei coloane pentru conditii
8     df["Conditii"] = df.apply(indeplineste_conditii, axis=1)
9
10    # Definirea multimii X
11    X = df[df["Conditii"]]
12
13    # Calcularea aproximărilor
14    X_lower = df[df.apply(aproximare_inferioara, axis=1)] # Aproximarea inferioară
15    X_upper = df[df.apply(aproximare_superioara, axis=1)] # Aproximarea superioară
16    X_boundary = X_upper[~X_upper["Nume"].isin(X_lower["Nume"])] # Regiunea de frontieră
17
18    # Afisarea rezultatelor
19    print("\nMultimea X (persoanele care indeplinesc conditiile):")
20    print(X)
21
22    print("\nAproximarea Inferioară:")
23    print(X_lower)
24
25    print("\nAproximarea Superioară:")
26    print(X_upper)
27
28    print("\nRegiunea de Frontieră:")
29    print(X_boundary)
30
31
32 if __name__ == "__main__" :
33     main()
34
```

### Funcția main:

- Integrează toate etapele, creând tabelul, calculând aproximările, și afișând rezultatele, inclusiv regulile certe și posibile.

## Tema 4 Multimi Rough



```
↑
↓
↑
↓
↺
↻
🖨
🗑

Tabelul inițial de informații:
  Nume  Varsta    Studii  Vechime  Certificat_formator
0  Marius    33  Postliceale      8             DA
1  Petre     27  Superioare       5             NU
2  Ana       20  Superioare       0             DA
3  Maria     39  Postliceale     14             NU
4  Nicu      45   Liceale       19             NU

Mulțimea X (persoanele care îndeplinesc condițiile):
  Nume  Varsta    Studii  Vechime  Certificat_formator  Conditii
0  Marius    33  Postliceale      8             DA      True
1  Petre     27  Superioare       5             NU      True
3  Maria     39  Postliceale     14             NU      True

Aproximarea Inferioară:
  Nume  Varsta    Studii  Vechime  Certificat_formator  Conditii
0  Marius    33  Postliceale      8             DA      True

Aproximarea Superioară:
  Nume  Varsta    Studii  Vechime  Certificat_formator  Conditii
0  Marius    33  Postliceale      8             DA      True
1  Petre     27  Superioare       5             NU      True
3  Maria     39  Postliceale     14             NU      True

Regiunea de Frontieră:
  Nume  Varsta    Studii  Vechime  Certificat_formator  Conditii
1  Petre     27  Superioare       5             NU      True
3  Maria     39  Postliceale     14             NU      True

Process finished with exit code 0
```

### 1. Tabelul inițial de informații

Tabelul conține informațiile inițiale despre persoanele analizate: nume, vârstă, studii, vechime și dacă posedă un certificat de formator. Este punctul de plecare pentru analiza ulterioară.

### 2. Mulțimea X

Mulțimea X include persoanele care îndeplinesc condițiile de angajare, conform regulii:

- **Vârsta** < 40
- **Studii** în „superioare” sau „postliceale”
- **Vechime** > 0

Rezultat:

- Persoanele selectate sunt: **Marius, Petre, Maria**

### 3. Aproximarea Inferioară

Aproximarea inferioară conține persoanele care îndeplinesc cu certitudine condițiile, inclusiv certificatul de formator ("DA"):

- Rezultat: doar **Marius**.

### 4. Aproximarea Superioară

Aproximarea superioară conține persoanele care pot îndeplini condițiile, dar nu neapărat cu certitudine (nu este nevoie de certificat "DA"):

- Rezultat: **Marius, Petre, Maria**.

### 5. Regiunea de Frontieră

Regiunea de frontieră este diferența dintre aproximarea superioară și inferioară, adică persoanele care sunt doar parțial în mulțimea X:

- Rezultat: **Petre, Maria**.

Aceasta reflectă incertitudinea cu privire la apartenența lor completă la X, deoarece nu au certificatul de formator.

### Observație

Rezultatele sunt coerente cu teoria mulțimilor rough:

1. **Aproximarea inferioară** este un subset al aproximării superioare.
2. **Regiunea de frontieră** reflectă elementele care nu pot fi clasificate cu certitudine.

### 2.3 Demo aplicativ – solution3.py

```
4
5 # Crearea sistemului de informatii (tabel initial)
6 data = {
7     "Nume" : ["Marius", "Petre", "Ana", "Maria", "Nicu"],
8     "Varsta" : [33, 27, 20, 39, 45],
9     "Studii" : ["Postliceale", "Superioare", "Superioare", "Postliceale", "Liceale"],
10    "Vechime" : [8, 5, 0, 14, 19],
11    "Certificat_formator" : ["DA", "NU", "DA", "NU", "NU"]
12 }
13
14
15 def creeaza_tabel() :
16     return pd.DataFrame(data)
17
18
19 def calculeaza_decizie(row) :
20     # Conditiiile de decizie
21     return row["Varsta"] < 40 and row["Studii"] in ["Superioare", "Postliceale"] and row["Vechime"] > 0
22
23
24 def indiscernabilitate(df, attribute) :
25     """Gruparea in clase de echivalență in functie de attributele specificate."""
26     echivalente = {}
27     for _, row in df.iterrows() :
28         cheie = tuple(row[atribut] for atribut in attribute)
29         if cheie not in echivalente :
30             echivalente[cheie] = []
31         echivalente[cheie].append(row["Nume"])
32     return echivalente
33
34
35 def aproximare_inferioara_superioara(clase_echivalenta, X) :
36     """Calculează aproximarea inferioară, superioară și identifică regiunea de frontieră."""
37     aproximare_inferioara = set()
38     aproximare_superioara = set()
39     regiune_frontiera = set()
40
41     for clasa in clase_echivalenta.values() :
42         clasa_set = set(clasa)
43         print(f"Clasă: {clasa_set}, Intersecție cu X: {clasa_set & X}") # Debugging
44         if clasa_set.issubset(X) :
45             aproximare_inferioara.update(clasa_set)
46         if clasa_set & X :
47             aproximare_superioara.update(clasa_set)
48         if clasa_set & X and not clasa_set.issubset(X) :
49             regiune_frontiera.update(clasa_set - X)
50
51     return aproximare_inferioara, aproximare_superioara, regiune_frontiera
52
53
54 def genereaza_reguli(clase_echivalenta, atribut_decizie, df) :
55     reguli_certe = []
56     reguli_posibile = []
57
58     for cheie, clasa in clase_echivalenta.items() :
59         clasa_decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
60         if len(clasa_decizii) == 1 :
61             reguli_certe.append((cheie, list(clasa_decizii)[0]))
62         else :
63             reguli_posibile.append((cheie, clasa_decizii))
64
65     return reguli_certe, reguli_posibile
66
```

## Tema 4 Multimi Rough

```
67
68 def calculeaza_reduct_nucleu(df, atribut_decizie) :
69     """Calculează reductele și nucleul sistemului de decizie."""
70     attribute = [col for col in df.columns if col not in ["Nume", atribut_decizie]]
71     toate_reductele = []
72
73     # Generăm toate combinațiile posibile de attribute
74     for lungime in range(1, len(attribute) + 1) :
75         for subset in combinations(attribute, lungime) :
76             subset = list(subset)
77             clasa_echivalenta = indiscernabilitate(df, subset)
78
79             # Verificăm dacă reductul păstrează deciziile
80             valid = True
81             for clasa in clasa_echivalenta.values() :
82                 decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
83                 if len(decizii) > 1 :
84                     valid = False
85                     break
86
87             if valid :
88                 toate_reductele.append(set(subset))
89                 print(f"Reduct valid: {subset}") # Debugging
90
91     # Calculăm nucleul ca intersecția tuturor reductelor
92     if toate_reductele :
93         print(f"Toate reductele: {toate_reductele}") # Debugging
94         nucleu = set.intersection(*toate_reductele)
95     else :
96         nucleu = set()
97
98     return toate_reductele, nucleu
99
```

```
101 def calculeaza_regiunea_positiva(df, clasa_echivalenta, atribut_decizie) :
102     """Calculează regiunea pozitivă pentru sistemul de decizie."""
103     regiune_positiva = set()
104     for clasa in clasa_echivalenta.values() :
105         decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
106         if len(decizii) == 1 : # Dacă toate elementele clasei au aceeași decizie
107             regiune_positiva.update(clasa)
108     return regiune_positiva
109
110
111 def functii_apartinenta_rough(clasa_echivalenta, X) :
112     """Calculează funcțiile de apartenență rough pentru fiecare element."""
113     apartenența = {}
114     for clasa in clasa_echivalenta.values() :
115         clasa_set = set(clasa)
116         intersectie = clasa_set & X
117         for element in clasa :
118             apartenența[element] = len(intersectie) / len(clasa_set) if len(clasa_set) > 0 else 0
119     return apartenența
120
```

## Tema 4 Multimi Rough

---

### 1. Crearea datelor și tabelului

Funcția `creeaza_tabel()` creează un DataFrame (df) cu datele inițiale:

- Fiecare rând reprezintă o persoană cu attribute precum vârsta, studii, vechime și certificatul de formator.
- Attributele sunt utilizate pentru a grupa datele în **clase de echivalență**.

### 2. Calcularea deciziei

Funcția `calculeaza_decizie(row)`:

- Determină dacă o persoană îndeplinește condițiile decizionale.
- O persoană este selectată dacă:
  - **Vârsta** este mai mică de 40.
  - **Studii** sunt în ["Superioare", "Postliceale"].
  - **Vechimea** este mai mare de 0.

Aceasta adaugă o coloană Decizie în tabel, cu valori True sau False.

### 3. Gruparea în clase de echivalență

Funcția `indiscernabilitate(df, attribute)`:

- Grupa datele în **clase de echivalență** pe baza atributelor specificate.
- Fiecare clasă conține persoane cu attribute identice.

### 4. Aproximări și regiunea de frontieră

Funcția `aproximare_inferioara_superioara(clase_echivalenta, X)`:

- **Aproximarea inferioară:** Clasele complet incluse în mulțimea X (persoanele cu Decizie=True).
- **Aproximarea superioară:** Clasele care intersectează mulțimea X (posibil incluse).
- **Regiunea de frontieră:** Diferența dintre aproximarea superioară și cea inferioară. Aceasta include elementele care sunt ambigue.

### 5. Generarea regulilor

Funcția `genereaza_reguli(clase_echivalenta, atribut_decizie, df)`:

- Creează reguli **certe** și **posibile**:
  - **Reguli certe:** Clasele care au o singură decizie (True sau False).
  - **Reguli posibile:** Clasele care pot avea decizii multiple.

### 6. Reducte și nucleu

Funcția calculeaza\_reduct\_nucleu(df, atribut\_decizie):

- **Reductele:** Submulțimi minimale de attribute care păstrează decizia sistemului.
- **Nucleul:** Intersecția tuturor reductelor. Dacă nucleul este gol, niciun atribut nu este indispensabil pentru decizie.

### 7. Regiunea pozitivă

Funcția calculeaza\_regiunea\_positiva(df, clase\_echivalenta, atribut\_decizie):

- Include elementele din clasele care au o singură decizie (100% certe).

### 8. Funcțiile de apartenență rough

Funcția functii\_apartinenta\_rough(clase\_echivalenta, X):

- Calculează **gradul de apartenență** al fiecărui element la mulțimea X, pe baza relației sale cu clasa de echivalență:
  - Valoarea este proporția între numărul elementelor comune clasei și mulțimii X și dimensiunea clasei.

```
121
122 def main() :
123     # Crearea tabelului
124     df = creeaza_tabel()
125     df["Decizie"] = df.apply(calculeaza_decizie, axis=1)
126
127     # Definirea atributelor pentru echivalență
128     attribute = ["Varsta", "Studii", "Vechime"]
129
130     # Gruparea în clase de echivalență
131     clase_echivalenta = indiscernabilitate(df, attribute)
132
133     # Definirea mulțimii X (persoanele cu Decizie = True)
134     X = set(df[df["Decizie"]]["Nume"])
135
136     # Calcularea aproximărilor și a regiunii de frontieră
137     aprox_inferioara, aprox_superioara, frontiera = aproximare_inferioara_superioara(clase_echivalenta, X)
138
139     # Generarea regulilor
140     reguli_certe, reguli_posibile = genereaza_reguli(clase_echivalenta, "Decizie", df)
141
142     # Calcularea reductelor și nucleului
143     reducte, nucleu = calculeaza_reduct_nucleu(df, "Decizie")
144
145     # Calcularea regiunii pozitive
146     regiune_positiva = calculeaza_regiunea_positiva(df, clase_echivalenta, "Decizie")
147
148     # Calcularea funcțiilor de apartenență rough
149     apartenenta = functii_apartinenta_rough(clase_echivalenta, X)
150
```



## Tema 4 Multimi Rough

```
151 # Afisarea rezultatelor
152 print("Tabelul initial de informatii:")
153 print(df)
154
155 print("\nClase de echivalență:")
156 for cheie, valoare in clase_echivalenta.items() :
157     print(f"{cheie}: {valoare}")
158
159 print("\nAproximarea inferioară:", aprox_inferioara)
160 print("\nAproximarea superioară:", aprox_superioara)
161 print("\nRegiunea de frontieră:", frontiera)
162
163 print("\nReguli certe:")
164 for regula in reguli_certe :
165     print(f"IF {regula[0]} THEN Decizie={regula[1]}")
166
167 print("\nReguli posibile:")
168 for regula in reguli_posibile :
169     print(f"IF {regula[0]} THEN Decizie poate fi {regula[1]}")
170
171 print("\nReducte:")
172 for reduct in reducte :
173     print(reduct)
174
175 print("\nNucleul:", nucleu)
176
177 print("\nRegiunea pozitivă:", regiune_positiva)
178
179 print("\nFuncții de apartenență rough:")
180 for element, val in apartenenta.items() :
181     print(f"{element}: {val}")
182
183
184 if __name__ == "__main__" :
185     main()
```

Integrează toate componentele și afișează rezultatele:

1. Clasele de echivalență.
2. Aproximările și regiunea de frontieră.
3. Regulile certe și posibile.
4. Reductele și nucleul.
5. Regiunea pozitivă.
6. Funcțiile de apartenență rough pentru fiecare element.

## Tema 4 Multimi Rough

Tabelul inițial de informații:

	Nume	Varsta	Studii	Vechime	Certificat_formator	Decizie
0	Marius	33	Postliceale	8	DA	True
1	Petre	27	Superioare	5	NU	True
2	Ana	20	Superioare	0	DA	False
3	Maria	39	Postliceale	14	NU	True
4	Nicu	45	Liceale	19	NU	False

Clase de echivalență:

(33, 'Postliceale', 8): ['Marius']

(27, 'Superioare', 5): ['Petre']

(20, 'Superioare', 0): ['Ana']

(39, 'Postliceale', 14): ['Maria']

(45, 'Liceale', 19): ['Nicu']

Aproximarea inferioară: {'Marius', 'Maria', 'Petre'}

Aproximarea superioară: {'Marius', 'Maria', 'Petre'}

Regiunea de frontieră: set()

Reguli certe:

IF (33, 'Postliceale', 8) THEN Decizie=True

IF (27, 'Superioare', 5) THEN Decizie=True

IF (20, 'Superioare', 0) THEN Decizie=False

IF (39, 'Postliceale', 14) THEN Decizie=True

IF (45, 'Liceale', 19) THEN Decizie=False

Reguli posibile:

Reducte:

{'Varsta'}

{'Vechime'}

{'Studii', 'Varsta'}

{'Vechime', 'Varsta'}

{'Certificat\_formator', 'Varsta'}

{'Studii', 'Vechime'}

{'Studii', 'Certificat\_formator'}

{'Vechime', 'Certificat\_formator'}

{'Studii', 'Varsta', 'Vechime'}

{'Studii', 'Certificat\_formator', 'Varsta'}

{'Vechime', 'Certificat\_formator', 'Varsta'}

{'Studii', 'Certificat\_formator', 'Vechime'}

{'Studii', 'Certificat\_formator', 'Varsta', 'Vechime'}

Nucleul: set()

Regiunea pozitivă: {'Maria', 'Petre', 'Marius', 'Nicu', 'Ana'}

Funcții de apartenență rough:

Marius: 1.0

Petre: 1.0

Ana: 0.0

Maria: 1.0

Nicu: 0.0

Process finished with exit code 0

## Tema 4 Multimi Rough

---

### Tabelul Inițial de Informații

- Tabelul include atributele persoanelor și deciziile calculate:
  - Ex: Marius: True, Ana: False.
- **Decizie:** Este determinată pe baza regulilor definite (Varsta, Studii, Vechime).

### Clasele de Echivalență

- **Exemple din output:**
  - (33, 'Postliceale', 8): ['Marius'] → Marius are această combinație unică de atribute.
  - (20, 'Superioare', 0): ['Ana'] → Clasa lui Ana nu este inclusă în X.

### Aproximări și Regiunea de Frontieră

- **Aproximarea inferioară:** {'Marius','Maria','Petre'}
  - Clase complet incluse în X.
- **Aproximarea superioară:** {'Marius','Maria','Petre'}
  - Clase care pot aparține X.
- **Regiunea de frontieră:** set() (goală):
  - Nu există clase care să fie parțial incluse în X.

### Reguli Certe

- Generate din clasele care au o decizie unică (True sau False).
- **Exemple din output:**
  - IF (33, 'Postliceale', 8) THEN Decizie=True.
  - IF (20, 'Superioare', 0) THEN Decizie=False.

### Reguli Posibile

- Nu există reguli posibile în acest caz, ceea ce indică un sistem bine definit, fără ambiguități.

### Nucleul

- Este intersecția tuturor reductelor.
- **Output:** set() (gol):

## Tema 4 Multimi Rough

---

- Nucleul gol înseamnă că nu există atribute indispensabile pentru păstrarea deciziilor.

### Regiunea Pozitivă

- Conține elementele din clasele care au o decizie certă.
- **Output:** {'Maria','Petre','Marius','Nicu','Ana'}

### Funcții de Apartenență Rough

- Gradul de apartenență al fiecărui element la X:
- **Exemple din output:**
  - Marius: 1.0 → Clasa lui este complet inclusă în X.
  - Ana: 0.0 → Clasa lui Ana nu intersectează X.
  - Maria: 1.0 → Clasa lui Maria este complet inclusă în X.

### Concluzie Generală

1. **Sistem clar definit:** Nu există ambiguități, iar toate clasele au decizii certe.
2. **Nucleul gol:** Deciziile pot fi luate folosind mai multe combinații de atribute.
3. **Regiunea de frontieră goală:** Toate clasele sunt complet incluse sau excluse din X.

### 3. Concluzii finale

- Dialogul construit în script permite utilizatorilor să specifice criterii detaliate, care sunt folosite direct pentru a defini mulțimea X de produse de interes. Această abordare îmbunătățește experiența utilizatorului și oferă rezultate personalizate.
- Folosirea Aproximărilor Rough:
  - **Aproximarea Inferioară:** Oferă o listă de produse care îndeplinesc toate criteriile, garantând satisfacția cerințelor utilizatorului fără ambiguitate.
  - **Aproximarea Superioară:** Extinde opțiunile utilizatorului la produse care pot îndeplini cel puțin una dintre condiții, mărinnd astfel diversitatea opțiunilor disponibile.
  - **Regiunea de Frontieră:** Identifică produsele pentru care există incertitudini legate de îndeplinirea criteriilor, informând utilizatorul despre posibilele riscuri sau neclarități.

### 4. Cod sursă

#### Solution.py

```
import pandas as pd

def filtreaza_produce(df, criterii):
    """Filtreaza produsele folosind criteriile specificate."""
    conditii = {
        'Categorie': df['Categorie'] == criterii['categorie'],
        'Preț ($)': df['Preț ($)'] <= criterii['pret_max'],
        'Stoc': df['Stoc'] == criterii['stoc'],
        'Rating mediu': df['Rating mediu'] >= criterii['rating_min'],
        'Vândut online': df['Vândut online'] == criterii['vandut_online'],
        'An lansare': df['An lansare'] >= criterii['an_min']
    }

    # C filtru pentru aproximarea inferioară si superioara
    filtru_inferior = pd.Series([True] * len(df))
    filtru_superior = pd.Series([False] * len(df))

    for key, value in conditii.items():
        filtru_inferior &= value
        filtru_superior |= value

    aproximare_inferioara = df[filtru_inferior]
    aproximare_superioara = df[filtru_superior]
    regiune_frontiera = aproximare_superioara[~aproximare_superioara.index.isin(aproximare_inferioara.index)]

    return aproximare_inferioara, aproximare_superioara, regiune_frontiera

def afiseaza_rezultate(aprox_inferioara, aprox_superioara, reg_frontiera):
    """Afișează rezultatele filtrării."""
    print("Aproximare inferioară:")
    print(aprox_inferioara)
    print("\nAproximare superioară:")
    print(aprox_superioara)
    print("\nRegiune de frontieră:")
    print(reg_frontiera)

def main():
    data = {
        'ID': [1, 2, 3, 4, 5, 6, 7, 9],
        'Produs': ['Telefon mobil Samsung', 'Tabletă Samsung', 'Laptop Asus', 'Smartwatch', 'Consolă Gaming',
                  'Laptop Dell', 'Telefon iPhone', 'Camera Canon'],
        'Categorie': ['Smartphone', 'Tabletă', 'Laptop', 'Ceas', 'Gaming', 'Laptop', 'Smartphone', 'Aparat Compact'],
        'Preț ($)': [300, 450, 800, 200, 350, 750, 350, 400],
        'Stoc': ['DA', 'NU', 'DA', 'NU', 'NU', 'DA', 'DA', 'DA'],
        'Vândut online': ['DA', 'DA', 'NU', 'DA', 'NU', 'DA', 'DA', 'DA'],
        'Rating mediu': [4.7, 4.0, 4.8, 4.0, 4.5, 4.3, 5.0, 4.3],
        'An lansare': [2021, 2020, 2022, 2023, 2023, 2022, 2024, 2021]
```

## Tema 4 Multimi Rough

---

```
}

df = pd.DataFrame(data)

criterii = {
    'categorie': input("Introduceți categoria produsului (ex. Laptop, Smartphone): "),
    'pret_max': int(input("Introduceți prețul maxim dorit: ")),
    'stoc': input("Produsul trebuie să fie în stoc? (DA/NU): "),
    'rating_min': float(input("Introduceți ratingul minim dorit: ")),
    'vandut_online': input("Produsul trebuie să fie vândut online? (DA/NU): "),
    'an_min': int(input("Introduceți anul minim de lansare: "))
}

# Aplică filtrarea
aprox_inferioara, aprox_superioara, reg_frontiera = filtreaza_produce(df, criterii)

# Afișează rezultatele
afiseaza_rezultate(aprox_inferioara, aprox_superioara, reg_frontiera)

if __name__ == "__main__":
    main()
```

### Solution2.py

```
import pandas as pd
```

```
def creeaza_tabel():
    # Crearea tabelului de informații
    data = {
        "Nume": ["Marius", "Petre", "Ana", "Maria", "Nicu"],
        "Varsta": [33, 27, 20, 39, 45],
        "Studii": ["Postliceale", "Superioare", "Superioare", "Postliceale", "Liceale"],
        "Vechime": [8, 5, 0, 14, 19],
        "Certificat_formator": ["DA", "NU", "DA", "NU", "NU"]
    }
    return pd.DataFrame(data)

def indeplineste_conditii(row):
    # Condițiile de angajare
    return row["Varsta"] < 40 and row["Studii"] in ["Superioare", "Postliceale"] and row["Vechime"] > 0

def aproximare_inferioara(row):
    # Aparține sigur dacă îndeplinește condițiile și are Certificat formator = "DA"
    return row["Conditii"] and row["Certificat_formator"] == "DA"

def aproximare_superioara(row):
    # Poate aparține dacă îndeplinește condițiile
    return row["Conditii"]
```

## Tema 4 Multimi Rough

---

```
def main() :
    # Crearea tabelului
    df = creeaza_tabel()
    print("Tabelul inițial de informații:")
    print(df)

    # Adăugarea unei coloane pentru condiții
    df["Conditii"] = df.apply(indeplineste_conditii, axis=1)

    # Definirea mulțimii X
    X = df[df["Conditii"]]

    # Calcularea aproximărilor
    X_lower = df[df.apply(aproximare_inferioara, axis=1)] # Aproximarea inferioară
    X_upper = df[df.apply(aproximare_superioara, axis=1)] # Aproximarea superioară
    X_boundary = X_upper[~X_upper["Nume"].isin(X_lower["Nume"])] # Regiunea de frontieră

    # Afișarea rezultatelor
    print("\nMulțimea X (persoanele care îndeplinesc condițiile):")
    print(X)

    print("\nAproximarea Inferioară:")
    print(X_lower)

    print("\nAproximarea Superioară:")
    print(X_upper)

    print("\nRegiunea de Frontieră:")
    print(X_boundary)

if __name__ == "__main__" :
    main()
```

### Solution3.py

```
from itertools import combinations
```

```
import pandas as pd
```

```
# Crearea sistemului de informații (tabel inițial)
```

```
data = {
    "Nume" : ["Marius", "Petre", "Ana", "Maria", "Nicu"],
    "Varsta" : [33, 27, 20, 39, 45],
    "Studii" : ["Postliceale", "Superioare", "Superioare", "Postliceale", "Liceale"],
    "Vechime" : [8, 5, 0, 14, 19],
    "Certificat_formator" : ["DA", "NU", "DA", "NU", "NU"]
}
```

## Tema 4 Multimi Rough

---

```
def creeaza_tabel() :
    return pd.DataFrame(data)

def calculeaza_decizie(row) :
    # Condițiile de decizie
    return row["Varsta"] < 40 and row["Studii"] in ["Superioare", "Postliceale"] and row["Vechime"] > 0

def indiscernabilitate(df, atribute) :
    """Gruparea în clase de echivalență în funcție de atributele specificate."""
    echivalente = {}
    for _, row in df.iterrows() :
        cheie = tuple(row[atribut] for atribut in atribute)
        if cheie not in echivalente :
            echivalente[cheie] = []
            echivalente[cheie].append(row["Nume"])
    return echivalente

def aproximare_inferioara_superioara(clase_echivalenta, X) :
    """Calculează aproximarea inferioară, superioară și identifică regiunea de frontieră."""
    aproximare_inferioara = set()
    aproximare_superioara = set()
    regiune_frontiera = set()

    for clasa in clase_echivalenta.values() :
        clasa_set = set(clasa)
        print(f"Clasă: {clasa_set}, Intersecție cu X: {clasa_set & X}") # Debugging
        if clasa_set.issubset(X) :
            aproximare_inferioara.update(clasa_set)
        if clasa_set & X :
            aproximare_superioara.update(clasa_set)
        if clasa_set & X and not clasa_set.issubset(X) :
            regiune_frontiera.update(clasa_set - X)

    return aproximare_inferioara, aproximare_superioara, regiune_frontiera

def genereaza_reguli(clase_echivalenta, atribut_decizie, df) :
    reguli_certe = []
    reguli_posibile = []

    for cheie, clasa in clase_echivalenta.items() :
        clasa_decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
        if len(clasa_decizii) == 1 :
            reguli_certe.append((cheie, list(clasa_decizii)[0]))
        else :
            reguli_posibile.append((cheie, clasa_decizii))

    return reguli_certe, reguli_posibile
```



## Tema 4 Multimi Rough

```
def calculeaza_reduct_nucleu(df, atribut_decizie) :
    """Calculează reductele și nucleul sistemului de decizie."""
    attribute = [col for col in df.columns if col not in ["Nume", atribut_decizie]]
    toate_reductele = []

    # Generăm toate combinațiile posibile de attribute
    for lungime in range(1, len(attribute) + 1) :
        for subset in combinations(attribute, lungime) :
            subset = list(subset)
            clase_echivalenta = indiscernabilitate(df, subset)

            # Verificăm dacă reductul păstrează deciziile
            valid = True
            for clasa in clase_echivalenta.values() :
                decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
                if len(decizii) > 1 :
                    valid = False
                    break

            if valid :
                toate_reductele.append(set(subset))
                print(f"Reduct valid: {subset}") # Debugging

    # Calculăm nucleul ca intersecția tuturor reductelor
    if toate_reductele :
        print(f"Toate reductele: {toate_reductele}") # Debugging
        nucleu = set.intersection(*toate_reductele)
    else :
        nucleu = set()

    return toate_reductele, nucleu

def calculeaza_regiunea_positiva(df, clase_echivalenta, atribut_decizie) :
    """Calculează regiunea pozitivă pentru sistemul de decizie."""
    regiune_positiva = set()
    for clasa in clase_echivalenta.values() :
        decizii = set(df[df["Nume"].isin(clasa)][atribut_decizie])
        if len(decizii) == 1 : # Dacă toate elementele clasei au aceeași decizie
            regiune_positiva.update(clasa)
    return regiune_positiva

def functii_apartinenta_rough(clase_echivalenta, X) :
    """Calculează funcțiile de apartenență rough pentru fiecare element."""
    apartenenta = { }
    for clasa in clase_echivalenta.values() :
        clasa_set = set(clasa)
        intersectie = clasa_set & X
        for element in clasa :
            apartenenta[element] = len(intersectie) / len(clasa_set) if len(clasa_set) > 0 else 0
    return apartenenta
```

## Tema 4 Multimi Rough

---

```
def main() :
    # Crearea tabelului
    df = creeaza_tabel()
    df["Decizie"] = df.apply(calculeaza_decizie, axis=1)

    # Definirea atributelor pentru echivalență
    attribute = ["Varsta", "Studii", "Vechime"]

    # Gruparea în clase de echivalență
    clase_echivalenta = indiscernabilitate(df, attribute)

    # Definirea mulțimii X (persoanele cu Decizie = True)
    X = set(df[df["Decizie"]]["Nume"])

    # Calcularea aproximărilor și a regiunii de frontieră
    aprox_inferioara, aprox_superioara, frontiera = aproximare_inferioara_superioara(clase_echivalenta, X)

    # Generarea regulilor
    reguli_certe, reguli_posibile = genereaza_reguli(clase_echivalenta, "Decizie", df)

    # Calcularea reductelor și nucleului
    reducte, nucleu = calculeaza_reduct_nucleu(df, "Decizie")

    # Calcularea regiunii pozitive
    regiune_positiva = calculeaza_regiunea_positiva(df, clase_echivalenta, "Decizie")

    # Calcularea funcțiilor de apartenență rough
    apartenenta = functii_apartinenta_rough(clase_echivalenta, X)

    # Afișarea rezultatelor
    print("Tabelul inițial de informații:")
    print(df)

    print("\nClase de echivalență:")
    for cheie, valoare in clase_echivalenta.items() :
        print(f"{cheie}: {valoare}")

    print("\nAproximarea inferioară:", aprox_inferioara)
    print("Aproximarea superioară:", aprox_superioara)
    print("Regiunea de frontieră:", frontiera)

    print("\nReguli certe:")
    for regula in reguli_certe :
        print(f"IF {regula[0]} THEN Decizie={regula[1]}")

    print("\nReguli posibile:")
    for regula in reguli_posibile :
        print(f"IF {regula[0]} THEN Decizie poate fi {regula[1]}")

    print("\nReducte:")
    for reduct in reducte :
        print(reduct)
```

## Tema 4 Multimi Rough

---

```
print("\nNucleul:", nucleu)

print("\nRegiunea pozitivă:", regiune_positiva)

print("\nFuncții de apartenență rough:")
for element, val in apartenenta.items():
    print(f"{element}: {val}")

if __name__ == "__main__":
    main()
```