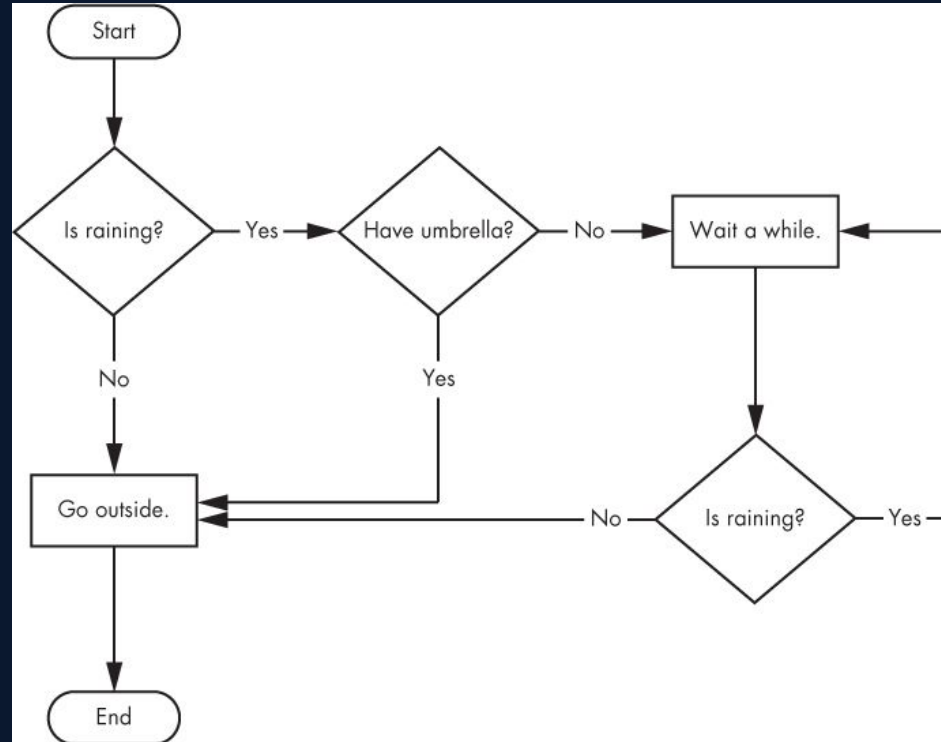


Controlul fluxului de programare - Partea 1

Controlul fluxului de programare [1]





Valorile Booleene

În informatică, tipul de date Boolean este un tip de date care este destinat să reprezinte cele două valori de adevăr ale logicii și algebrei booleene. Acestea au fost denumite după George Boole, care a definit pentru prima dată un sistem algebric de logică la mijlocul secolului al XIX-lea.

În timp ce tipurile de date întregi, cu virgulă mobilă și șir de caractere au un număr nelimitat de valori posibile, tipul de date boolean are doar două valori: **Adevărat** și **Fals**.



Operațiile de comparație

Operatorii de comparație, numiți și operatori relaționali, compară două valori booleene (sau logice) și le evaluează până la o singură valoare booleană.




Operațiile de comparație

Operator	Operație
<code>==</code>	Egalitate
<code>!=</code>	Diferență
<code><</code>	Mai mic decât
<code>></code>	Mai mare decât
<code><=</code>	Mai mic decât sau egal
<code>>=</code>	Mai mare decât sau egal



Operațiile Booleene

Cei trei operatori Booleani *and*, *or* și *not* (*și*, *sau*, și *nu logic*) sunt utilizați pentru a opera cu valorile booleene. La fel ca operatorii de comparație, aceste expresii se evaluează până la o valoare booleană.



Operația ȘI - AND

Tabelul de adevăr

Expresie	Rezultat
True and True	True
True and False	False
False and True	False
False and False	False



Operația SAU - OR

Tabelul de adevăr

Expresie	Rezultat
True or True	True
True or False	True
False or True	True
False or False	False



Operația NU LOGIC - NOT

Tabelul de adevăr

Expresie	Rezultat
<code>not True</code>	<code>False</code>
<code>not False</code>	<code>True</code>



Combinații ale operațiilor de comparație și Booleene [1]

```
>>> (4 < 5) and (5 < 6)
```

```
True
```

```
>>> (4 < 5) and (9 < 6)
```

```
False
```

```
>>> (1 == 2) or (2 == 2)
```

```
True
```

Combinatii ale operațiilor de comparație și Booleene [1]

$(4 < 5) \text{ and } (5 < 6)$
↓
True and (5 < 6)
↓
True and True
↓
True



Elemente ale fluxului de control [1]

Instrucțiunile de control al fluxului încep adesea cu o parte numită *condiție* și sunt întotdeauna urmate de un *bloc de cod* numit *clauză*. Înainte de a afla despre declarațiile specifice de control al fluxului Python, trebuie discutate care sunt *condițiile* și care sunt *blocurile de cod*.

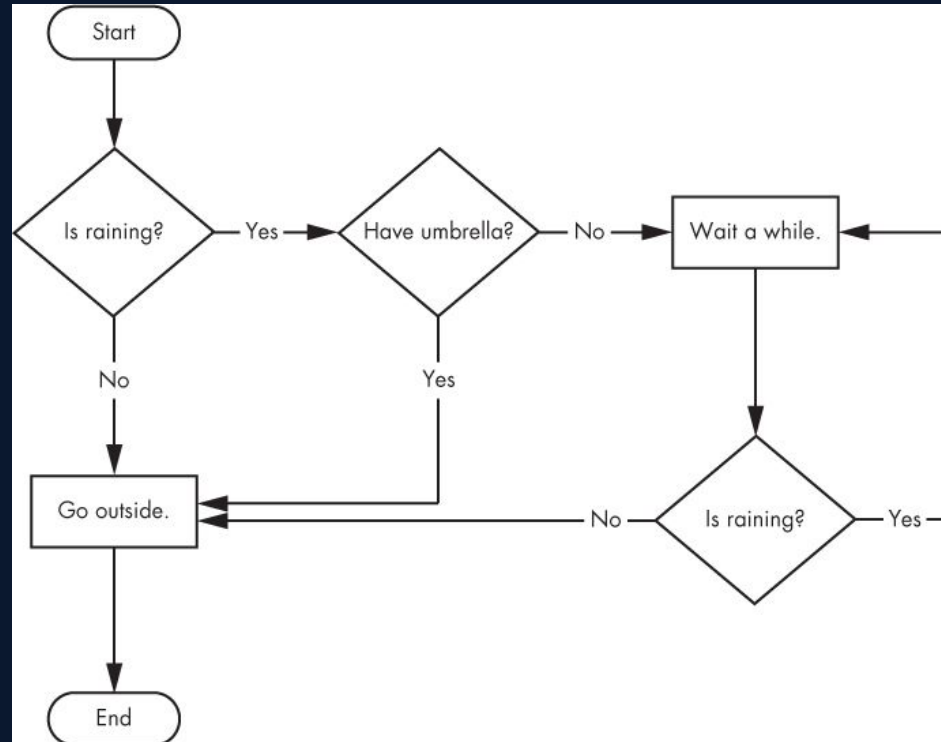


Elemente ale fluxului de control [1]

Expresiile Booleene pe care le-ați văzut până acum ar putea fi considerate *condiții*, care sunt același lucru cu expresiile; condiția este doar un nume mai specific în contextul instrucțiunilor de control ale fluxului de programare.

Condițiile se evaluează întotdeauna până la o valoare booleană, *Adevărat* sau *Fals*. O instrucțiune de control al fluxului decide ce să facă în funcție de starea sa, *adevărată sau falsă*.

Controlul fluxului de programare [1]





Elemente ale fluxului de control [1]

Liniile de cod Python pot fi grupate împreună în blocuri. Puteți afla când începe și se termină un bloc după indentarea liniilor de cod. Există trei reguli pentru blocuri.

- Blocurile încep când indentarea crește.
- Blocurile pot conține alte blocuri.
- Blocurile se termină atunci când indentarea scade la zero sau la indentarea unui bloc de care acesta aparține.



Instrucțiunea IF [1]

```
1. name = 'Mary'  
2. password = 'swordfish'  
3. if name == 'Mary':  
4.     print('Hello, Mary')  
5.     if password == 'swordfish':  
6.         print('Access granted.')  
7.     else:  
8.         print('Wrong password.')
```



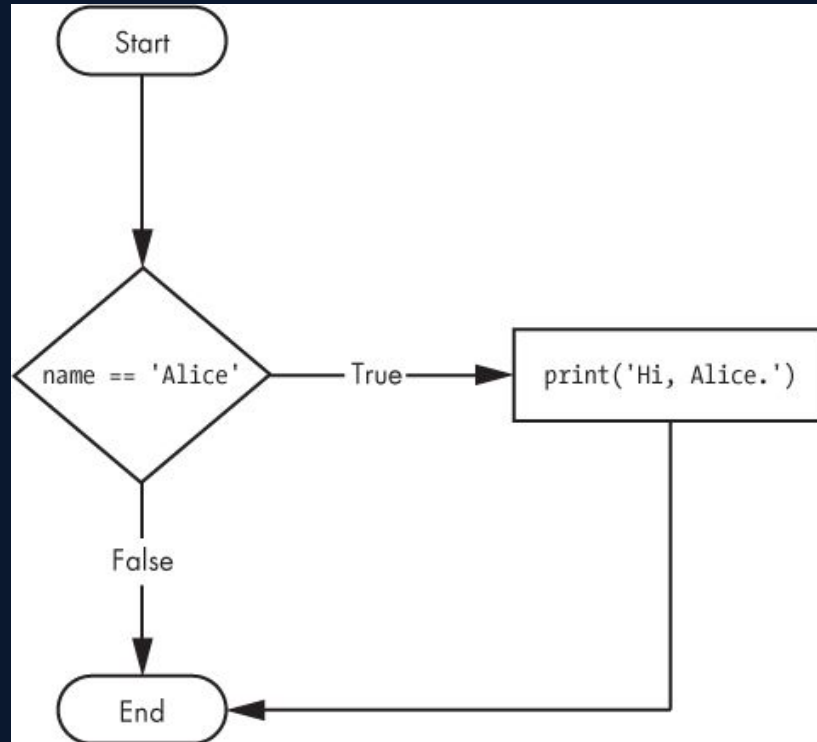

Instrucțiunea IF [1]

Cel mai comun tip de instrucțiune de control al fluxului este instrucțiunea *if*. Clauza unei instrucțiuni *if* (adică blocul care urmează instrucțiunii *if*) se va executa în cazul în care *condiția declarației* este **Adevărată**. Clauza este omisă în cazul în care condiția este falsă.

În română, o declarație *if* ar putea fi citită ca „Dacă această condiție este adevărată, executați codul din clauză”. În Python, o instrucțiune *if* constă din următoarele:

- Cuvântul cheie *if*;
- O condiție (adică o expresie care se evaluează la adevărat sau fals);
- Două puncte;
- Începând cu următoarea linie, un bloc de cod indentat (numit *clauza if*).

Instrucțiunea IF [1]



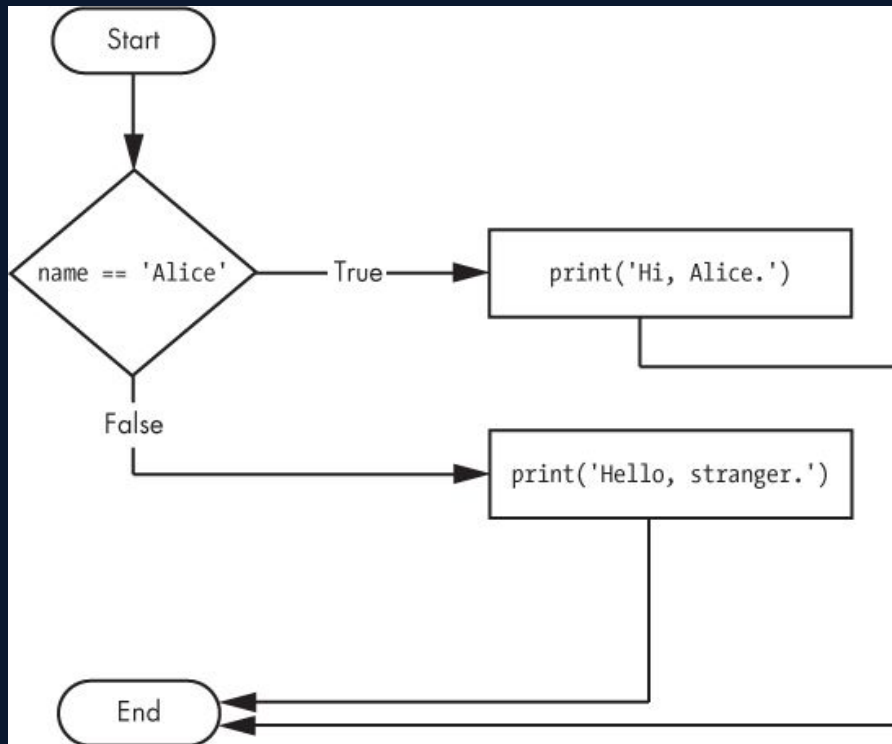


Instrucțiunea ELSE [1]

O clauză *if* poate fi urmată, opțional, de o instrucțiune *else*. Clauza *else* este executată numai atunci când condiția instrucțiunii *if* este **Falsă**. În română, o afirmație *else* ar putea fi citită ca „Dacă această condiție este adevărată, executați acest cod. Sau altfel, executați acel cod.” O instrucțiune *else* nu are o condiție și, în cod, o instrucțiune *else* constă întotdeauna în următoarele:

- Cuvântul cheie ***else***;
- Două puncte;
- Începând cu următoarea linie, un bloc de cod indentat (numit *clauza else*).

Instrucțiunea ELSE [1]



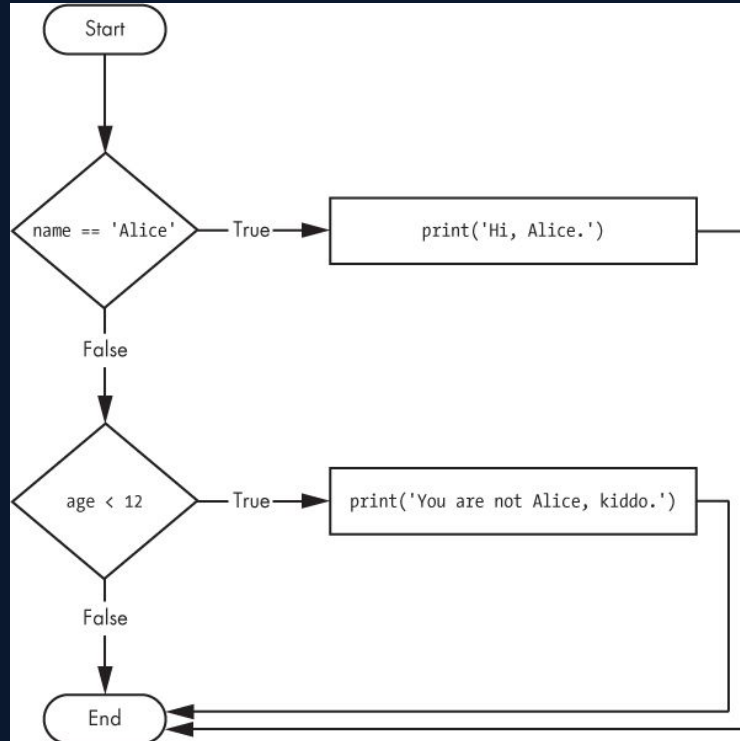


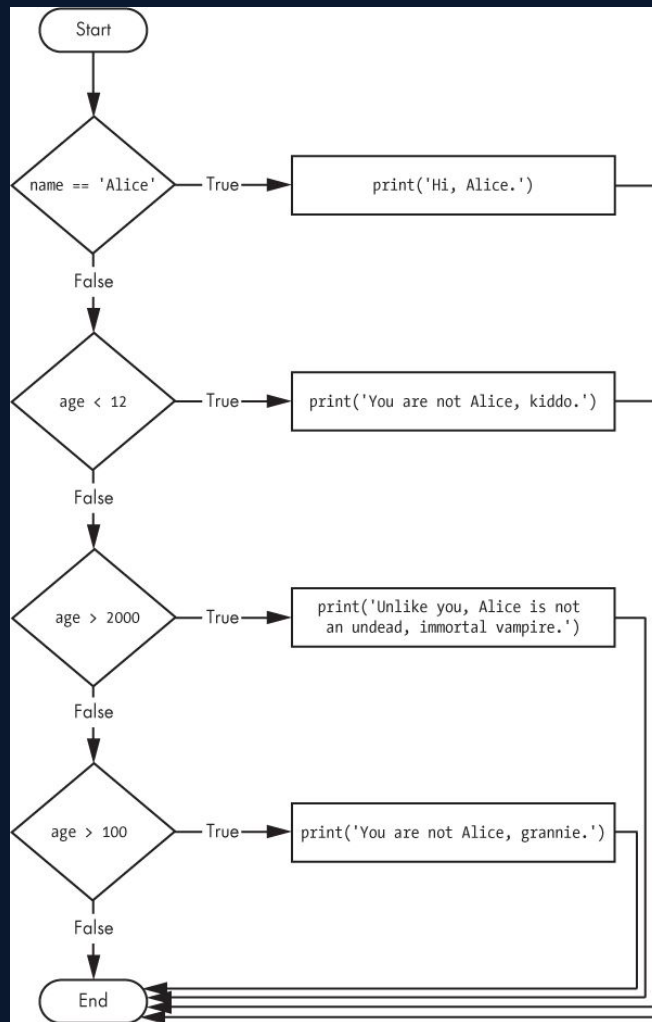
Instrucțiunea ELIF [1]

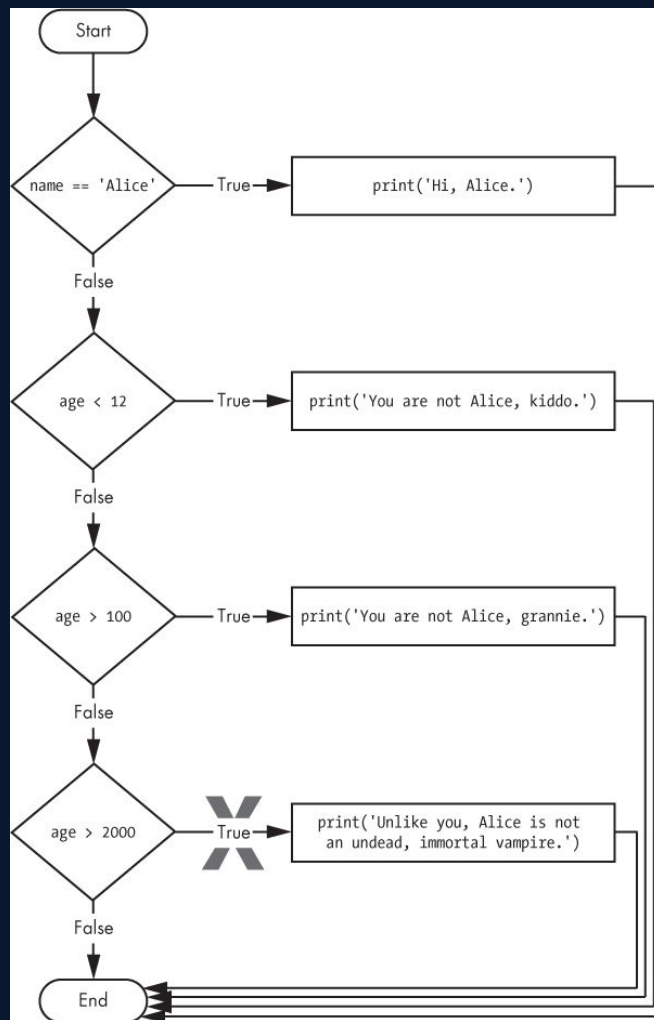
În timp ce se poate executa doar una dintre clauzele *if* sau *else*, este posibil să aveți un caz în care doriți să se execute una dintre multele clauze posibile. Declarația *elif* este o declarație „*else if*” care urmează întotdeauna unei afirmații *if* sau a unei alte afirmații *elif*. *elif* oferă o altă condiție care este verificată numai dacă toate condițiile anterioare sunt false. În cod, o declarație *elif* constă întotdeauna din următoarele:

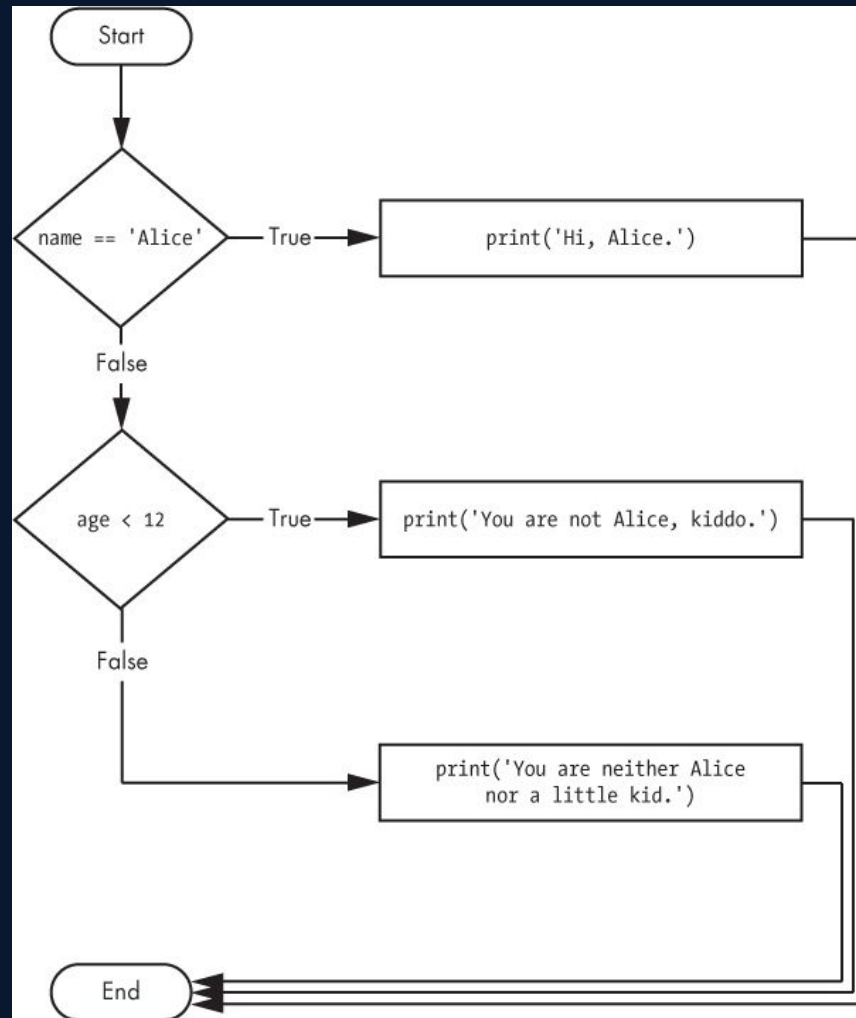
- Cuvântul cheie ***elif***;
- O condiție (adică o expresie care se evaluează la adevărat sau fals);
- Două puncte;
- Începând cu următoarea linie, un bloc de cod indentat (numit *clauza elif*).

Instrucțiunea ELIF [1]











Instrucțiuni noi în Python - `input()`

Instrucțiunea `foo = input()` așteaptă ca utilizatorul să introducă text pe tastatură și să apese ENTER. Aceasta se evaluează la un șir egal cu textul utilizatorului, iar linia de cod atribuie variabilei `foo` valoarea șirului de caractere.



Instrucțiuni noi în Python - `len()`

Puteți transmite funcției `len(foo)` o valoare de tip șir de caractere (sau o variabilă care conține un șir de caractere), iar funcția se evaluează la valoarea întreagă a numărului de caractere din acel șir.



Exemplu [1]

```
print('What is your age?') # ask for their age  
  
myAge = input()  
  
print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

Exemplu [1]

```
print('You will be ' + str(int(myAge) + 1) + ' in a year.')
print('You will be ' + str(int( '4' ) + 1) + ' in a year.')
print('You will be ' + str(    4 + 1    ) + ' in a year.')
print('You will be ' + str(        5        ) + ' in a year.')
print('You will be ' +          '5'          + ' in a year.')
print('You will be 5'                + ' in a year.')
print('You will be 5 in a year.')
```



Temă

Pentru data viitoare, scrieți un program care interoghează utilizatorul pentru următoarele:

- nume;
- vârstă;
- listă de cumpărături (stocată ca șir de caractere).

Dacă lista de cumpărături este goală atunci i se va comunica utilizatorului să se întoarcă cu o listă de cumpărături iar programul se va încheia. Dacă lista de cumpărături are conținut i se va comunica utilizatorului că durata de așteptare pentru a fi servit cu produsele din listă este egală cu numărul de caractere al listei de cumpărături ca număr de secunde.



Referințe

1 - Automate the Boring Stuff with Python, 2nd Edition

- by Al Sweigart
- Publisher: No Starch Press
- Release Date: November 2019
- ISBN: 9781593279929



That's all Folks!