Depanarea programelor Python în PyCharm

Tutorial pas cu pas pentru începători

Acest ghid explică modul în care putem opri temporar un program, îl putem executa linie cu linie și putem urmări valorile variabilelor. La final, elevul va putea folosi breakpoint-uri, comenzile Step Over și Resume, panoul Threads & Variables și consola de depanare.

Ce vei învăța

- să deosebești o eroare de sintaxă de o eroare de logică;
- să plasezi și să elimini un breakpoint;
- să pornești, să urmărești și să oprești o sesiune Debug;
- să explici evoluția variabilelor într-o buclă;
- să identifici cauza unei valori finale greșite.

Recomandare pentru lecție: demonstrați mai întâi exemplul cu suma cifrelor, apoi lăsați elevii să rezolve individual exemplele 2-4.

1 Ce înseamnă depanarea

Depanarea sau debugging-ul este investigarea controlată a execuției unui program. Nu înseamnă doar corectarea erorilor de scriere. Depanatorul ne ajută mai ales atunci când programul se execută, dar produce un rezultat greșit.

Tip de problemă	Exemplu	Cum ajută depanatorul
Eroare de sintaxă	Lipsește caracterul : după while	PyCharm o marchează înaintea executării.
Eroare la executare	Împărțire la zero	Programul se oprește și arată linia problemei.
Eroare de logică	Suma începe de la 1 în loc de 0	Urmărim variabilele și descoperim momentul în care valoarea devine greșită.

Ideea centrală

În modul Debug, programul execută aceleași instrucțiuni ca în modul Run, dar îi putem controla ritmul și putem inspecta starea lui internă.

2 Instrumentele principale

Instrument	Rol	Scurtătură Windows	Când îl folosim
Breakpoint	Oprește programul înainte de executarea liniei marcate	clic în marginea stângă	La linia pe care vrem să o analizăm
Debug	Pornește programul sub controlul depanatorului	Shift + F9	La începutul investigației
Step Over	Execută linia curentă și trece la următoarea	F8	Pentru urmărire linie cu linie
Step Into	Intră în funcția apelată	F7	Numai când vrem să analizăm funcția
Resume	Continuă până la următorul breakpoint	F9	Pentru a sări peste zone deja verificate
Stop	Oprește sesiunea	Ctrl + F2	La final sau când vrem să reluăm

Atenție: pentru primele exerciții folosiți F8, nu F7. F7 poate deschide fișiere interne ale Pythonului, de exemplu runpy.py. Nu este o eroare și fișierul nu trebuie modificat.

3 Prima depanare pas cu pas

Exemplul 1 Suma cifrelor

Scop: pentru n = 123, programul trebuie să afișeze 6.

```
n = int(input("n = "))
s = 0
while n > 0:
    c = n % 10
    s = s + c
    n = n // 10
print(s)
```

Pașii de lucru:

1. Faceți clic în marginea stângă a liniei c = n % 10. Apare un cerc roșu.
2. Porniți Debug cu Shift + F9 și introduceți 123 în consolă.
3. Programul se oprește înaintea liniei marcate. Observați n = 123 și s = 0.
4. Apăsați F8 și urmăriți apariția variabilei c.
5. Continuați cu F8 până când n devine 0 și se execută print(s).

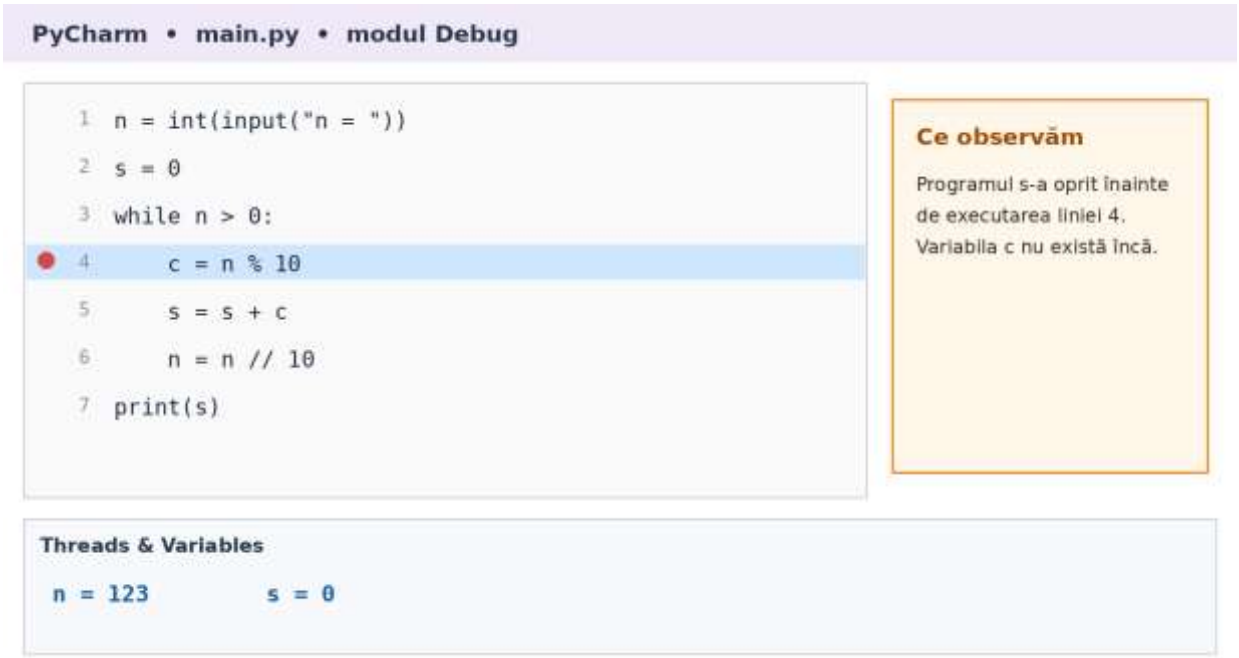


Figura 1 Program oprit la breakpoint

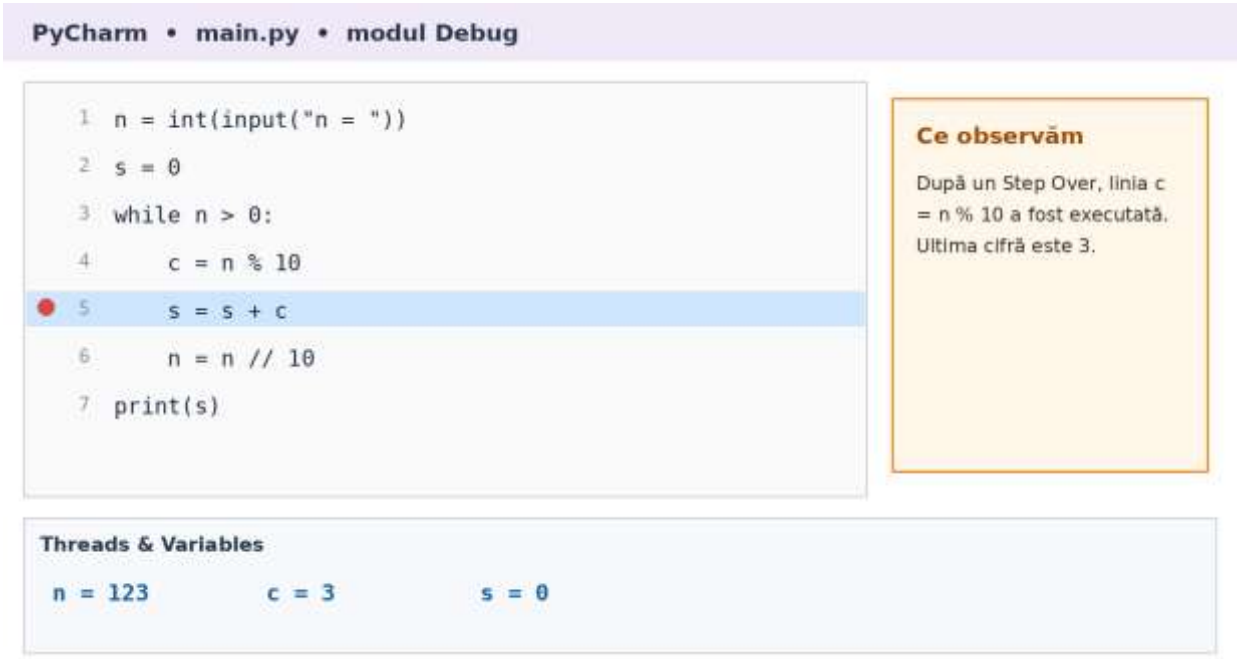


Figura 2 Valoarea lui c după primul Step Over

Tabelul de urmărire

Pas	Linia executată	n înainte	c	s după	n după
1	c = n % 10	123	3	0	123
2	s = s + c	123	3	3	123
3	n = n // 10	123	3	3	12
4	c = n % 10	12	2	3	12
5	s = s + c	12	2	5	12
6	n = n // 10	12	2	5	1
7	c = n % 10	1	1	5	1
8	s = s + c	1	1	6	1
9	n = n // 10	1	1	6	0

4 Cum citim fereastra Debug

Linia albastră: Este următoarea linie care urmează să fie executată, nu neapărat linia deja executată.

Threads & Variables: Afișează variabilele existente în momentul opririi și valorile lor curente.

Console: Primește datele citite cu input și afișează rezultatele trimise prin print.

MainThread și cadrele: Arată traseul apelurilor. Pentru exerciții simple, selectați cadrul care conține main.py.

Cercul roșu: Marchează un breakpoint activ. Un nou clic îl elimină.

De ce nu apare încă o variabilă

O variabilă apare în panou numai după ce instrucțiunea care o creează a fost executată. Dacă programul este oprit înainte de c = n % 10, variabila c nu există încă.

Run și Debug nu sunt același lucru

Caracteristică	Run	Debug
Viteză	Execută fără opriri	Poate opri la breakpoint
Valori intermediare	Nu sunt afișate automat	Se văd în Variables
Control pas cu pas	Nu	Da, prin F8/F7
Utilizare	Verificarea rezultatului final	Investigarea comportamentului

5 Exemple progresive

Exemplul 2 Maximul dintre două numere

```
a = int(input("a = "))
b = int(input("b = "))
if a > b:
    maxim = a
else:
    maxim = b
print(maxim)
```

Activitate: Puneți breakpoint la linia if a > b. Testați a = 7, b = 4, apoi a = 2, b = 9. Urmăriți ramura aleasă și valoarea lui maxim.

Exemplul 3 Numărarea numerelor pare

```
n = int(input("n = "))
contor = 0
for i in range(1, n + 1):
    if i % 2 == 0:
        contor = contor + 1
print(contor)
```

Activitate: Puneți breakpoint la if. Pentru n = 6, urmăriți i și contor. Contorul se modifică numai pentru 2, 4 și 6.

Exemplul 4 Căutarea într-o listă

```
v = [4, 7, 2, 9]
x = int(input("x = "))
gasit = False
for element in v:
    if element == x:
        gasit = True
print(gasit)
```

Activitate: Testați x = 2 și x = 5. Urmăriți element și gasit. Observați că bucla continuă chiar după găsire; ulterior puteți discuta instrucțiunea break.

Exemplul 5 O funcție proprie

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

n = int(input("n = "))
p = patrat(n)
print(p)
```

Activitate: Puneți breakpoint la p = patrat(n). Folosiți F7 o singură dată pentru a intra în funcția patrat, apoi F8 pentru liniile din funcție.

6 Depistarea unei erori de logică

Programul următor ar trebui să calculeze suma numerelor de la 1 la n, dar pentru n = 4 afișează 11 în loc de 10.

```
n = int(input("n = "))
s = 1
for i in range(1, n + 1):
    s = s + i
print(s)
```

Investigație: puneți breakpoint la s = s + i și urmăriți s. Înainte de prima adunare, s este deja 1. Cauza este inițializarea greșită. Corectarea este s = 0.

i	s înainte	s după	Observație
1	1	2	Deja cu 1 prea mare
2	2	4	Eroarea se păstrează
3	4	7	Eroarea se păstrează
4	7	11	Rezultat final greșit

Metoda de investigație

6. Stabiliți rezultatul corect pentru un exemplu mic.
7. Alegeți variabilele care descriu progresul algoritmului.
8. Plasați breakpoint-ul înaintea primei modificări importante.
9. Comparați valorile observate cu valorile calculate manual.
10. Corectați o singură cauză și testați din nou.

7 Breakpoint-uri mai utile

Breakpoint condiționat

Într-o buclă lungă nu este eficient să apăsați F9 de zeci de ori. Faceți clic dreapta pe breakpoint și introduceți o condiție, de exemplu i == 50. Programul se va opri numai când condiția este adevărată.

Mai multe breakpoint-uri

Puteți marca mai multe linii. F9 continuă execuția până la următorul breakpoint. Această metodă este utilă pentru a compara starea programului înainte și după o etapă importantă.

Dezactivare fără ștergere

Breakpoint-urile pot fi dezactivate temporar din fereastra Breakpoints. Ele rămân configurate, dar nu mai opresc programul.

8 Greșeli frecvente și soluții

Situație	Cauză probabilă	Soluție
Programul nu se oprește	A fost pornit cu Run	Porniți cu Debug sau Shift + F9.
Breakpoint-ul este ignorat	Linia nu este executată	Alegeți o linie din ramura care se execută.
S-a deschis runpy.py	Ați folosit F7	Închideți fila și folosiți F8.
Nu apare variabila c	Linia care o creează nu a fost executată	Apăsați o dată F8.
Programul așteaptă	input solicită o valoare	Deschideți Console, scrieți valoarea și apăsați Enter.
Valorile dispar	Programul s-a terminat	Porniți din nou Debug și opriți la breakpoint.
Nu puteți edita codul	Sesiunea este oprită într-un punct	Opriți cu Ctrl + F2, editați, apoi reporniți.

9 Activități practice

A. Depanați programul pentru suma cifrelor cu n = 407. Completați valorile lui n, c și s după fiecare parcurgere.

B. În exemplul maximului, testați și cazul a = b. Decideți dacă programul rămâne corect.

C. Modificați intenționat contor = 1 în exemplul numerelor pare. Folosiți depanatorul pentru a demonstra cauza rezultatului greșit.

D. În exemplul listei, adăugați break după gasit = True. Observați cum se schimbă numărul de pași.

10 Chestionar final

Alegeți un singur răspuns corect. Fiecare item valorează 1 punct; se acordă 1 punct din oficiu. Timp recomandat: 10 minute.

1. Ce este un breakpoint?

- a) O eroare de sintaxă
- b) Un punct în care execuția se oprește temporar
- c) O comandă care șterge variabilele
- d) Rezultatul programului

2. Cu ce comandă executăm linia curentă fără a intra în funcțiile apelate?

- a) Step Over
- b) Step Into
- c) Stop
- d) Run

3. Ce scurtătură corespunde comenzii Step Over în configurația standard Windows?

- a) F7
- b) F8
- c) F9
- d) Ctrl + F2

4. Unde urmărim valorile variabilelor?

- a) Project
- b) Threads & Variables
- c) Version Control
- d) External Libraries

5. Ce înseamnă linia albastră din timpul depanării?

- a) Linia conține sigur o eroare
- b) Linia urmează să fie executată
- c) Linia a fost ștearsă
- d) Linia este un comentariu

6. De ce poate fi deschis runpy.py?

- a) Python este dezinstalat
- b) A fost folosit Step Into și s-a intrat în cod intern
- c) Breakpoint-ul este greșit
- d) Programul nu are input

7. Cum continuăm până la următorul breakpoint?

- a) Resume, F9
- b) Step Over, F8
- c) Stop, Ctrl + F2
- d) Delete

8. Când apare o variabilă în panoul Variables?

- a) Înainte de scrierea programului
- b) După executarea instrucțiunii care o creează
- c) Numai după print
- d) Numai dacă este text

9. Cum oprim sesiunea Debug?

- a) Shift + F9
- b) F8
- c) Ctrl + F2
- d) Alt + Enter

Barem și răspunsuri

Răspunsuri: 1 b, 2 a, 3 b, 4 b, 5 b, 6 b, 7 a, 8 b, 9 c.

Punctaj: 9 puncte pentru răspunsuri + 1 punct din oficiu. Nota recomandată este punctajul total.

Răspunsuri orientative pentru activități

A. Pentru 407: (n=407, c=7, s=7), (n=40, c=0, s=7), (n=4, c=4, s=11), apoi n=0. B. Pentru valori egale se execută ramura else, iar maxim primește valoarea comună; rezultatul este corect. C. Contorul pornește cu o unitate în plus. D. Bucla se încheie imediat după găsirea valorii.

Fișă rapidă de memorat

Acțiune	Comandă
Pornește depanarea	Shift + F9
Execută o linie	F8
Intră într-o funcție proprie	F7
Continuă până la următorul breakpoint	F9
Oprește depanarea	Ctrl + F2