Python Unit Testing

Drobnițchi Daniel – Nicușor

Cuprins

1. Introducerea funcției

Pentru proiectul de testare unitară în python am decis să folosesc următoarele tool-uri :

- Pytest
- Coverage for pytest
- Mutmut – Mutation Testing

Funcția testată preia 4 argumente :

- N – Număr de caractere ale unui șir. [Int, 1-20]
- Nr – Număr de la tastatură [Int, >0]
- Np – Număr de la tastatură [Int, >0]
- Sir – Sir de N caractere.

De fiecare când o data este introdusă împotriva specificațiilor, funcția cere un nou input în locul variabilei introduse eronat.

Funcția preia cele două numere, și, în funcție de două criterii, modifică sau nu șirul:

1. Numerele Nr si Np sunt coprime.
2. Nr este palindrom.

Atunci, dacă una dintre acestea este satisfăcută :

Primele [x | x este nr. de cifre din nr1] litere din sir sunt înlocuite cu litere corespunzătoare cifrelor din Nr. [E.g. : Nr = 0010 și sir = ‘Animal’, sir devine ‘aabaal’.

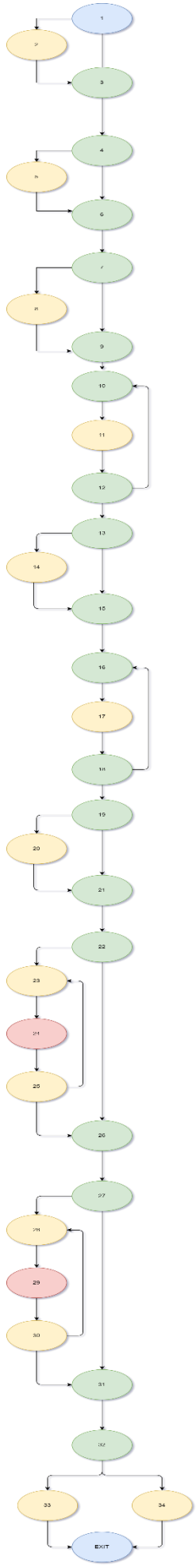
Dacă ambele condiții sunt satisfăcute :

Primele [x | x este nr. de cifre din nr1] litere din sir sunt înlocuite cu litere corespunzătoare cifrelor din Nr + 10. [E.g. : Nr = 0010 și sir = ‘Animal’, sir devine ‘kklkal’.

Dacă niciuna din cele două condiții nu este satisfăcută, atunci șirul nu este schimbat.

Mai jos este prezentat codul sursă și graful funcției.

```python
def simple(n, nr1, nr2, sir):
    string1 = "abcdefghij", string2 = "klmnopqrst"
    sirRetinut = sir

    if n <= 0 or n > 20:
        print("Intrudouceti valoare corecta ")
        x = int(input())
        return simple(x, nr1, nr2, sir)
    if nr1 <= 0:
        print("Intrudouceti valoare corecta ")
        y = int(input())
        return simple(n, y, nr2, sir)
    if nr2 <= 0:
        print("Intrudouceti valoare corecta ")
        z = int(input())
        return simple(n, nr1, z, sir)

    clonaNr1 = nr1,clonaNr2 = nr2,  count = 0

    while clonaNr2 != 0:
        clonaNr1, clonaNr2 = clonaNr2, clonaNr1 % clonaNr2 #Nr Prime

    if clonaNr1 == 1:
        count += 1

    nr1Flipped = 0
    clona2Nr1 = nr1
    while (clona2Nr1 > 0):
        temp = clona2Nr1 % 10
        nr1Flipped = nr1Flipped * 10 + temp
        clona2Nr1 = clona2Nr1 // 10

    if (nr1Flipped == nr1):
        count += 1

    if (count == 1):
        nr1 = str(nr1)
        for i in range(len(nr1)):
            x = nr1[i]
            sir = sir[:i % n] + string1[int(x)] + sir[i % n + 1:]
    if count == 2:
        nr1 = str(nr1)
        for i in range(len(nr1)):
            x = nr1[i]
            sir = sir[:i % n] + string2[int(x)] + sir[i % n + 1:]

    if sirRetinut == sir:
        return True
    else:
        return False
```



La finalul funcției, dacă șirul este schimbat, funcția returnează ‘False’, dacă acesta rămâne la fel, aceasta returnează ‘True’.

Precizări cu privire la funcțiile de mutare și eventuale erori :

Pentru acest test, am rezumat la folosirea librăriei mutmut pentru generarea de mutanți, dar nu am putut să măsor eficiența acestora, deoarece funcția aplica doar mutațiile, dar nu efectua și testele prezentate în folderul pentru test.

Am încercat să implementez și librăria MutPy, însă după lupte îndelungate și multe versiuni diferite de python, pycharm, anaconda și schimbări de kernel-uri și interpretatoare, nu am reușit să fac scriptul să ruleze pe niciun calculator pe care îl dețin.

Suita de teste pe care am implementat-o pe această funcție simplă urmărește două strategii de BlackBox testing și două strategii de WhiteBox Testing.

2. Blackbox testing

A. Equivalence Partitioning:

După cum am prezentat, avem 4 date de intrare și una de ieșire.

Una din date de intrare este determinată direct de cifra n (sir).

Pentru n:

1. n_1 : n | 0<n<20
2. n_2: n | n<1
3. n_3: n | n>20

Pentru nr:

1. nr : nr | nr<1
2. nr : nr| nr>0

Pentru np:

1. np : np | np<1

2. np : np| np>0

Pentru datele de iesire:

1. False
2. True

Metoda pe care am ales-o se apropie foarte mult ca și eficiență și ca tipul de date pe care-l țintește față de metoda valorilor de frontieră.

Este o metodă naivă, în care, realistic, testăm o cantitate foarte mică din funcția propriu-zisă, dar este rapidă și ușor de implementat.

Obținem următoarele date de testat:

1. c_1111 = (3, 215, 5, 'abc')
2. c_1112 = (3, 20, 2, 'abc')
3. c_1121 = (3, 215, -8, ' ')
4. c_1122 = (3, 215, -8 , ' ')
5. c_1211 = (3, -80, ' ', ' ')
6. c_1212 = (3, -80, ' ' , ' ')
7. c_2111 = (-5, ' ' , ' ' ,' ')
8. c_...

```
def test_partitioning(self):
     self.assertEqual(simple(3,20, 2, "superanimal"),  True)
     self.assertEqual(simple(3,215, 18, "superanimal"),  False)
     self.assertEqual(simple(3,215, -8, "superanimal"),  True)
     self.assertEqual(simple(3,215, -8, "superanimal"),  True)
     self.assertEqual(simple(3,-80, 8, "superanimal"),  True)
     self.assertEqual(simple(3,-80, -8, "superanimal"),  True)
     self.assertEqual(simple(0,-20, 2, "superanimal"),  True)
     self.assertEqual(simple(0, 215, 18, "superanimal"), False)
     self.assertEqual(simple(0, 215, -8, "superanimal"), True)
     self.assertEqual(simple(0, 215, -8, "superanimal"), True)
     self.assertEqual(simple(0, -80, 8, "superanimal"), True)
     self.assertEqual(simple(0, -80, -8, "superanimal"), True)
     self.assertEqual(simple(21, -20, 2, "superanimal"), True)
     self.assertEqual(simple(21, 215, 18, "superanimal"), False)
     self.assertEqual(simple(21, 215, -8, "superanimal"), True)
     self.assertEqual(simple(21, 215, -8, "superanimal"), True)
     self.assertEqual(simple(21, -80, 8, "superanimal"), True)
     self.assertEqual(simple(21, -80, -8, "superanimal"), True)
```

B. Category partitioning

Pentru această metodă, impărțim funcția în unități care sunt formate din variabile, iar pe acestea le împărțim pe categorii.

Pentru n :

1. N<0
2. N=0
3. N=1
4. N=2…19
5. N=20
6. N=21

Pentru nr :

1. Nr<0
2. Nr=0
3. Nr = Palindrom mai mare decat n
4. Nr1= Palindrom mai mic decat n
5. Nr = Numar care nu este palindrom

Pentru np:

1. Np<0
2. Np =0
3. Np = Coprim cu nr1 , mai mic
4. Np = Coprim cu nr1, mai mare

Pentru sir (Conform N).

Pentru Output :

1. True
2. False

In total avem 240 de teste, însă putem să eliminăm cazurile care nu au sens:

1. Valorile aflate în afara valorilor de frontieră (care au fost testate precedent)

2. Cazurile care nu pot exista: palindrom mai mic decât n, mai mare decat n.
3. Cazurile în care output-ul nu poate fi atins.

Obținem următoarele valori pentru fiecare variabilă :

False :

1. n3xnr3xnp3xsir1xo1 2. n3xnr3xnp4xsir1xo1 3. n3xnr3xnp5xsir1xo1 4. n4xnr3xnp3xsir2xo1 5. n4xnr3xnp4xsir2xo1 6. n4xnr3xnp5xsir2xo1 7. n5xnr3xnp3xsir3xo1 8. n5xnr3xnp4xsir3xo1 9. n5xnr3xnp5xsir3xo1 10. n4xnr4xnp3xsir2xo1 11. n4xnr4xnp4xsir2xo1 12. n4xnr4xnp5xsir2xo1 13. n5xnr4xnp3xsir3xo1 14. n5xnr4xnp4xsir3xo1 15. n5xnr4xnp5xsir3xo1 16. n4xnr5xnp3xsir2xo1 17. n4xnr5xnp4xsir2xo1 18. n5xnr5xnp3xsir3xo1 19. n5xnr5xnp4xsir3xo1	1. n3xnr3xnp3xsir1xo2 2. n3xnr3xnp4xsir1xo2 3. n3xnr3xnp5xsir1xo2 4. n4xnr3xnp3xsir2xo2 5. n4xnr3xnp4xsir2xo2 6. n4xnr3xnp5xsir2xo2 7. n5xnr3xnp3xsir3xo2 8. n5xnr3xnp4xsir3xo2 9. n5xnr3xnp5xsir3xo2 10. n4xnr4xnp3xsir2xo2 11. n4xnr4xnp4xsir2xo2 12. n4xnr4xnp5xsir2xo2 13. n5xnr4xnp3xsir3xo2 14. n5xnr4xnp4xsir3xo2 15. n5xnr4xnp5xsir3xo2 16. n3xnr5xnp3xsir1xo2 17. n3xnr5xnp4xsir1xo2 18. n4xnr5xnp3xsir2xo2 19. n4xnr5xnp4xsir2xo2 20. n5xnr5xnp3xsir3xo2 21. n5xnr5xnp4xsir3xo2 22. n5xnr5xnp5xsir3xo2

20. n3xnr3xnp3xsir1xo1
21. n3xnr3xnp4xsir1xo1

22. n3xnr3xnp5xsir1xo1
23. n4xnr3xnp3xsir2xo1
24. n4xnr3xnp4xsir2xo1
25. n4xnr3xnp5xsir2xo1
26. n5xnr3xnp3xsir3xo1
27. n5xnr3xnp4xsir3xo1
28. n5xnr3xnp5xsir3xo1
29. n4xnr4xnp3xsir2xo1
30. n4xnr4xnp4xsir2xo1
31. n4xnr4xnp5xsir2xo1
32. n5xnr4xnp3xsir3xo1
33. n5xnr4xnp4xsir3xo1
34. n5xnr4xnp5xsir3xo1
35. n4xnr5xnp3xsir2xo1
36. n4xnr5xnp4xsir2xo1
37. n5xnr5xnp3xsir3xo1
38. n5xnr5xnp4xsir3xo1

TRUE

23. n3xnr3xnp3xsir1xo2
24. n3xnr3xnp4xsir1xo2
25. n3xnr3xnp5xsir1xo2
26. n4xnr3xnp3xsir2xo2
27. n4xnr3xnp4xsir2xo2
28. n4xnr3xnp5xsir2xo2
29. n5xnr3xnp3xsir3xo2
30. n5xnr3xnp4xsir3xo2

31. n5xnr3xnp5xsir3xo2
32. n4xnr4xnp3xsir2xo2
33. n4xnr4xnp4xsir2xo2
34. n4xnr4xnp5xsir2xo2
35. n5xnr4xnp3xsir3xo2
36. n5xnr4xnp4xsir3xo2
37. n5xnr4xnp5xsir3xo2
38. n3xnr5xnp3xsir1xo2
39. n3xnr5xnp4xsir1xo2
40. n4xnr5xnp3xsir2xo2
41. n4xnr5xnp4xsir2xo2
42. n5xnr5xnp3xsir3xo2
43. n5xnr5xnp4xsir3xo2
44. n5xnr5xnp5xsir3xo2