محمد مهدی نظری – 9931061

بخش اول:

فرمت گرفتن ورودی به این شکل است: مثلا با ورودی A=[a1,a2,a3]-B=[b1,b2]-C=[c1] یعنی Characteristics های ما شامل A,B,C میباشد و Abstract blocks شامل A=[a1,a2,a3] و B=[b1,b2]و C=[c1] است.

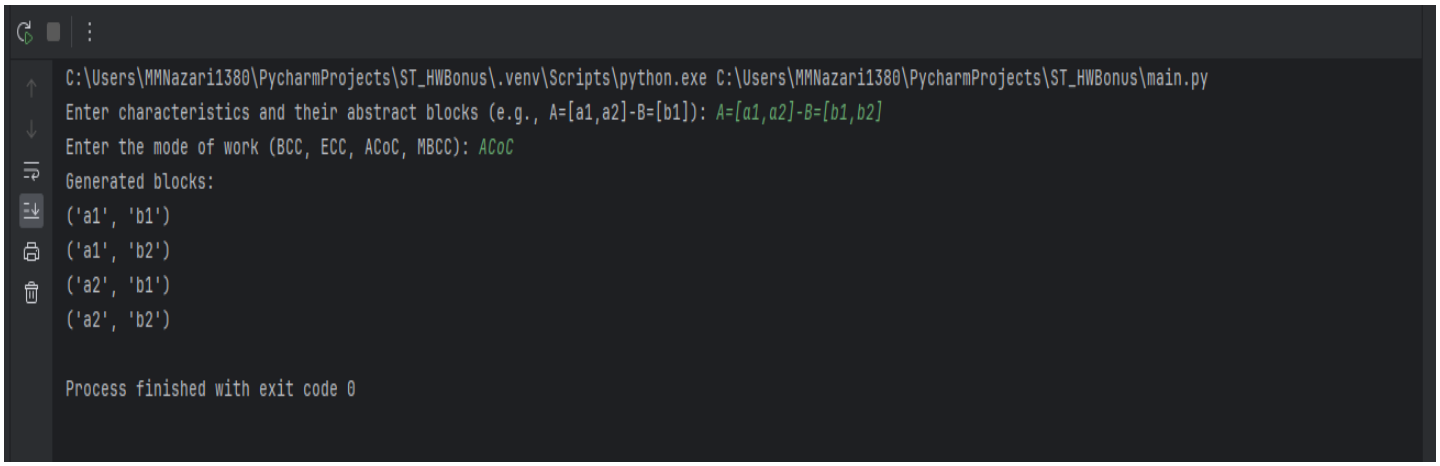
در قدم بعدی انواع مدکاری از کاربر پرسیده میشود که اگر ورودی کاربر BCC یا MBCC باشد در خط بعدی باید Base یا Base ها به فرمت مثلا A=a1,A=a2,B=b1,C=c1 باشد که بدین معنی است که بیس های ما مثلا (a1, b1, c1) و (a2,b1,c1) است.

تست کیس ها :

مود ACoC:

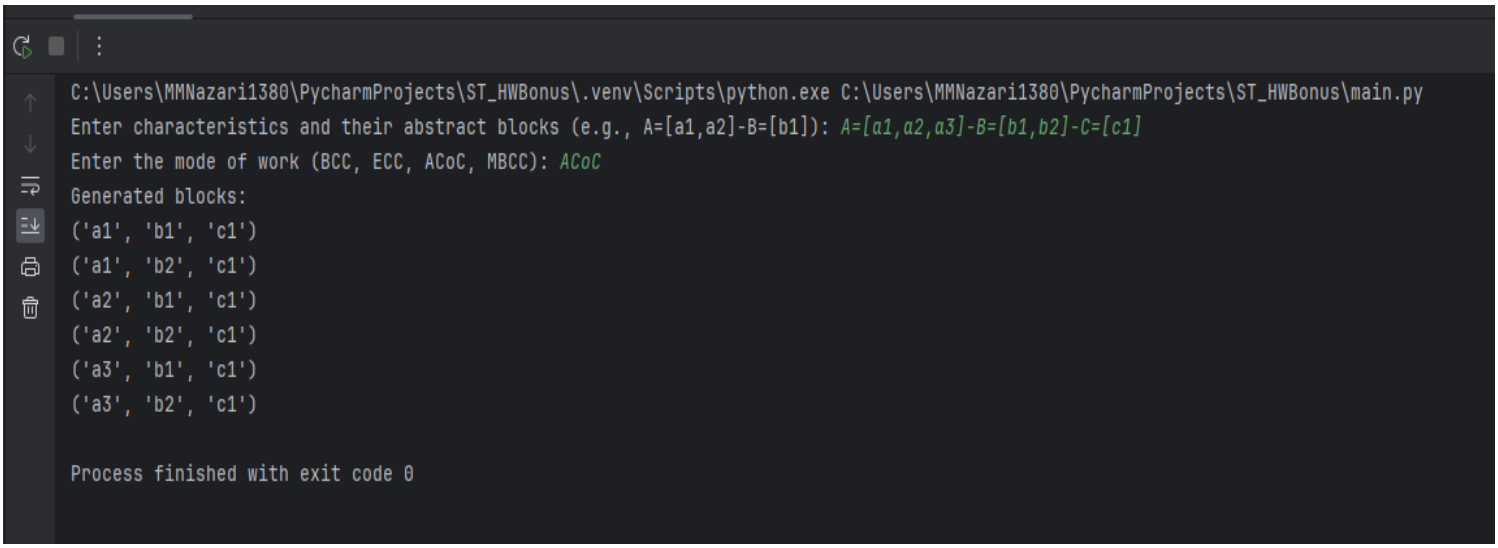
با ورودی : A=[a1,a2]-B=[b1,b2]

خروجی:



با ورودی : A=[a1,a2,a3]-B=[b1,b2]-C=[c1]

خروجی:



با ورودی:

A=[a1,a2,a3,a4,a5]-B=[b1,b2,b3,b4,b5]-C=[c1,c2,c3,c4,c5]- D=[d1,d2,d3,d4,d5]

خروجی:

به علت تعداد زیاد خروجی فقط صفحه اول نمایش داده شده است.

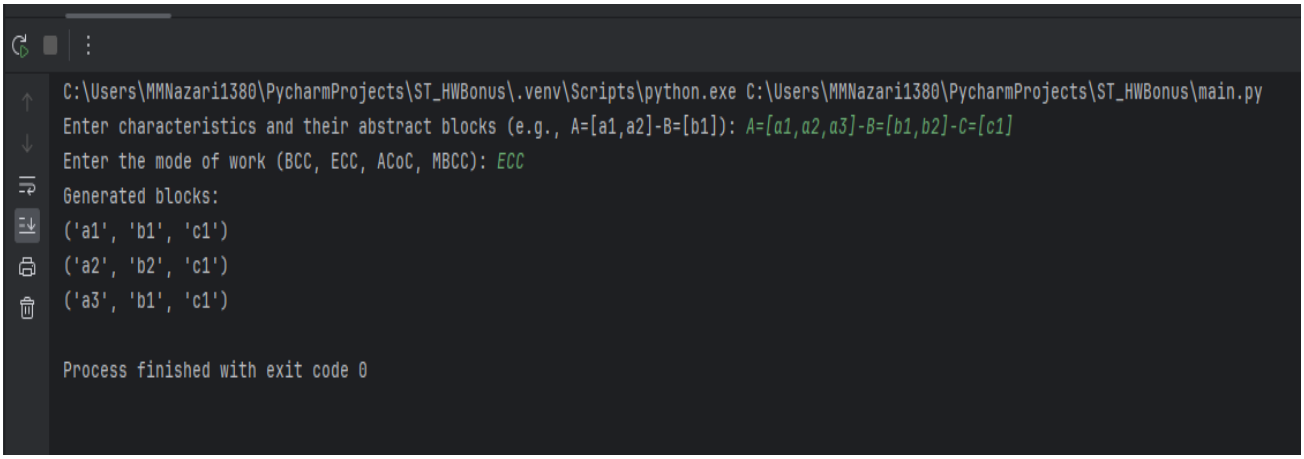
مود ECC:

با ورودی : A=[a1,a2]-B=[b1,b2]

خروجی:

با ورودی :A=[a1,a2,a3]-B=[b1,b2]-C=[c1]

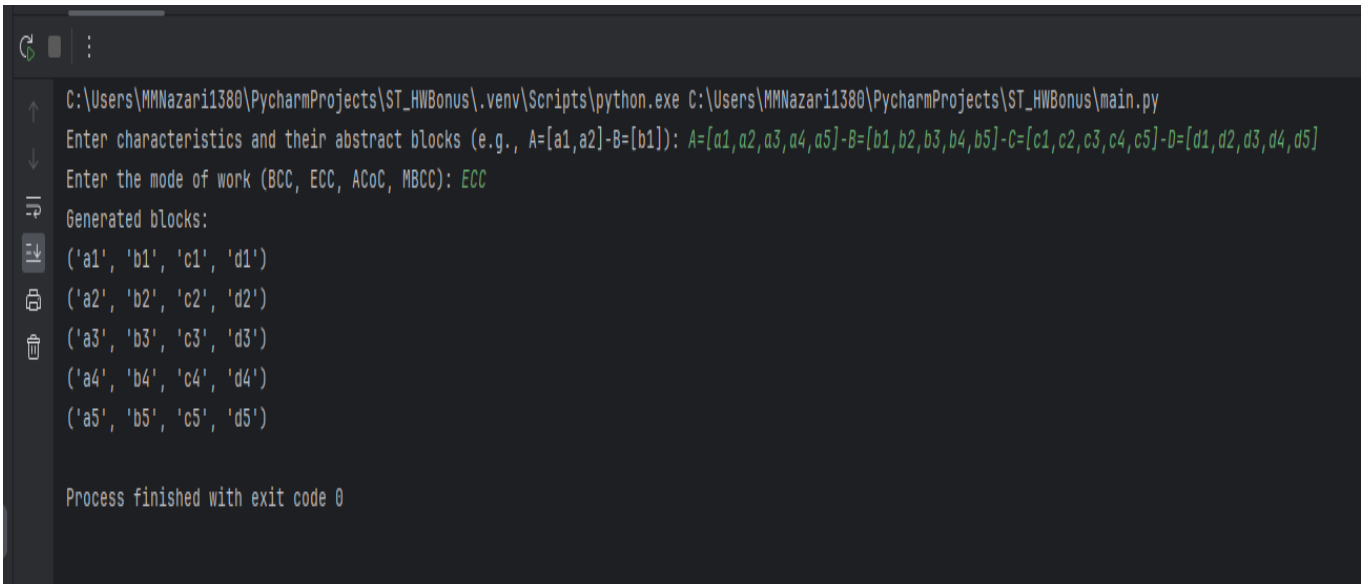
خروجی:



با ورودی:

A=[a1,a2,a3,a4,a5]-B=[b1,b2,b3,b4,b5]-C=[c1,c2,c3,c4,c5]-D=[d1,d2,d3,d4,d5]

خروجی:

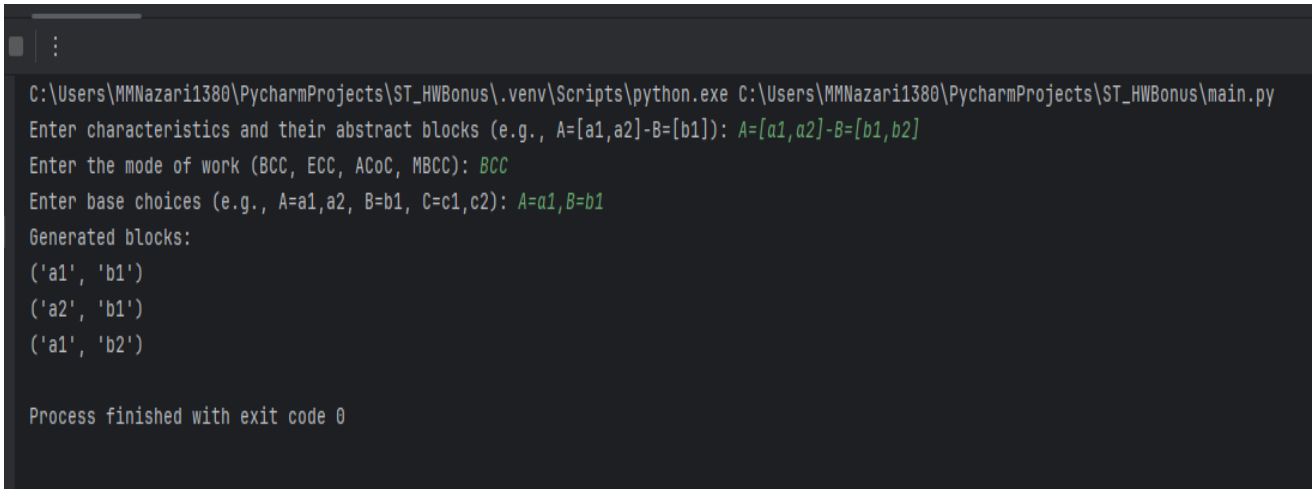


مود BCC:

با ورودی : A=[a1,a2]-B=[b1,b2]

با بیس: A=a1,B=b1

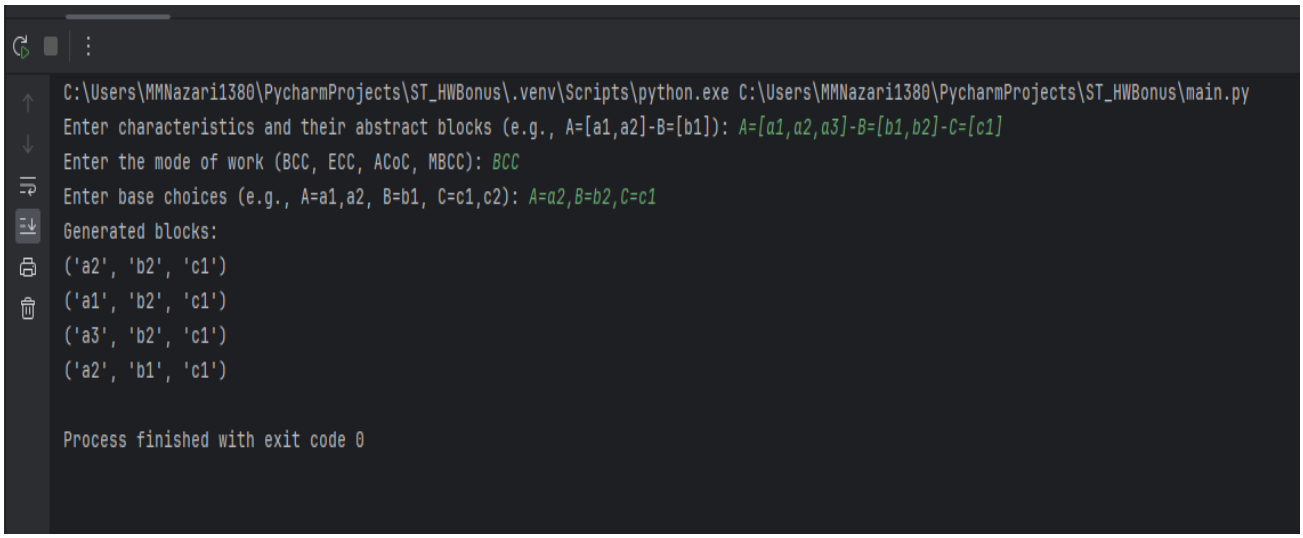
خروجی:



با ورودی :A=[a1,a2,a3]-B=[b1,b2]-C=[c1]

با بیس: A=a2,B=b2,C=c1

خروجی:

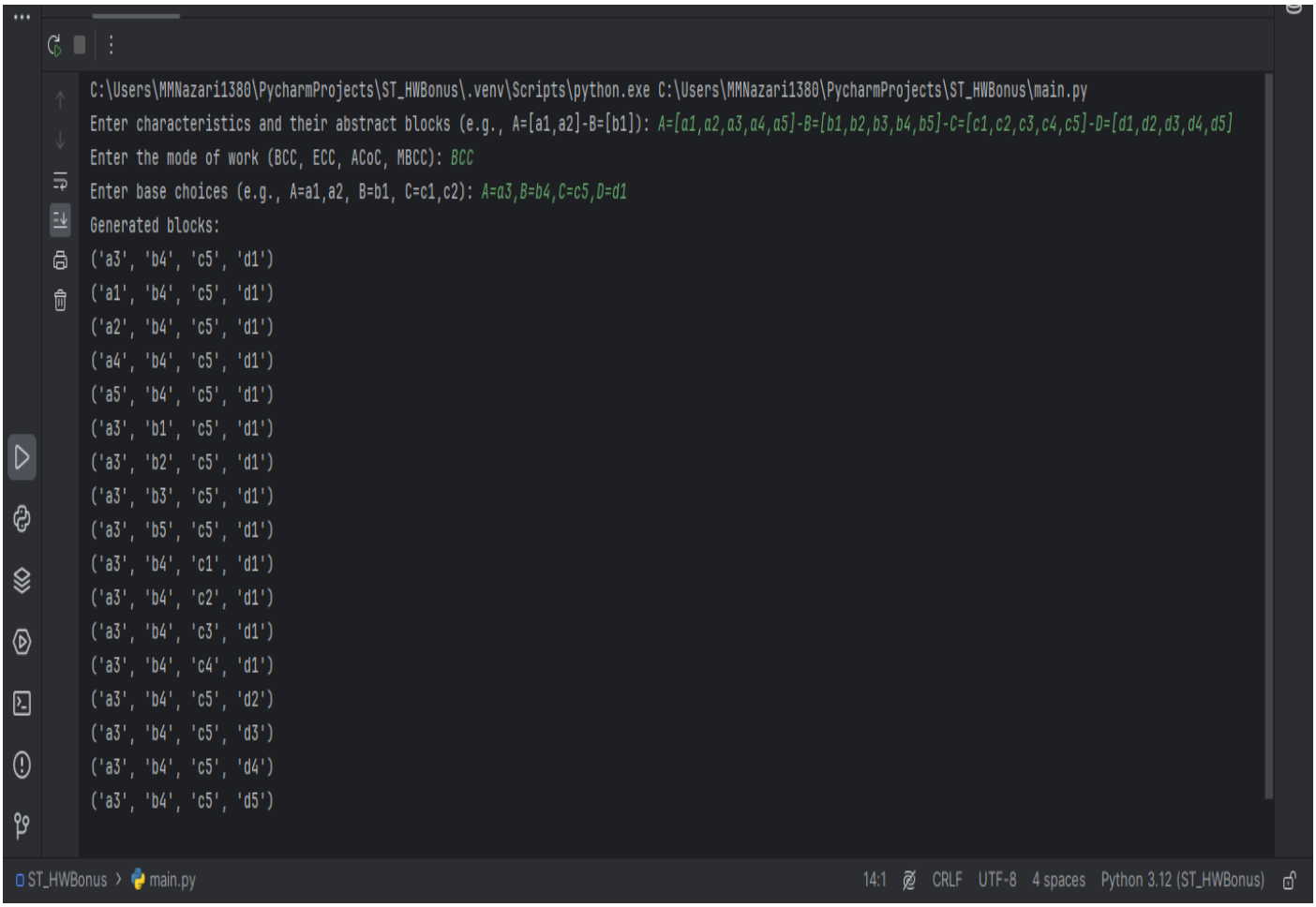


با ورودی:

A=[a1,a2,a3,a4,a5]-B=[b1,b2,b3,b4,b5]-C=[c1,c2,c3,c4,c5]-D=[d1,d2,d3,d4,d5]

با بیس: A=a3,B=b4,C=c5,D=d1

خروجی:

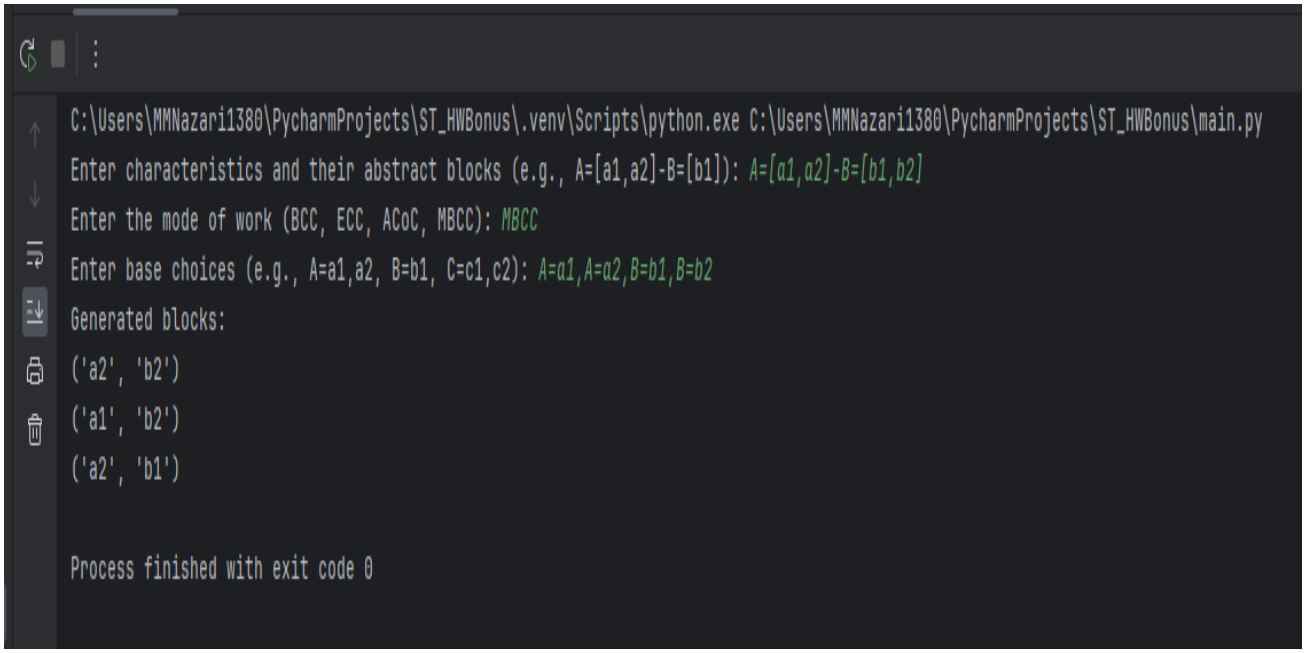


مود MBCC:

با ورودی : A=[a1,a2]-B=[b1,b2]

با بیس های: A=a1,A=a2,B=b1,B=b2

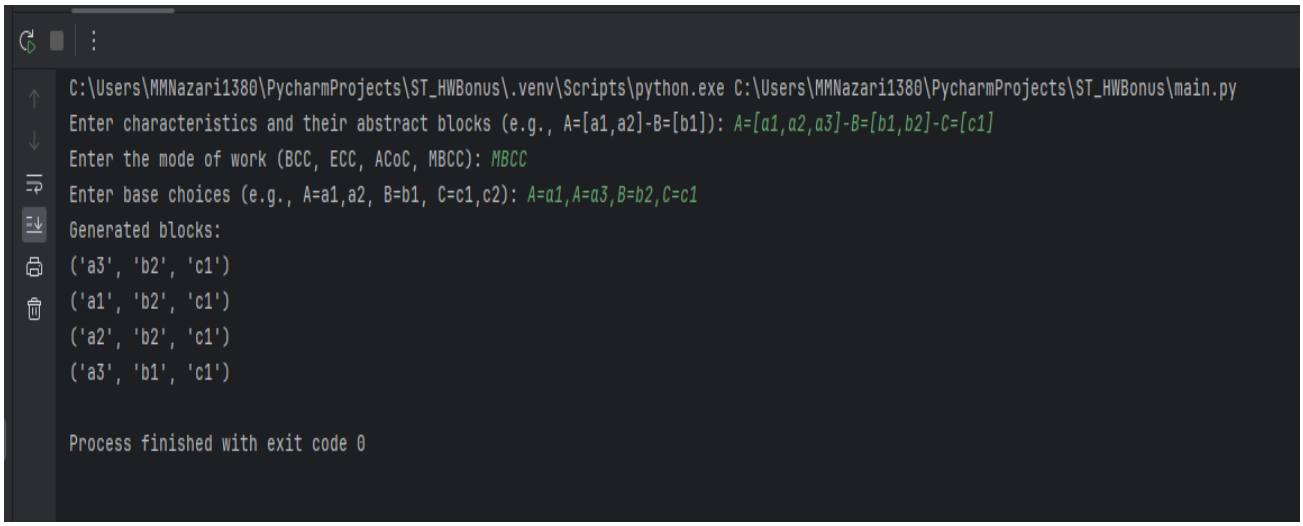
خروجی:



با ورودی : A=[a1,a2,a3]-B=[b1,b2]-C=[c1]

با بیس های: A=a1,A=a3,B=b2,C=c1

خروجی:

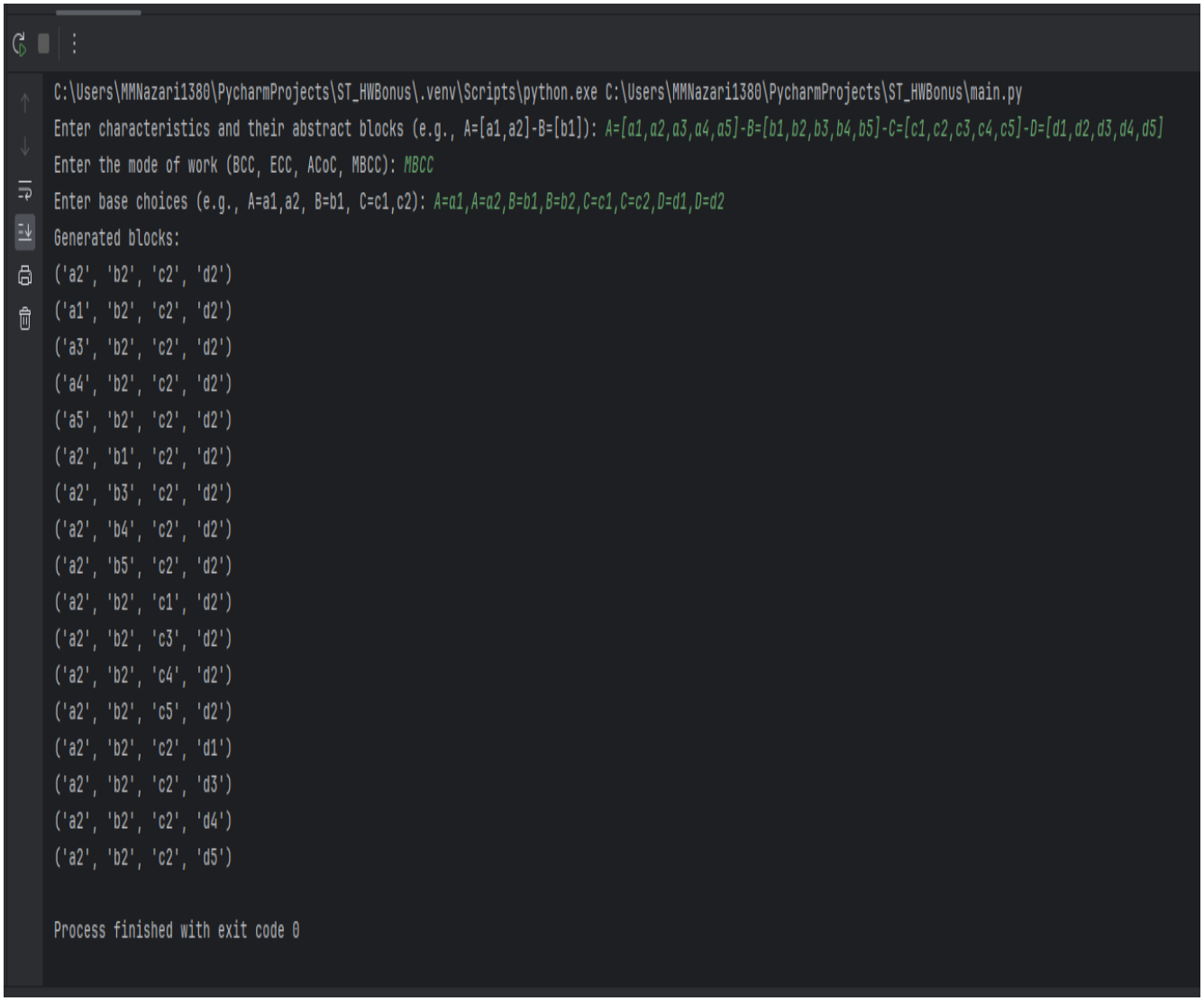


با ورودی:

A=[a1,a2,a3,a4,a5]-B=[b1,b2,b3,b4,b5]-C=[c1,c2,c3,c4,c5]-D=[d1,d2,d3,d4,d5]

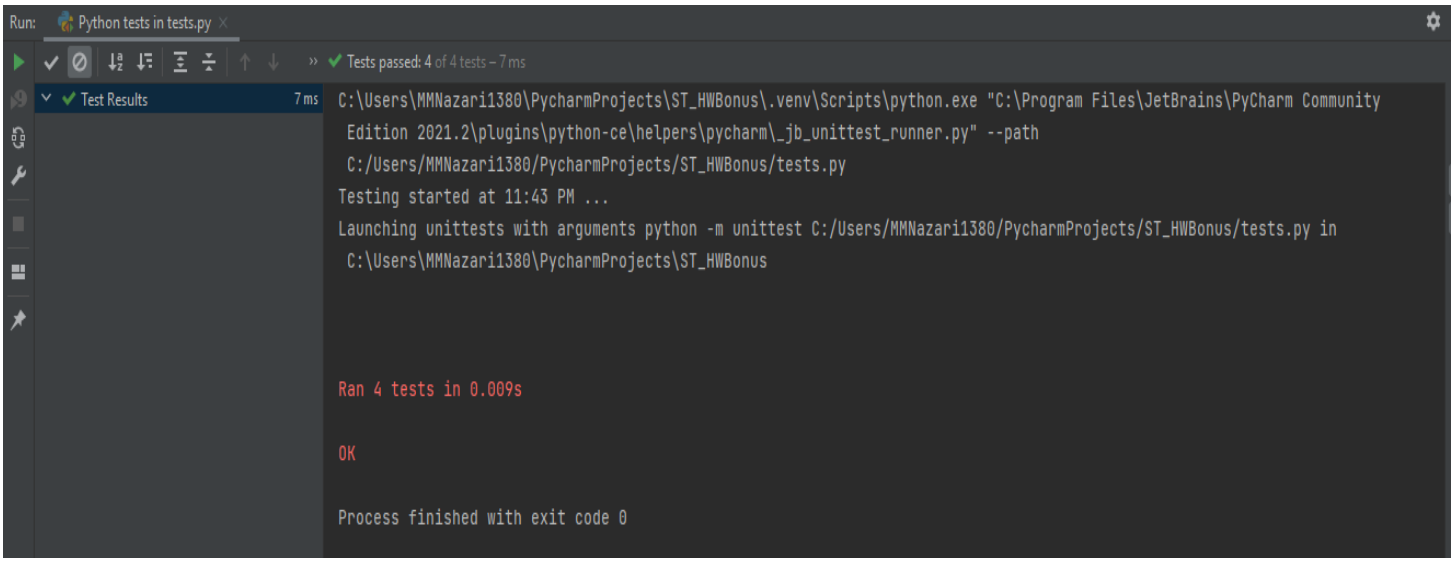
با بیس های: A=a1,A=a2,B=b1,B=b2,C=c1,C=c2,D=d1,D=d2

خروجی:



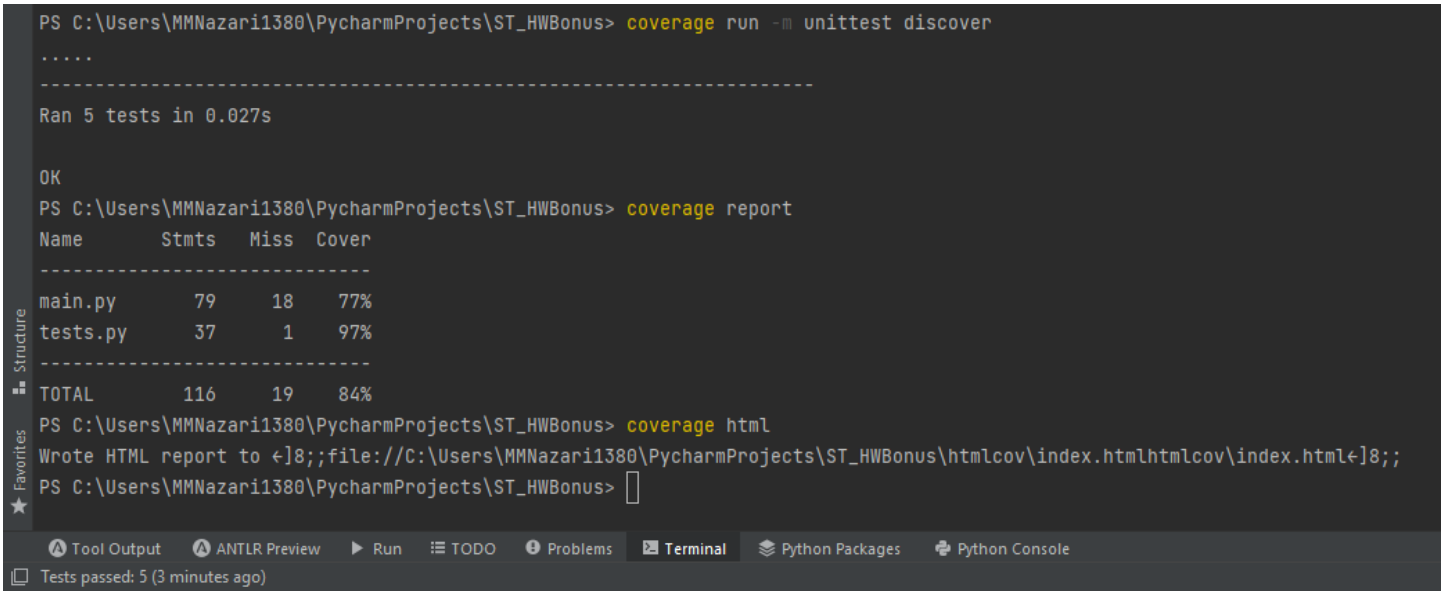
بخش دوم

برای این قسمت برای هر کدام از 4 مود کاری یک تست واحد با اولین حالت ورودی نوشته شده که در فایل tests.py قرار دارند.



یک تست واحد هم برای تابع main مینویسیم تا از جهت میزان پوشش کد به مشکل نخوریم.

برای بدست آوردن میزان پوشش کد از کتابخانه coverage استفاده میکنیم به شرح زیر :



طبق گزارش بدست آمده 77 درصد کد برنامه اصلی توسط تست واحدها پوشش داده شده اند. جزییات پوشش کد در فایل html ساخته شده قابل مشاهده است.

بخش سوم

کد قسمت های feature و steps در دایرکتوری های هرکدام پیاده شده است. یک تست برای حالت AcoC و ورودی A=[a1,a2]-B=[b1,b2] تست شده است با خروجی زیر :

```
PS C:\Users\MMNazari1380\PycharmProjects\ST_HWBonus\features> behave BDD.feature
Feature: Generate Blocks # BDD.feature:1

  Scenario: ACoC mode with given characteristics    # BDD.feature:3
    Given characteristics are "A=[a1,a2]-B=[b1,b2]" # steps/BDD_Steps.py:159
    And the mode is "ACoC"                          # steps/BDD_Steps.py:169
    Then generated blocks should be                 # steps/BDD_Steps.py:174
      |    |    |
      | a1 | b1 |
      | a1 | b2 |
      | a2 | b1 |
      | a2 | b2 |

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s
PS C:\Users\MMNazari1380\PycharmProjects\ST_HWBonus\features>
```