Installation Python and Jupyter Notebook

sudo apt update

sudo apt install python3-pip python3-dev

sudo -H pip3 install --upgrade pip

sudo -H pip3 install virtualenv

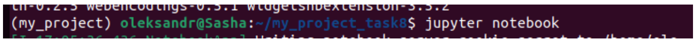
mkdir ~/my_project

cd ~/my_project

virtualenv my_project

source my_project/bin/activate

Your command line should change to indicate the name of the Python virtual environment you are currently running in. It will look like this:



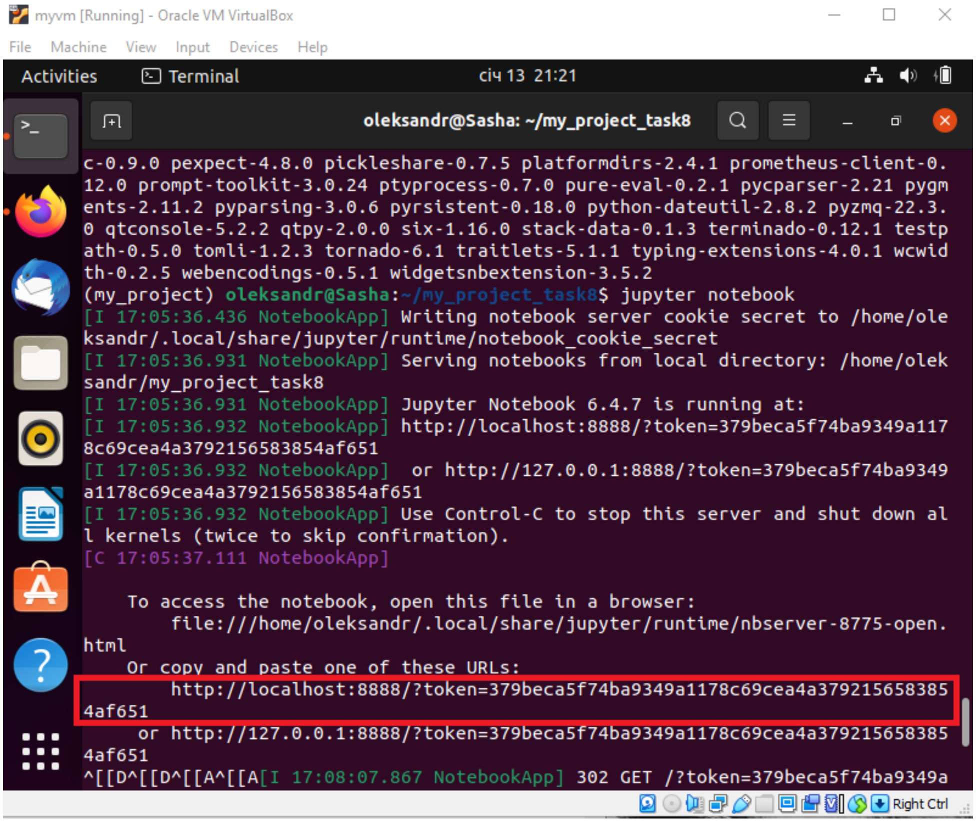
```
(my_project) oleksandr@Sasha:~/my_project_task8$ jupyter notebook
```

Installation Jupyter:

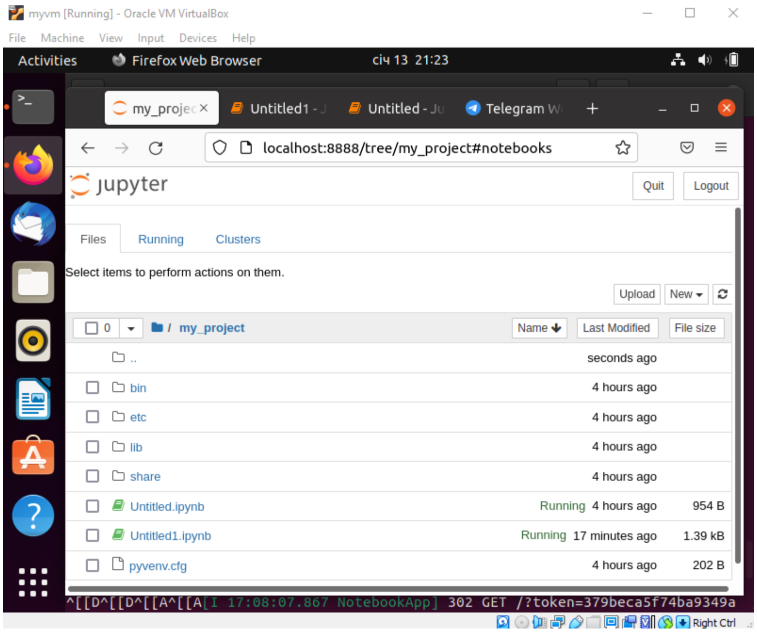
pip install jupyter

jupyter notebook

Go to the displayed URL and connect to Jupyter Notebook


myvm [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
ciч 13 21:21
oleksandr@Sasha: ~/my_project_task8
Right Ctrl

```
c-0.9.0 pexpect-4.8.0 pickleshare-0.7.5 platformdirs-2.4.1 prometheus-client-0.
12.0 prompt-toolkit-3.0.24 ptyprocess-0.7.0 pure-eval-0.2.1 pycparser-2.21 pygm
ents-2.11.2 pyparsing-3.0.6 pyrsistent-0.18.0 python-dateutil-2.8.2 pyzmq-22.3.
0 qtconsole-5.2.2 qtpy-2.0.0 six-1.16.0 stack-data-0.1.3 terminado-0.12.1 testp
ath-0.5.0 tomli-1.2.3 tornado-6.1 traitlets-5.1.1 typing-extensions-4.0.1 wcwid
th-0.2.5 webencodings-0.5.1 widgetsnbextension-3.5.2
(my_project) oleksandr@Sasha:~/my_project_task8$ jupyter notebook
[I 17:05:36.436 NotebookApp] Writing notebook server cookie secret to /home/ole
ksandr/.local/share/jupyter/runtime/notebook_cookie_secret
[I 17:05:36.931 NotebookApp] Serving notebooks from local directory: /home/olek
sandr/my_project_task8
[I 17:05:36.931 NotebookApp] Jupyter Notebook 6.4.7 is running at:
[I 17:05:36.932 NotebookApp] http://localhost:8888/?token=379beca5f74ba9349a117
8c69cea4a3792156583854af651
[I 17:05:36.932 NotebookApp]  or http://127.0.0.1:8888/?token=379beca5f74ba9349
a1178c69cea4a3792156583854af651
[I 17:05:36.932 NotebookApp] Use Control-C to stop this server and shut down al
l kernels (twice to skip confirmation).
[C 17:05:37.111 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/oleksandr/.local/share/jupyter/runtime/nbserver-8775-open.
html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=379beca5f74ba9349a1178c69cea4a379215658385
4af651
     or http://127.0.0.1:8888/?token=379beca5f74ba9349a1178c69cea4a379215658385
4af651
^[[D^[[D^[[A^[[A[I 17:08:07.867 NotebookApp] 302 GET /?token=379beca5f74ba9349a
```

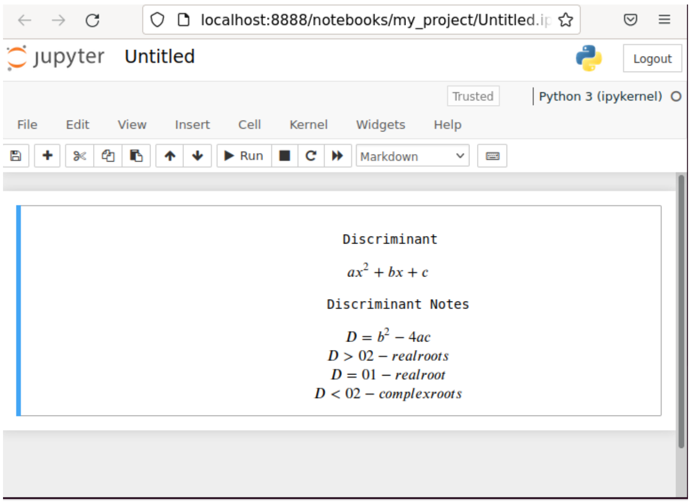


Select items to perform actions on them.

☐ 0 / my_project	Name	Last Modified	File size
..		seconds ago	
☐ bin		4 hours ago	
☐ etc		4 hours ago	
☐ lib		4 hours ago	
☐ share		4 hours ago	
☐ Untitled.ipynb	Running	4 hours ago	954 B
☐ Untitled1.ipynb	Running	17 minutes ago	1.39 kB
☐ pyvenv.cfg		4 hours ago	202 B

```
^[[D^[[D^[[A^[[A[I 17:08:07.867 NotebookApp] 302 GET /?token=379beca5f74ba9349a
```

Let's write the conditions of the problem Markdown



Discriminant

$$ax^2 + bx + c$$

Discriminant Notes

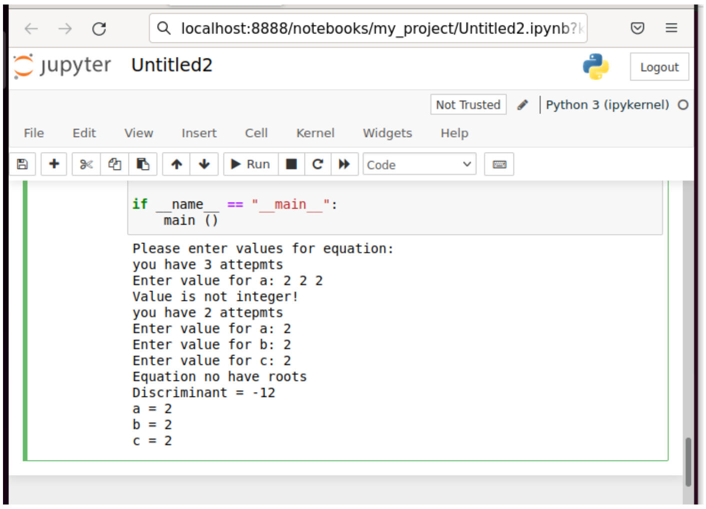
$$D = b^2 - 4ac$$
$$D > 02 - realroots$$
$$D = 01 - realroot$$
$$D < 02 - complexroots$$

Write program code

```
import math


def validate_param():
    attepmts = 3
    while attepmts > 0:
        try:
            print(f'you have {attepmts} attepmts')
            a = int(input("Enter value for a: "))
            b = int(input("Enter value for b: "))
            c = int(input("Enter value for c: "))
        except ValueError:
            print("Value is not integer!")
            attepmts -= 1
            # validate_param(a, b, c)
            continue
        else:
            return a, b, c


def discriminant(a, b, c):
    discr = b ** 2 - 4 * a * c
    return discr


def roots(discr, a, b, c):
    if discr > 0:
        x1 = (-b + math.sqrt(discr)) / (2 * a)
        x2 = (-b - math.sqrt(discr)) / (2 * a)
        print("x1 = %.2f \nx2 = %.2f" % (x1, x2))
        return x1, x2
    elif discr == 0:
        x = -b / (2 * a)
        print("x = %.2f" % x)
        return x
    else:
        print("Equation not have roots")
```

```
def solv_square(a, b, c) -> roots:
    discr = discriminant(a, b, c)
    roots(discr, a, b, c)
    print("Discriminant =", discr)


def square_print(a, b, c, roots):
    print("a =", a)
    print("b =", b)
    print("c =", c)
    roots


def main():
    print("Please enter values for equation:")
    valid_params = validate_param()
    a = valid_params[0]
    b = valid_params[1]
    c = valid_params[2]
    solv_square(a, b, c)
    square_print(a, b, c, roots)

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

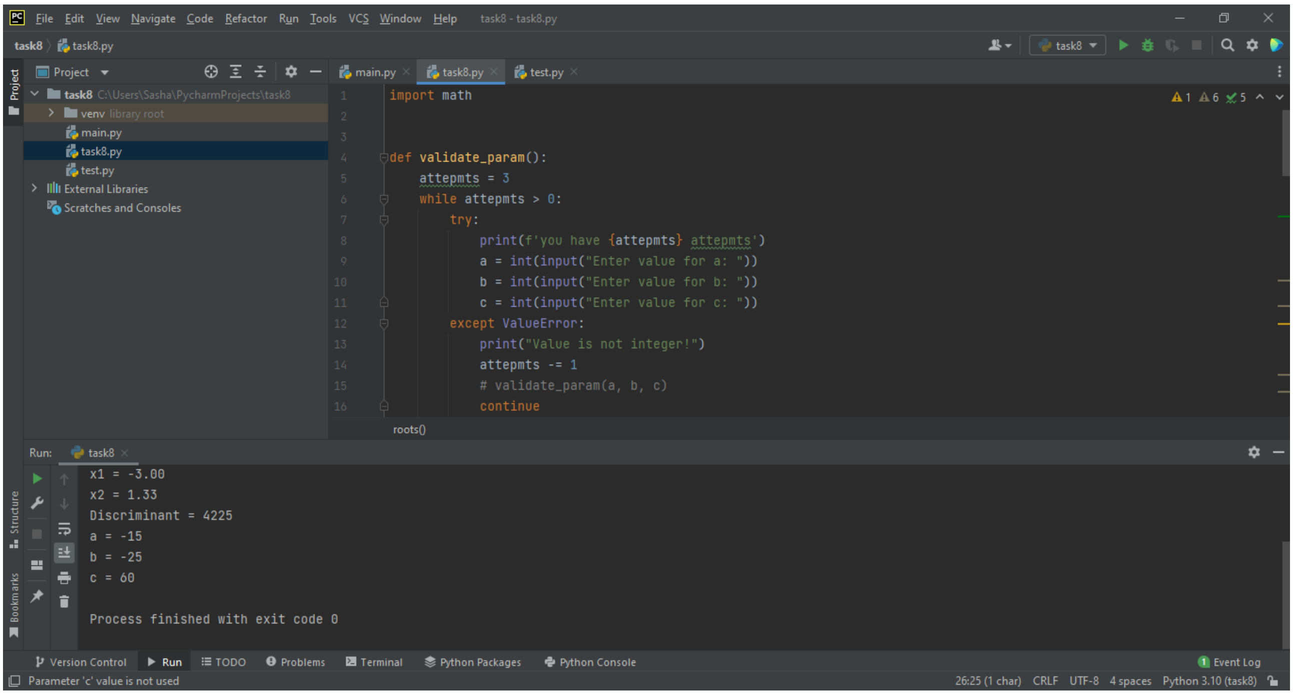
Enter the parameters and see the result



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

```
Please enter values for equation:
you have 3 attepmts
Enter value for a: 2 2 2
Value is not integer!
you have 2 attepmts
Enter value for a: 2
Enter value for b: 2
Enter value for c: 2
Equation no have roots
Discriminant = -12
a = 2
b = 2
c = 2
```

Now let's try on PyCharm



```
import math


def validate_param():
    attepmts = 3
    while attepmts > 0:
        try:
            print(f'you have {attepmts} attepmts')
            a = int(input("Enter value for a: "))
            b = int(input("Enter value for b: "))
            c = int(input("Enter value for c: "))
        except ValueError:
            print("Value is not integer!")
            attepmts -= 1
            # validate_param(a, b, c)
            continue
```

```
x1 = -3.00
x2 = 1.33
Discriminant = 4225
a = -15
b = -25
c = 60

Process finished with exit code 0
```

Code to test

import unittest

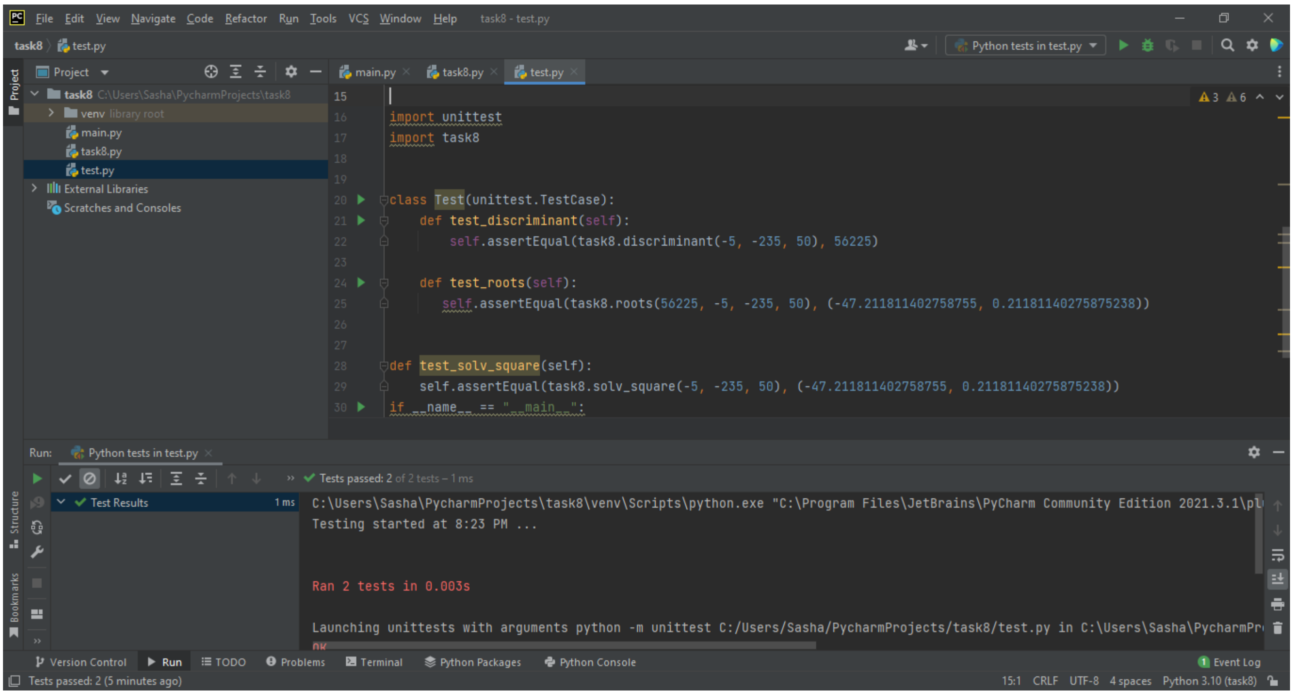
import task8

class Test(unittest.TestCase):

    def test_discriminant(self):

        self.assertEqual(task8.discriminant(-5, -235, 50), 56225)

```
    def test_roots(self):
       self.assertEqual(task8.roots(56225, -5, -235, 50), (-47.211811402758755, 0.21181140275875238))



def test_solv_square(self):
    self.assertEqual(task8.solv_square(-5, -235, 50), (-47.211811402758755, 0.21181140275875238))

import unittest
import task8



class Test(unittest.TestCase):
    def test_discriminant(self):
        self.assertEqual(task8.discriminant(-5, -235, 50), 56225)

    def test_roots(self):
       self.assertEqual(task8.roots(56225, -5, -235, 50), (-47.211811402758755, 0.21181140275875238))



def test_solv_square(self):
    self.assertEqual(task8.solv_square(-5, -235, 50), (-47.211811402758755, 0.21181140275875238))
if __name__ == "__main__":
    unittest.main()
```



```python

import unittest
import task8


class Test(unittest.TestCase):
    def test_discriminant(self):
        self.assertEqual(task8.discriminant(-5, -235, 50), 56225)

    def test_roots(self):
       self.assertEqual(task8.roots(56225, -5, -235, 50), (-47.211811402758755, 0.21181140275875238))


def test_solv_square(self):
    self.assertEqual(task8.solv_square(-5, -235, 50), (-47.211811402758755, 0.21181140275875238))
if __name__ == "__main__":
```

Tests passed: 2 of 2 tests – 1 ms

```
C:\Users\Sasha\PycharmProjects\task8\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2021.3.1\pl
Testing started at 8:23 PM ...


Ran 2 tests in 0.003s

Launching unittests with arguments python -m unittest C:/Users/Sasha/PycharmProjects/task8/test.py in C:\Users\Sasha\PycharmPr
OK
```

Tests passed: 2 (5 minutes ago)