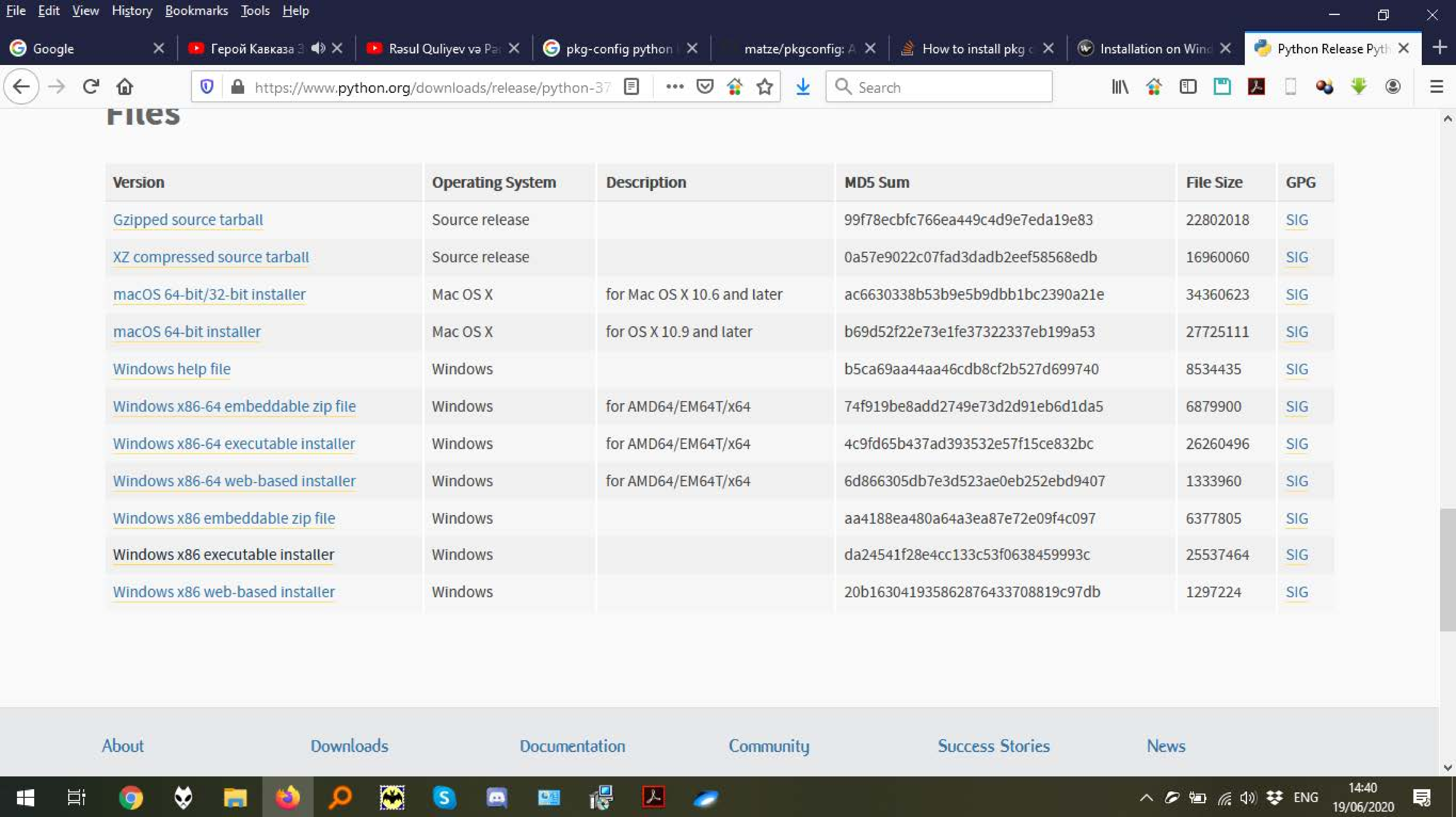
Certification Page

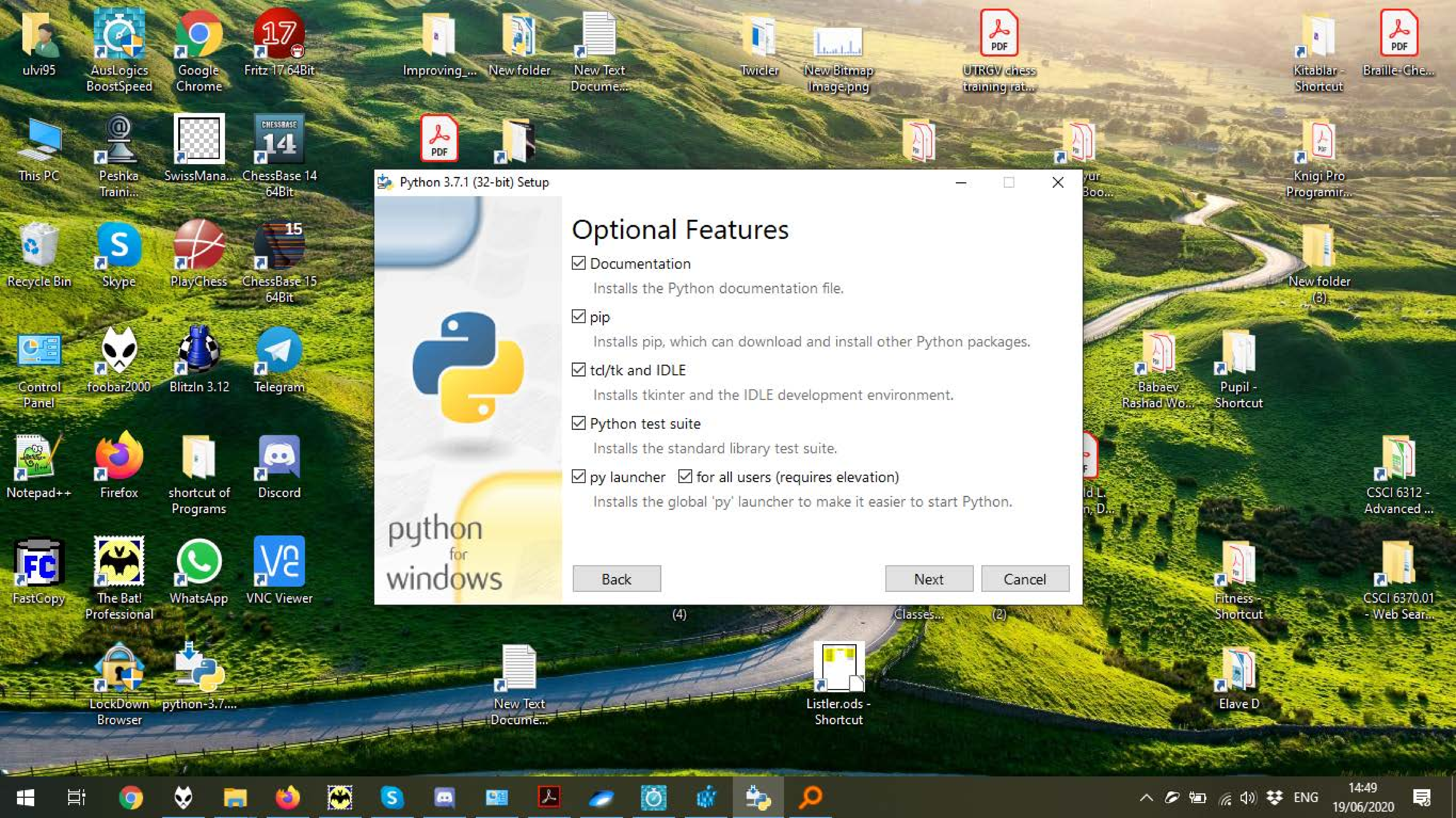
This page must be the first page of your uploaded document.
Your assignment will *not be graded* *without this page (completed with your full name in the area provided) as the first page of your uploaded document.*

I, **Ulvi Bajarani**, certify that the work I am uploading represents my own efforts, and is not copied from anyone else or any other resource (such as Internet). *Furthermore, I certify that I have not let anyone copy from my work.*



Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		99f78ecbfc766ea449c4d9e7eda19e83	22802018	SIG
XZ compressed source tarball	Source release		0a57e9022c07fad3dadb2eef58568edb	16960060	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	ac6630338b53b9e5b9dbb1bc2390a21e	34360623	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	b69d52f22e73e1fe37322337eb199a53	27725111	SIG
Windows help file	Windows		b5ca69aa44aa46cdb8cf2b527d699740	8534435	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	74f919be8add2749e73d2d91eb6d1da5	6879900	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	4c9fd65b437ad393532e57f15ce832bc	26260496	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	6d866305db7e3d523ae0eb252ebd9407	1333960	SIG
Windows x86 embeddable zip file	Windows		aa4188ea480a64a3ea87e72e09f4c097	6377805	SIG
Windows x86 executable installer	Windows		da24541f28e4cc133c53f0638459993c	25537464	SIG
Windows x86 web-based installer	Windows		20b163041935862876433708819c97db	1297224	SIG



Python 3.7.1 (32-bit) Setup

Optional Features

☑ Documentation

Installs the Python documentation file.

☑ pip

Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE

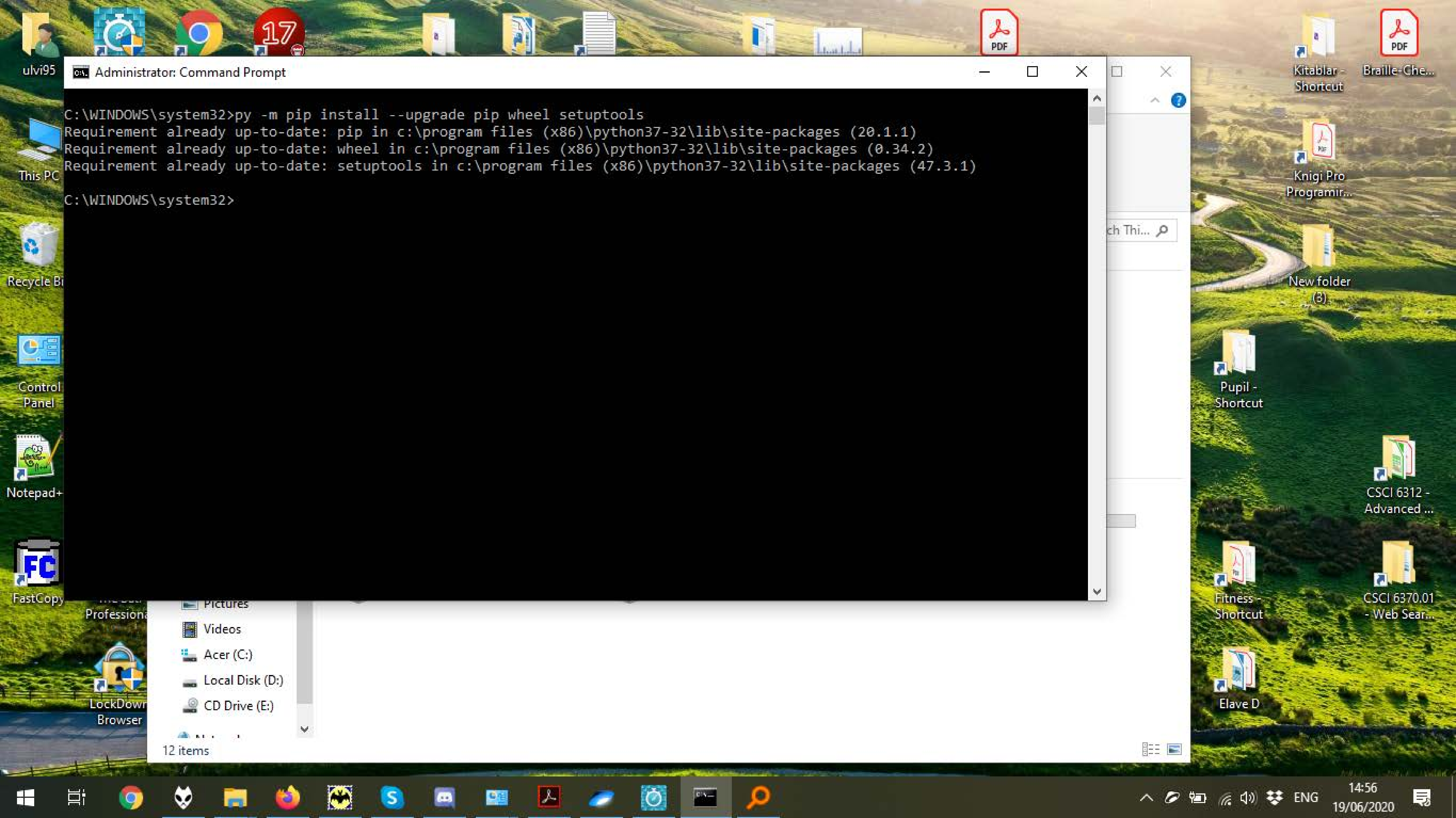
Installs tkinter and the IDLE development environment.

☑ Python test suite

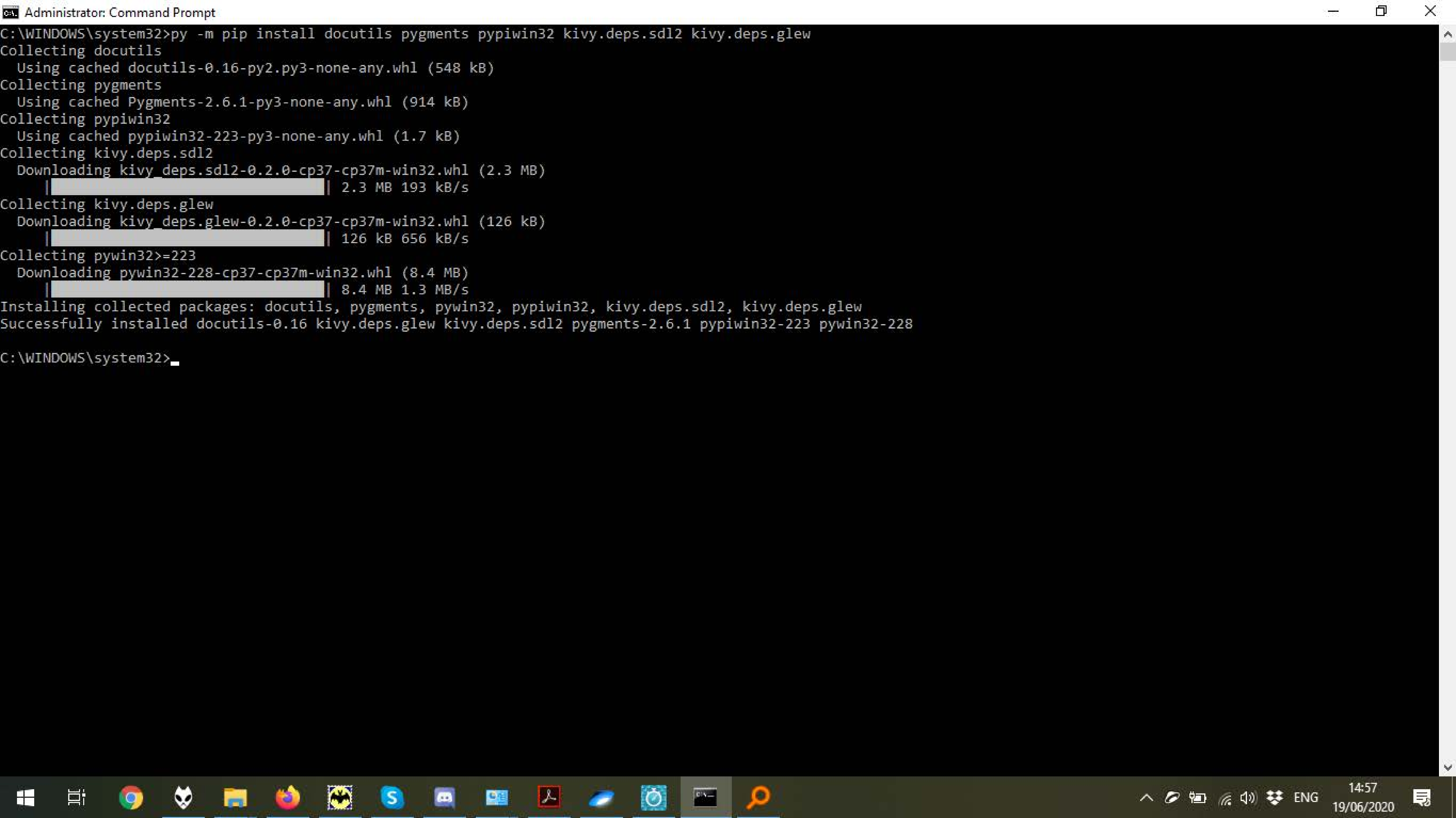
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)

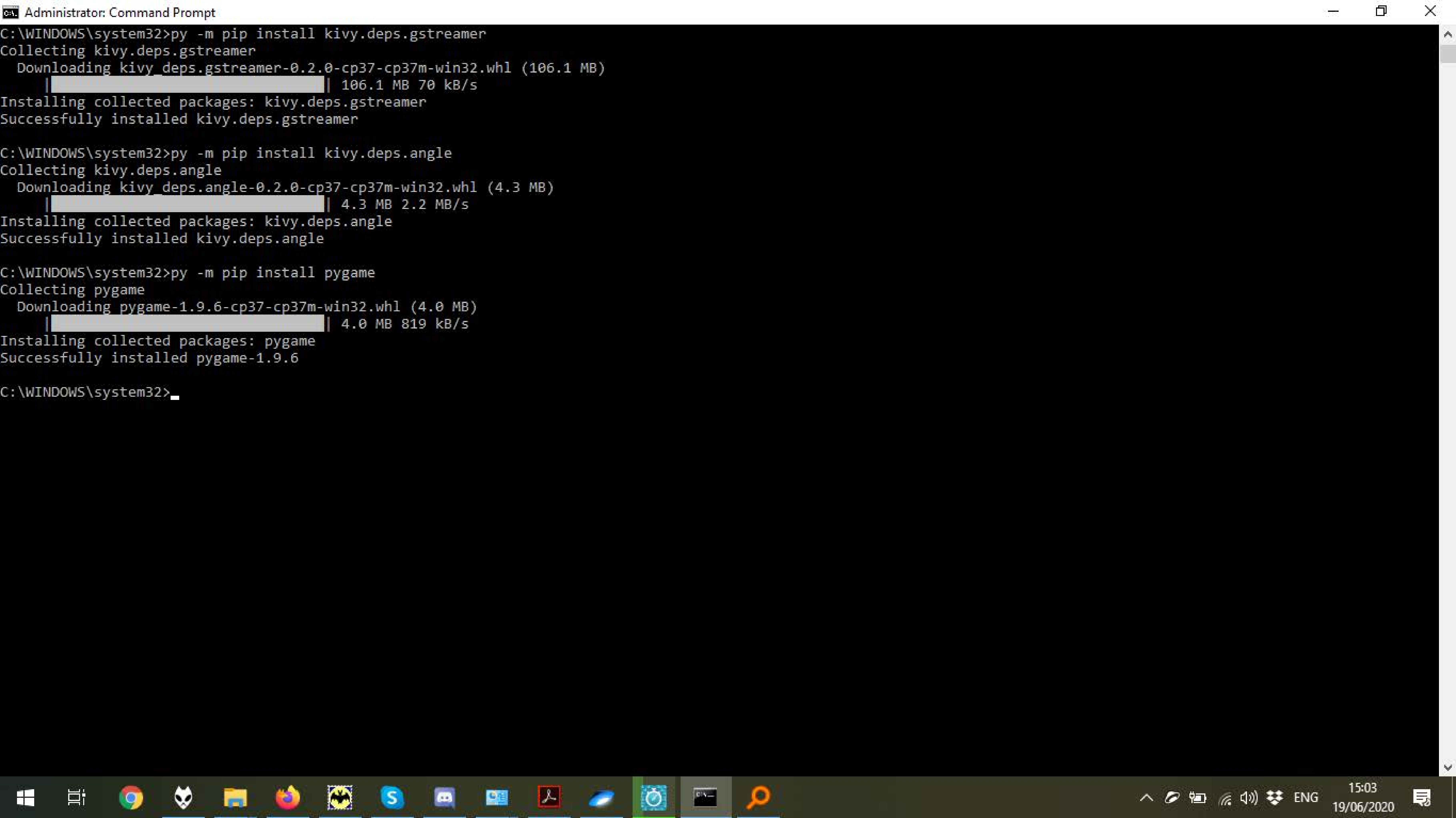
Installs the global 'py' launcher to make it easier to start Python.



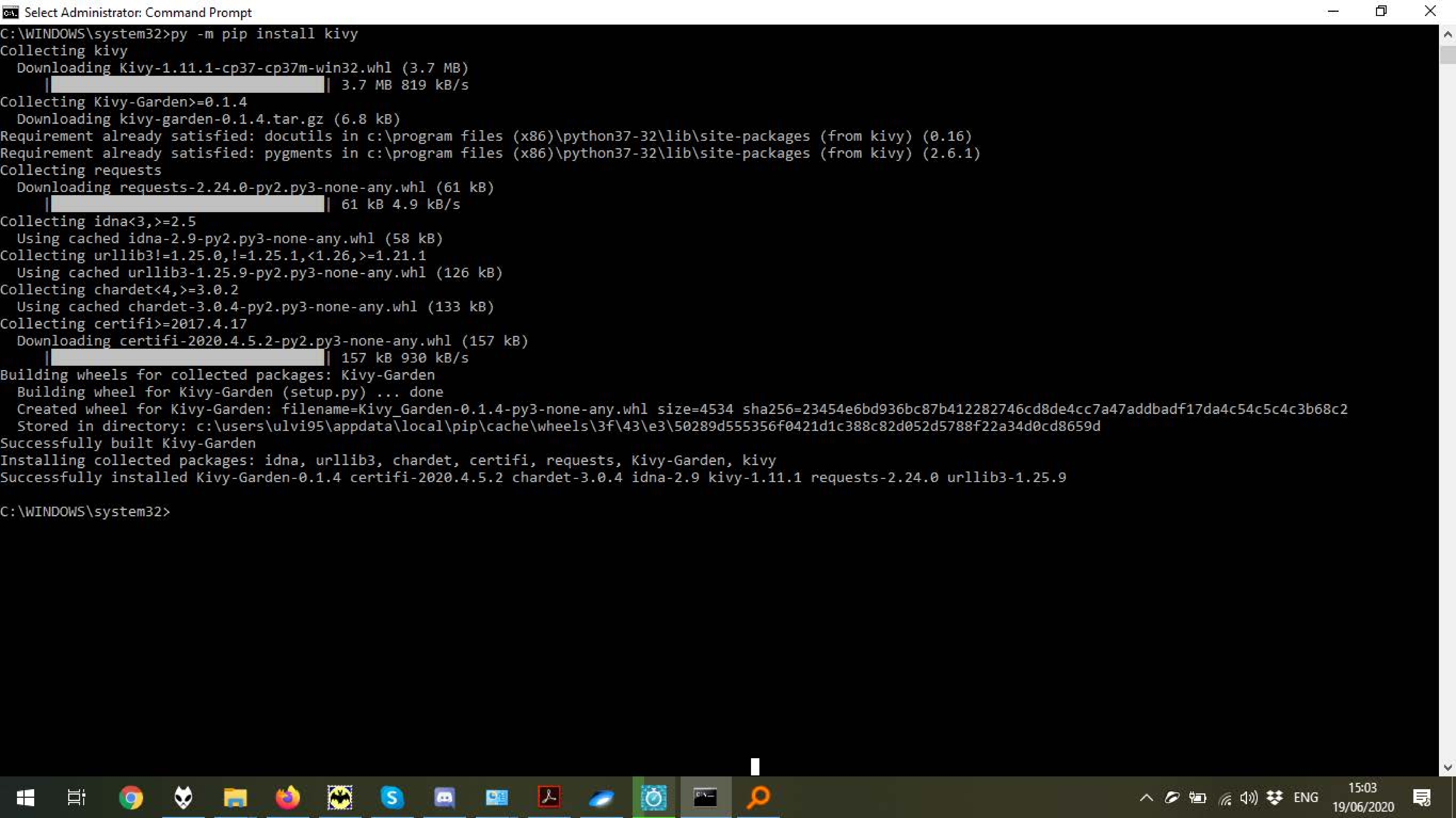
```
C:\WINDOWS\system32>py -m pip install --upgrade pip wheel setuptools
Requirement already up-to-date: pip in c:\program files (x86)\python37-32\lib\site-packages (20.1.1)
Requirement already up-to-date: wheel in c:\program files (x86)\python37-32\lib\site-packages (0.34.2)
Requirement already up-to-date: setuptools in c:\program files (x86)\python37-32\lib\site-packages (47.3.1)

C:\WINDOWS\system32>
```



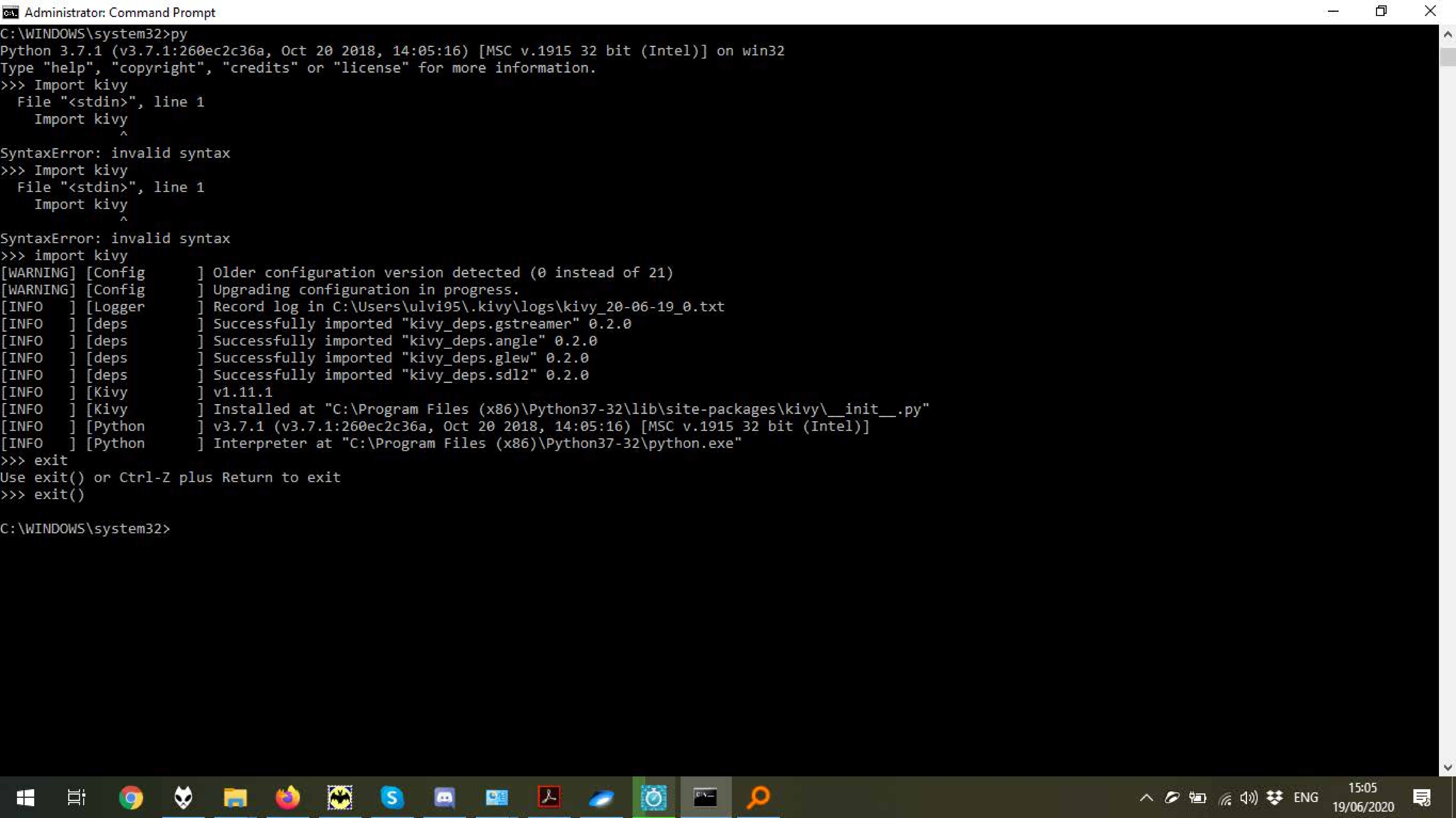
```
C:\WINDOWS\system32>py -m pip install docutils pygments pypiwin32 kivy.deps.sdl2 kivy.deps.glew
Collecting docutils
  Using cached docutils-0.16-py2.py3-none-any.whl (548 kB)
Collecting pygments
  Using cached Pygments-2.6.1-py3-none-any.whl (914 kB)
Collecting pypiwin32
  Using cached pypiwin32-223-py3-none-any.whl (1.7 kB)
Collecting kivy.deps.sdl2
  Downloading kivy_deps.sdl2-0.2.0-cp37-cp37m-win32.whl (2.3 MB)
     |████████████████████████████████| 2.3 MB 193 kB/s
Collecting kivy.deps.glew
  Downloading kivy_deps.glew-0.2.0-cp37-cp37m-win32.whl (126 kB)
     |████████████████████████████████| 126 kB 656 kB/s
Collecting pywin32>=223
  Downloading pywin32-228-cp37-cp37m-win32.whl (8.4 MB)
     |████████████████████████████████| 8.4 MB 1.3 MB/s
Installing collected packages: docutils, pygments, pywin32, pypiwin32, kivy.deps.sdl2, kivy.deps.glew
Successfully installed docutils-0.16 kivy.deps.glew kivy.deps.sdl2 pygments-2.6.1 pypiwin32-223 pywin32-228

C:\WINDOWS\system32>
```



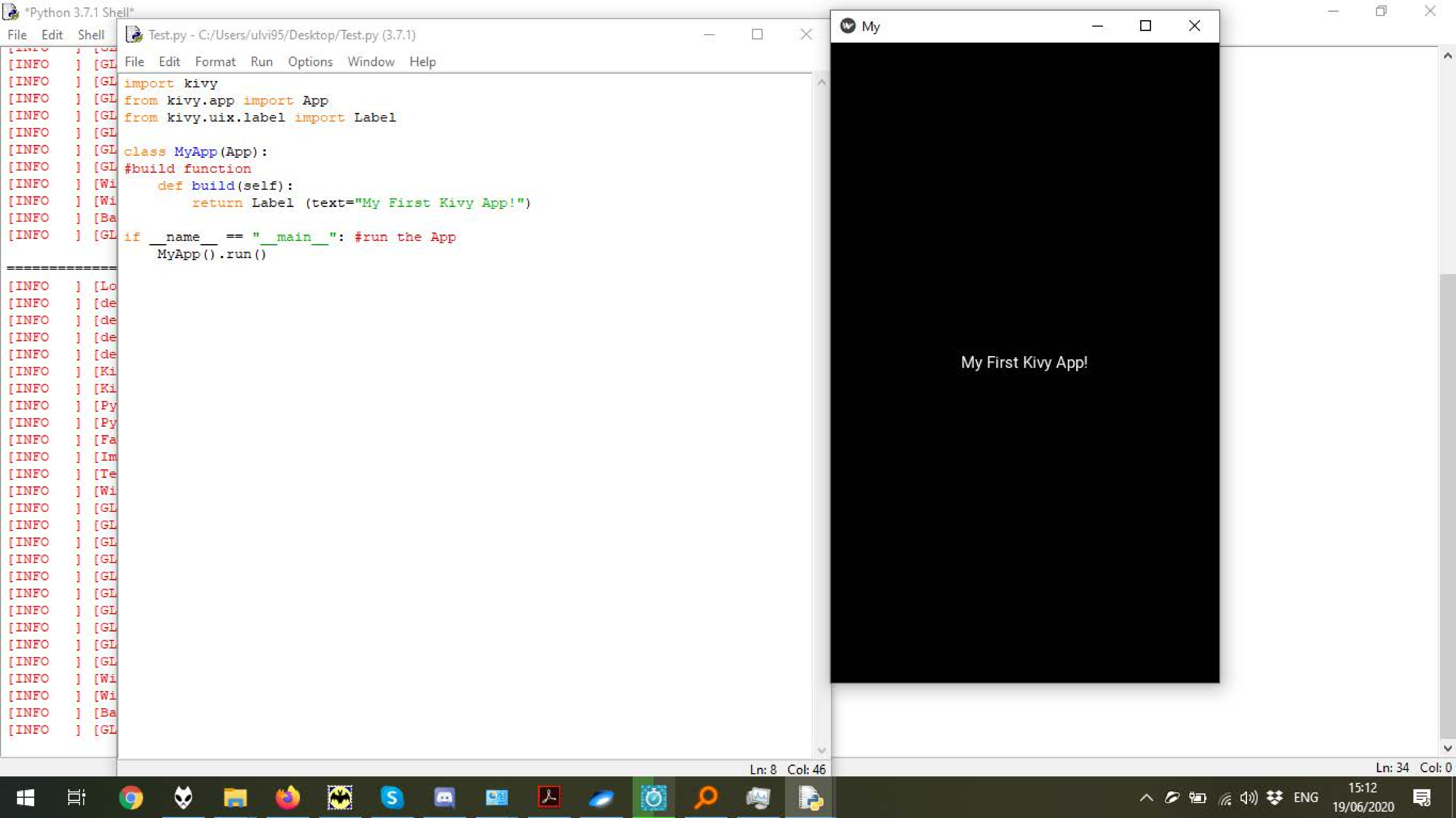
```
C:\WINDOWS\system32>py -m pip install kivy.deps.gstreamer
Collecting kivy.deps.gstreamer
  Downloading kivy_deps.gstreamer-0.2.0-cp37-cp37m-win32.whl (106.1 MB)
     |████████████████████████████████| 106.1 MB 70 kB/s
Installing collected packages: kivy.deps.gstreamer
Successfully installed kivy.deps.gstreamer

C:\WINDOWS\system32>py -m pip install kivy.deps.angle
Collecting kivy.deps.angle
  Downloading kivy_deps.angle-0.2.0-cp37-cp37m-win32.whl (4.3 MB)
     |████████████████████████████████| 4.3 MB 2.2 MB/s
Installing collected packages: kivy.deps.angle
Successfully installed kivy.deps.angle

C:\WINDOWS\system32>py -m pip install pygame
Collecting pygame
  Downloading pygame-1.9.6-cp37-cp37m-win32.whl (4.0 MB)
     |████████████████████████████████| 4.0 MB 819 kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.6

C:\WINDOWS\system32>_
```



```
C:\WINDOWS\system32>py -m pip install kivy
Collecting kivy
  Downloading Kivy-1.11.1-cp37-cp37m-win32.whl (3.7 MB)
     |████████████████████████████████| 3.7 MB 819 kB/s
Collecting Kivy-Garden>=0.1.4
  Downloading kivy-garden-0.1.4.tar.gz (6.8 kB)
Requirement already satisfied: docutils in c:\program files (x86)\python37-32\lib\site-packages (from kivy) (0.16)
Requirement already satisfied: pygments in c:\program files (x86)\python37-32\lib\site-packages (from kivy) (2.6.1)
Collecting requests
  Downloading requests-2.24.0-py2.py3-none-any.whl (61 kB)
     |████████████████████████████████| 61 kB 4.9 kB/s
Collecting idna<3,>=2.5
  Using cached idna-2.9-py2.py3-none-any.whl (58 kB)
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
  Using cached urllib3-1.25.9-py2.py3-none-any.whl (126 kB)
Collecting chardet<4,>=3.0.2
  Using cached chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Collecting certifi>=2017.4.17
  Downloading certifi-2020.4.5.2-py2.py3-none-any.whl (157 kB)
     |████████████████████████████████| 157 kB 930 kB/s
Building wheels for collected packages: Kivy-Garden
  Building wheel for Kivy-Garden (setup.py) ... done
  Created wheel for Kivy-Garden: filename=Kivy_Garden-0.1.4-py3-none-any.whl size=4534 sha256=23454e6bd936bc87b412282746cd8de4cc7a47addbadf17da4c54c5c4c3b68c2
  Stored in directory: c:\users\ulvi95\appdata\local\pip\cache\wheels\3f\43\e3\50289d555356f0421d1c388c82d052d5788f22a34d0cd8659d
Successfully built Kivy-Garden
Installing collected packages: idna, urllib3, chardet, certifi, requests, Kivy-Garden, kivy
Successfully installed Kivy-Garden-0.1.4 certifi-2020.4.5.2 chardet-3.0.4 idna-2.9 kivy-1.11.1 requests-2.24.0 urllib3-1.25.9

C:\WINDOWS\system32>
```

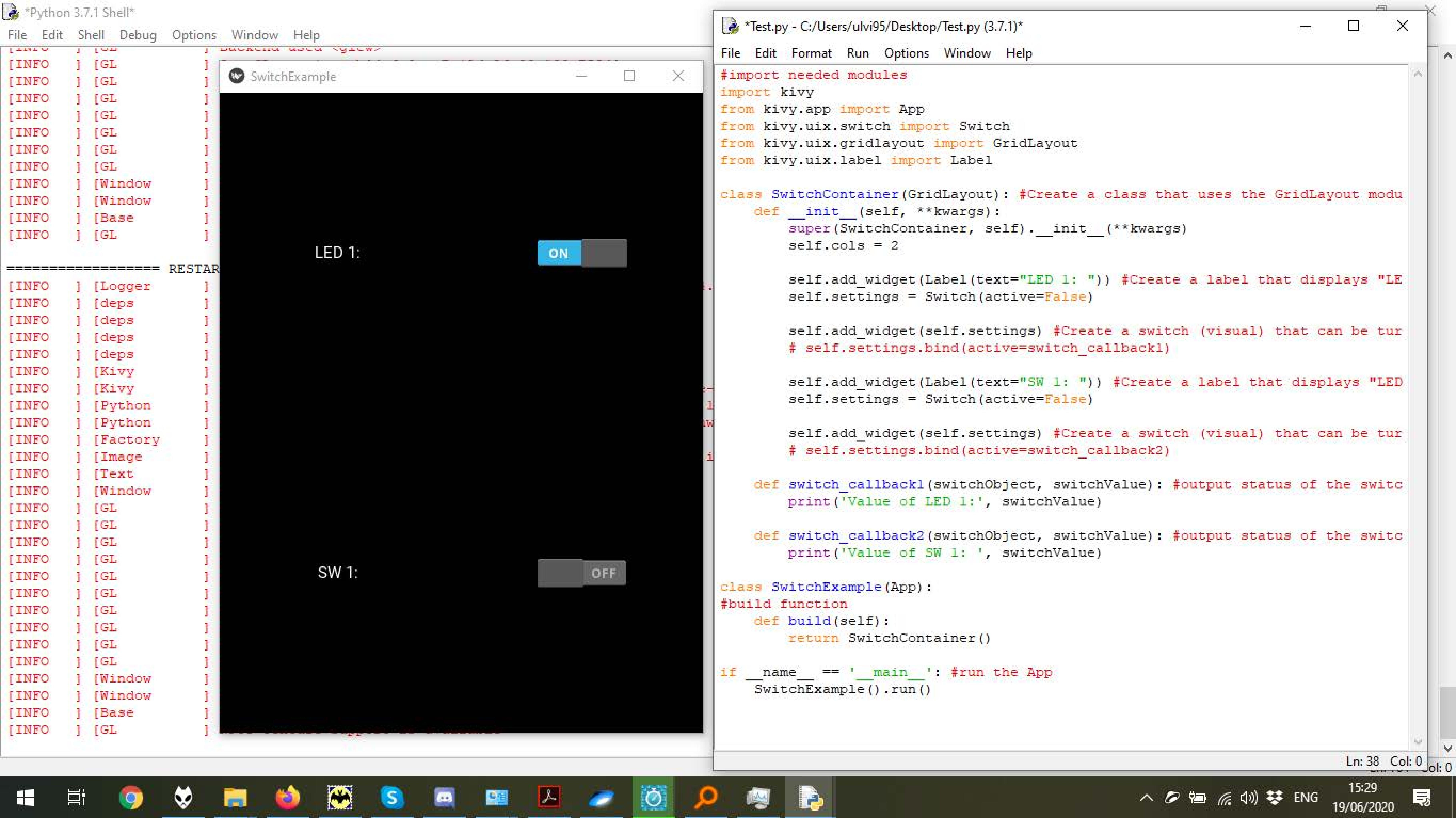


```
C:\WINDOWS\system32>py
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> Import kivy
  File "<stdin>", line 1
    Import kivy
              ^
SyntaxError: invalid syntax
>>> Import kivy
  File "<stdin>", line 1
    Import kivy
              ^
SyntaxError: invalid syntax
>>> import kivy
[WARNING] [Config      ] Older configuration version detected (0 instead of 21)
[WARNING] [Config      ] Upgrading configuration in progress.
[INFO   ] [Logger      ] Record log in C:\Users\ulvi95\.kivy\logs\kivy_20-06-19_0.txt
[INFO   ] [deps        ] Successfully imported "kivy_deps.gstreamer" 0.2.0
[INFO   ] [deps        ] Successfully imported "kivy_deps.angle" 0.2.0
[INFO   ] [deps        ] Successfully imported "kivy_deps.glew" 0.2.0
[INFO   ] [deps        ] Successfully imported "kivy_deps.sdl2" 0.2.0
[INFO   ] [Kivy        ] v1.11.1
[INFO   ] [Kivy        ] Installed at "C:\Program Files (x86)\Python37-32\lib\site-packages\kivy\__init__.py"
[INFO   ] [Python      ] v3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)]
[INFO   ] [Python      ] Interpreter at "C:\Program Files (x86)\Python37-32\python.exe"
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()

C:\WINDOWS\system32>
```



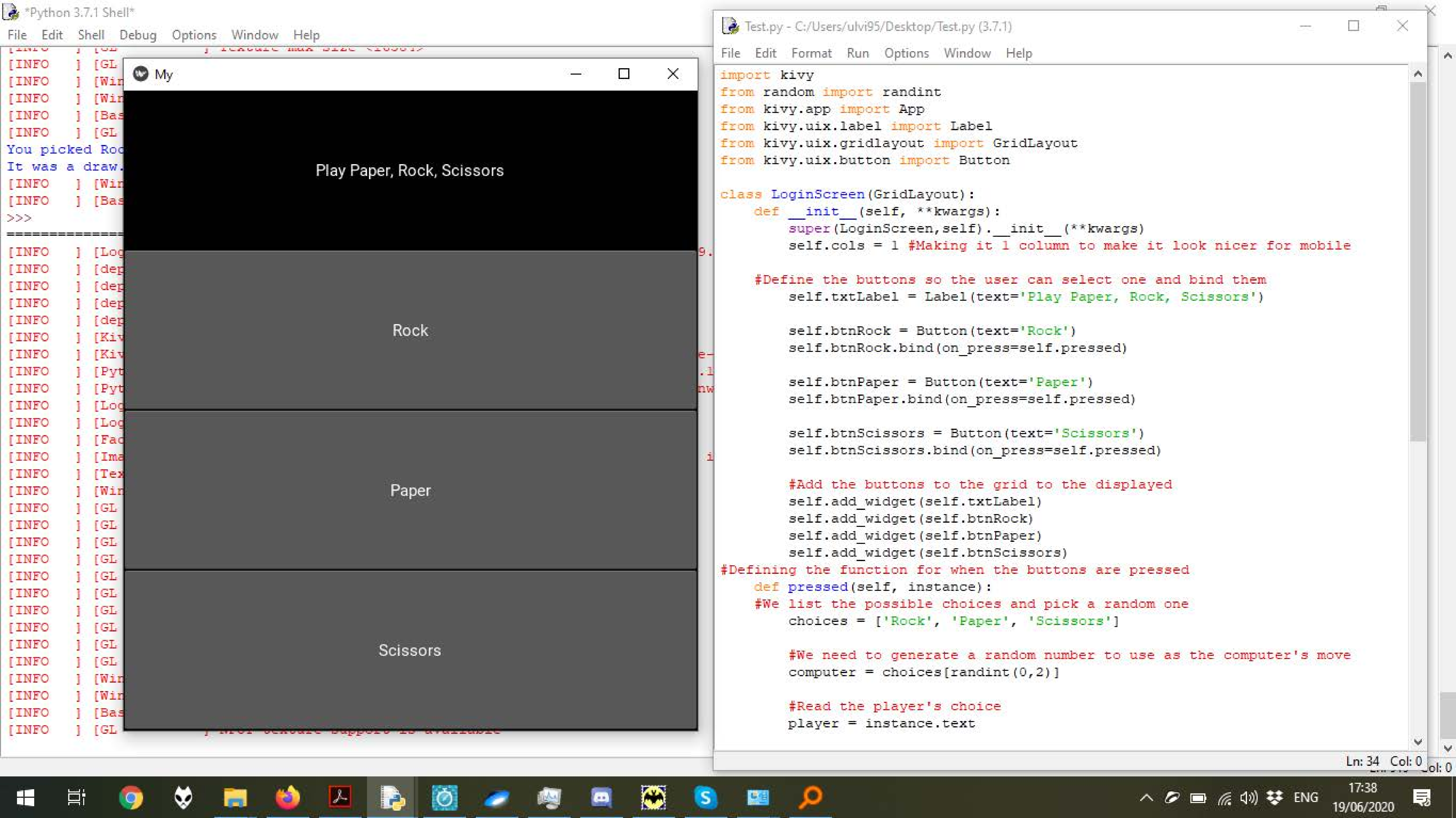
```
import kivy
from kivy.app import App
from kivy.uix.label import Label

class MyApp(App):
#build function
    def build(self):
        return Label (text="My First Kivy App!")

if __name__ == "__main__": #run the App
    MyApp().run()
```

```
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Wi
[INFO   ] [Wi
[INFO   ] [Ba
[INFO   ] [GL

=============
[INFO   ] [Lo
[INFO   ] [de
[INFO   ] [de
[INFO   ] [de
[INFO   ] [de
[INFO   ] [Ki
[INFO   ] [Ki
[INFO   ] [Py
[INFO   ] [Py
[INFO   ] [Fa
[INFO   ] [Im
[INFO   ] [Te
[INFO   ] [Wi
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Wi
[INFO   ] [Wi
[INFO   ] [Ba
[INFO   ] [GL
```


Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
SwitchExample
LED 1:
ON
SW 1:
OFF
Test.py - C:/Users/ulvi95/Desktop/Test.py (3.7.1)
File Edit Format Run Options Window Help
Ln: 38 Col: 0
ENG
15:29
19/06/2020

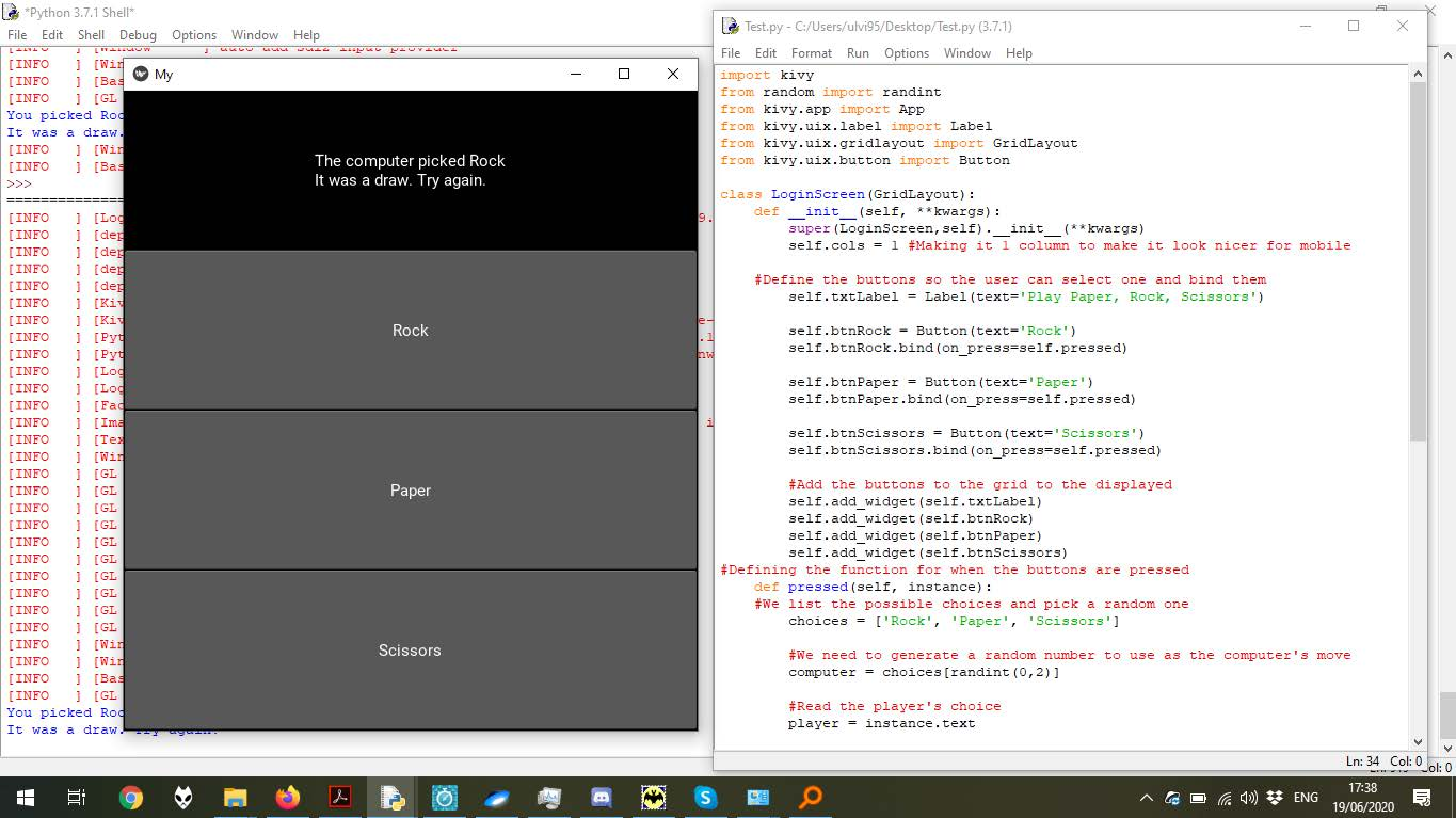
```
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [Window      ]
[INFO   ] [Window      ]
[INFO   ] [Base        ]
[INFO   ] [GL          ]

================== RESTAR
[INFO   ] [Logger      ]
[INFO   ] [deps        ]
[INFO   ] [deps        ]
[INFO   ] [deps        ]
[INFO   ] [deps        ]
[INFO   ] [Kivy        ]
[INFO   ] [Kivy        ]
[INFO   ] [Python      ]
[INFO   ] [Python      ]
[INFO   ] [Factory     ]
[INFO   ] [Image       ]
[INFO   ] [Text        ]
[INFO   ] [Window      ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [GL          ]
[INFO   ] [Window      ]
[INFO   ] [Window      ]
[INFO   ] [Base        ]
[INFO   ] [GL          ]
```

```python
#import needed modules
import kivy
from kivy.app import App
from kivy.uix.switch import Switch
from kivy.uix.gridlayout import GridLayout
from kivy.uix.label import Label

class SwitchContainer(GridLayout): #Create a class that uses the GridLayout modu
    def __init__(self, **kwargs):
        super(SwitchContainer, self).__init__(**kwargs)
        self.cols = 2

        self.add_widget(Label(text="LED 1: ")) #Create a label that displays "LE
        self.settings = Switch(active=False)

        self.add_widget(self.settings) #Create a switch (visual) that can be tur
        # self.settings.bind(active=switch_callback1)

        self.add_widget(Label(text="SW 1: ")) #Create a label that displays "LED
        self.settings = Switch(active=False)

        self.add_widget(self.settings) #Create a switch (visual) that can be tur
        # self.settings.bind(active=switch_callback2)

    def switch_callback1(switchObject, switchValue): #output status of the switc
        print('Value of LED 1:', switchValue)

    def switch_callback2(switchObject, switchValue): #output status of the switc
        print('Value of SW 1: ', switchValue)

class SwitchExample(App):
#build function
    def build(self):
        return SwitchContainer()

if __name__ == '__main__': #run the App
    SwitchExample().run()
```


Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
My
Play Paper, Rock, Scissors
Rock
Paper
Scissors
Test.py - C:/Users/ulvi95/Desktop/Test.py (3.7.1)
File Edit Format Run Options Window Help
Ln: 34 Col: 0
ENG
17:38
19/06/2020

```
[INFO   ] [GL          ] Texture max size <16384>
[INFO   ] [GL
[INFO   ] [Win
[INFO   ] [Win
[INFO   ] [Bas
[INFO   ] [GL
You picked Roc
It was a draw.
[INFO   ] [Win
[INFO   ] [Bas
>>>
==============
[INFO   ] [Log                                                    9.
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [Kiv
[INFO   ] [Kiv                                                    e-
[INFO   ] [Pyt                                                    .l
[INFO   ] [Pyt                                                    nw
[INFO   ] [Log
[INFO   ] [Log
[INFO   ] [Fac
[INFO   ] [Ima                                                     i
[INFO   ] [Tex
[INFO   ] [Win
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Win
[INFO   ] [Win
[INFO   ] [Bas
[INFO   ] [GL          ] NPOT texture support is available
```

```python
import kivy
from random import randint
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button

class LoginScreen(GridLayout):
    def __init__(self, **kwargs):
        super(LoginScreen,self).__init__(**kwargs)
        self.cols = 1 #Making it 1 column to make it look nicer for mobile

    #Define the buttons so the user can select one and bind them
        self.txtLabel = Label(text='Play Paper, Rock, Scissors')

        self.btnRock = Button(text='Rock')
        self.btnRock.bind(on_press=self.pressed)

        self.btnPaper = Button(text='Paper')
        self.btnPaper.bind(on_press=self.pressed)

        self.btnScissors = Button(text='Scissors')
        self.btnScissors.bind(on_press=self.pressed)

        #Add the buttons to the grid to the displayed
        self.add_widget(self.txtLabel)
        self.add_widget(self.btnRock)
        self.add_widget(self.btnPaper)
        self.add_widget(self.btnScissors)
#Defining the function for when the buttons are pressed
    def pressed(self, instance):
    #We list the possible choices and pick a random one
        choices = ['Rock', 'Paper', 'Scissors']

        #We need to generate a random number to use as the computer's move
        computer = choices[randint(0,2)]

        #Read the player's choice
        player = instance.text
```

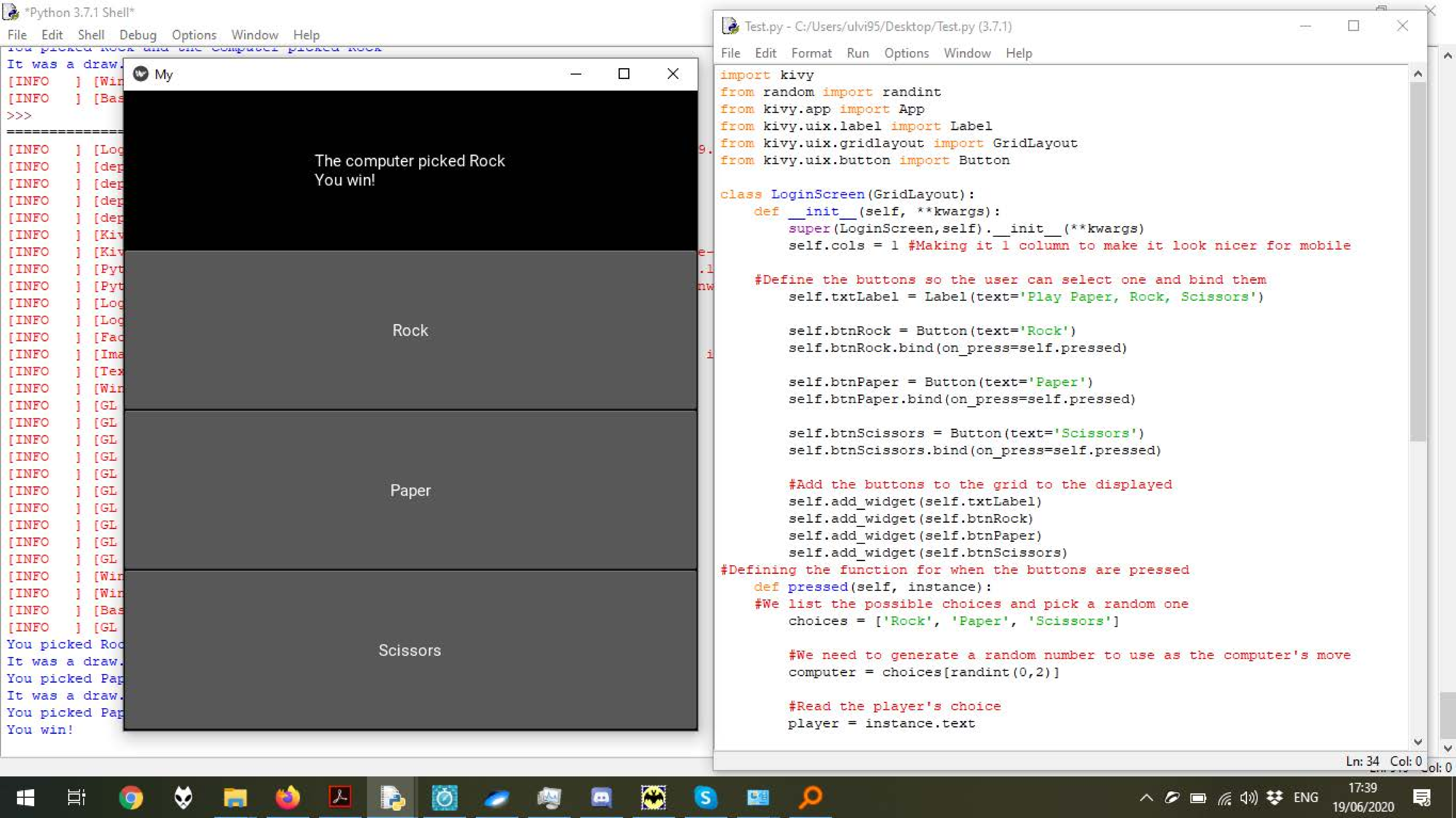


```
[INFO   ] [Window      ] auto add sdl2 input provider
[INFO   ] [Wir
[INFO   ] [Bas
[INFO   ] [GL
You picked Roc
It was a draw.
[INFO   ] [Wir
[INFO   ] [Bas
>>>
==============
[INFO   ] [Log                                           9.
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [Kiv
[INFO   ] [Kiv                                           e-
[INFO   ] [Pyt                                           .1
[INFO   ] [Pyt                                           nw
[INFO   ] [Log
[INFO   ] [Log
[INFO   ] [Fac
[INFO   ] [Ima                                            i
[INFO   ] [Tex
[INFO   ] [Wir
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Wir
[INFO   ] [Wir
[INFO   ] [Bas
[INFO   ] [GL
You picked Roc
It was a draw. Try again.
```

The computer picked Rock
It was a draw. Try again.

Rock
Paper
Scissors
Test.py - C:/Users/ulvi95/Desktop/Test.py (3.7.1)
File Edit Format Run Options Window Help

```python
import kivy
from random import randint
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button

class LoginScreen(GridLayout):
    def __init__(self, **kwargs):
        super(LoginScreen,self).__init__(**kwargs)
        self.cols = 1 #Making it 1 column to make it look nicer for mobile

    #Define the buttons so the user can select one and bind them
        self.txtLabel = Label(text='Play Paper, Rock, Scissors')

        self.btnRock = Button(text='Rock')
        self.btnRock.bind(on_press=self.pressed)

        self.btnPaper = Button(text='Paper')
        self.btnPaper.bind(on_press=self.pressed)

        self.btnScissors = Button(text='Scissors')
        self.btnScissors.bind(on_press=self.pressed)

        #Add the buttons to the grid to the displayed
        self.add_widget(self.txtLabel)
        self.add_widget(self.btnRock)
        self.add_widget(self.btnPaper)
        self.add_widget(self.btnScissors)
#Defining the function for when the buttons are pressed
    def pressed(self, instance):
    #We list the possible choices and pick a random one
        choices = ['Rock', 'Paper', 'Scissors']

        #We need to generate a random number to use as the computer's move
        computer = choices[randint(0,2)]

        #Read the player's choice
        player = instance.text
```

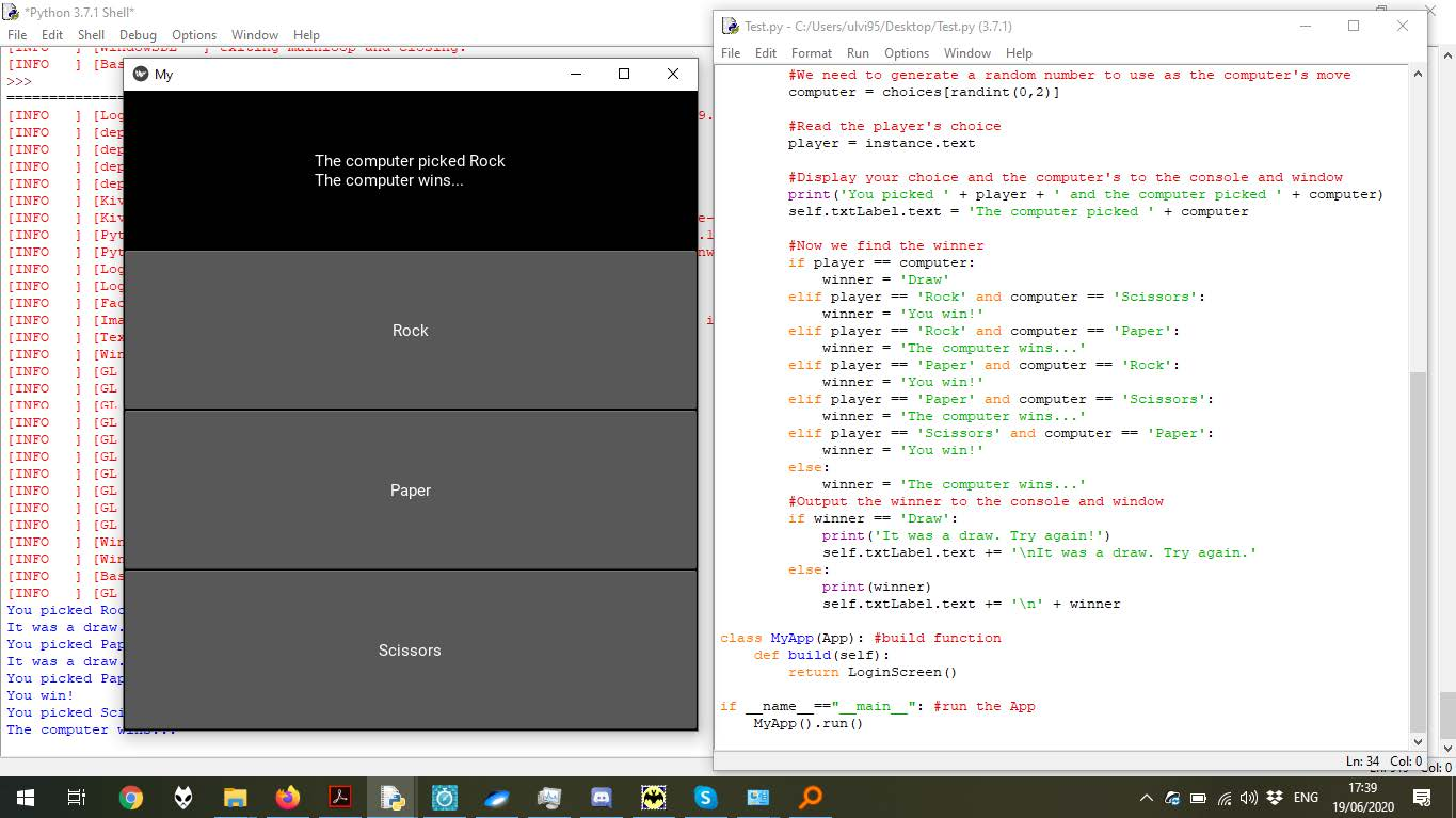


```
You picked Rock and the computer picked Rock
It was a draw.
[INFO   ] [Wir
[INFO   ] [Bas
>>>
==============
[INFO   ] [Log
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [Kiv
[INFO   ] [Kiv
[INFO   ] [Pyt
[INFO   ] [Pyt
[INFO   ] [Log
[INFO   ] [Log
[INFO   ] [Fac
[INFO   ] [Ima
[INFO   ] [Tex
[INFO   ] [Wir
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Wir
[INFO   ] [Wir
[INFO   ] [Bas
[INFO   ] [GL
You picked Roc
It was a draw.
You picked Pap
It was a draw.
You picked Pap
You win!
```

The computer picked Rock
You win!

Test.py - C:/Users/ulvi95/Desktop/Test.py (3.7.1)
File Edit Format Run Options Window Help

```
import kivy
from random import randint
from kivy.app import App
from kivy.uix.label import Label
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button

class LoginScreen(GridLayout):
    def __init__(self, **kwargs):
        super(LoginScreen,self).__init__(**kwargs)
        self.cols = 1 #Making it 1 column to make it look nicer for mobile

    #Define the buttons so the user can select one and bind them
        self.txtLabel = Label(text='Play Paper, Rock, Scissors')

        self.btnRock = Button(text='Rock')
        self.btnRock.bind(on_press=self.pressed)

        self.btnPaper = Button(text='Paper')
        self.btnPaper.bind(on_press=self.pressed)

        self.btnScissors = Button(text='Scissors')
        self.btnScissors.bind(on_press=self.pressed)

        #Add the buttons to the grid to the displayed
        self.add_widget(self.txtLabel)
        self.add_widget(self.btnRock)
        self.add_widget(self.btnPaper)
        self.add_widget(self.btnScissors)
#Defining the function for when the buttons are pressed
    def pressed(self, instance):
    #We list the possible choices and pick a random one
        choices = ['Rock', 'Paper', 'Scissors']

        #We need to generate a random number to use as the computer's move
        computer = choices[randint(0,2)]

        #Read the player's choice
        player = instance.text
```



```
[INFO   ] [WindowSDL   ] Exiting mainloop and closing.
[INFO   ] [Bas
>>>
==============
[INFO   ] [Log                                              9.
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [dep
[INFO   ] [Kiv
[INFO   ] [Kiv                                              e-
[INFO   ] [Pyt                                              .l
[INFO   ] [Pyt                                              nw
[INFO   ] [Log
[INFO   ] [Log
[INFO   ] [Fac
[INFO   ] [Ima                                              i
[INFO   ] [Tex
[INFO   ] [Win
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [GL
[INFO   ] [Win
[INFO   ] [Win
[INFO   ] [Bas
[INFO   ] [GL
You picked Roc
It was a draw.
You picked Pap
It was a draw.
You picked Pap
You win!
You picked Sci
The computer wins...
```

```
The computer picked Rock
The computer wins...
```

Test.py - C:/Users/ulvi95/Desktop/Test.py (3.7.1)
File Edit Format Run Options Window Help

```
        #We need to generate a random number to use as the computer's move
        computer = choices[randint(0,2)]

        #Read the player's choice
        player = instance.text

        #Display your choice and the computer's to the console and window
        print('You picked ' + player + ' and the computer picked ' + computer)
        self.txtLabel.text = 'The computer picked ' + computer

        #Now we find the winner
        if player == computer:
            winner = 'Draw'
        elif player == 'Rock' and computer == 'Scissors':
            winner = 'You win!'
        elif player == 'Rock' and computer == 'Paper':
            winner = 'The computer wins...'
        elif player == 'Paper' and computer == 'Rock':
            winner = 'You win!'
        elif player == 'Paper' and computer == 'Scissors':
            winner = 'The computer wins...'
        elif player == 'Scissors' and computer == 'Paper':
            winner = 'You win!'
        else:
            winner = 'The computer wins...'
        #Output the winner to the console and window
        if winner == 'Draw':
            print('It was a draw. Try again!')
            self.txtLabel.text += '\nIt was a draw. Try again.'
        else:
            print(winner)
            self.txtLabel.text += '\n' + winner

class MyApp(App): #build function
    def build(self):
        return LoginScreen()

if __name__=="__main__": #run the App
    MyApp().run()
```