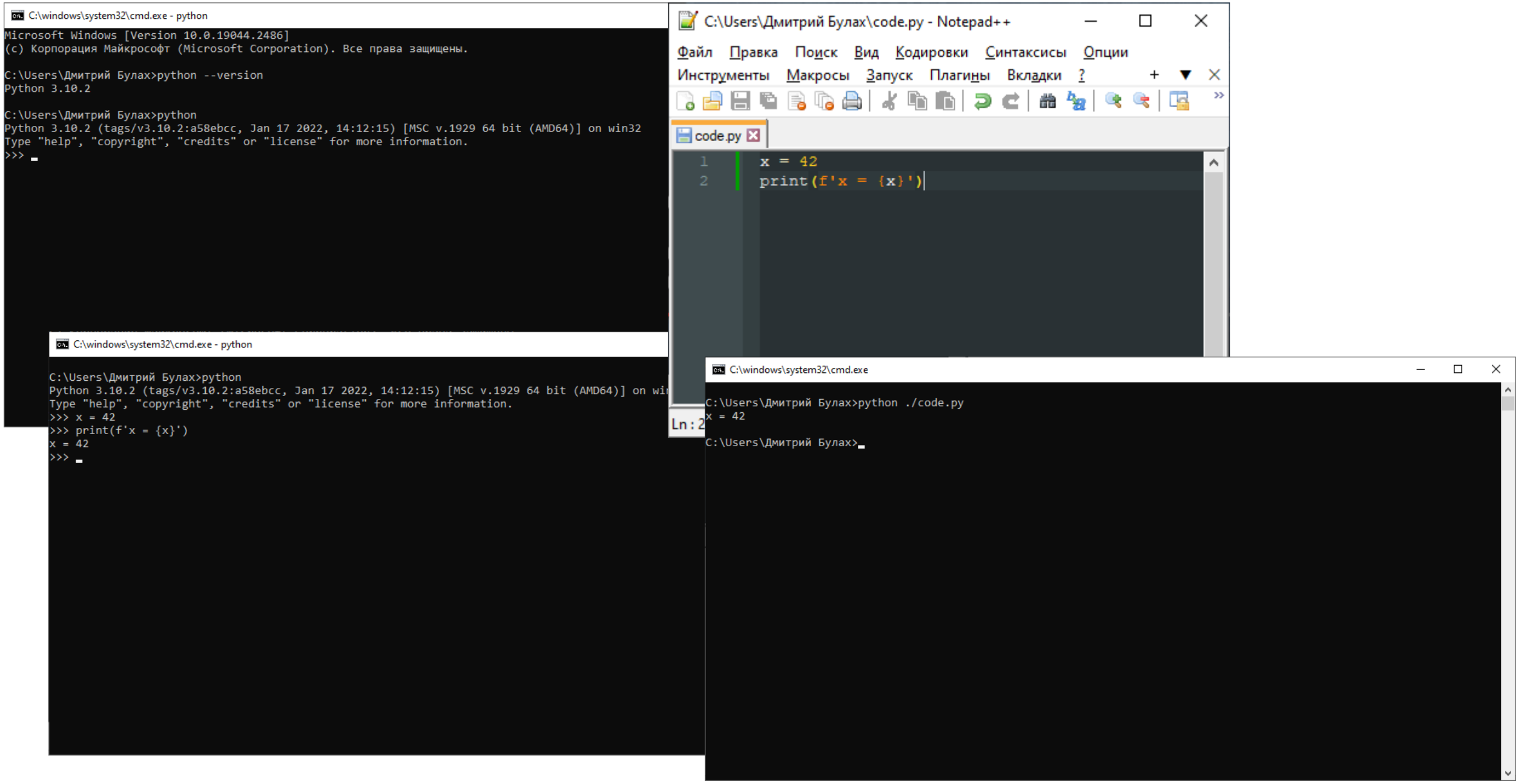
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Компьютерные технологии в научных исследованиях

Семинар №7

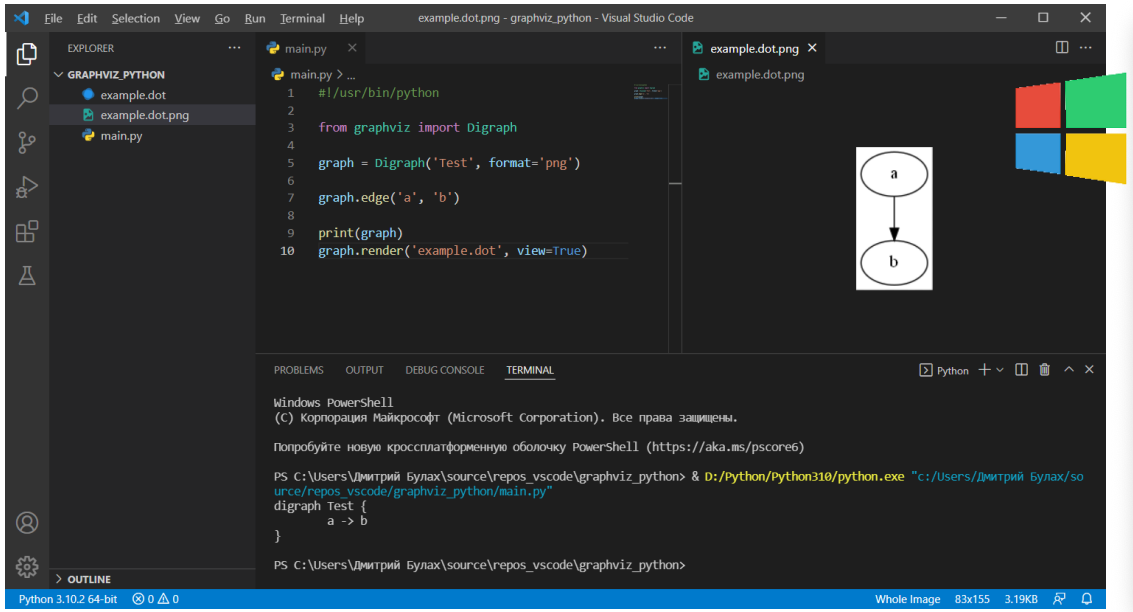
Язык Python. Основные сведения

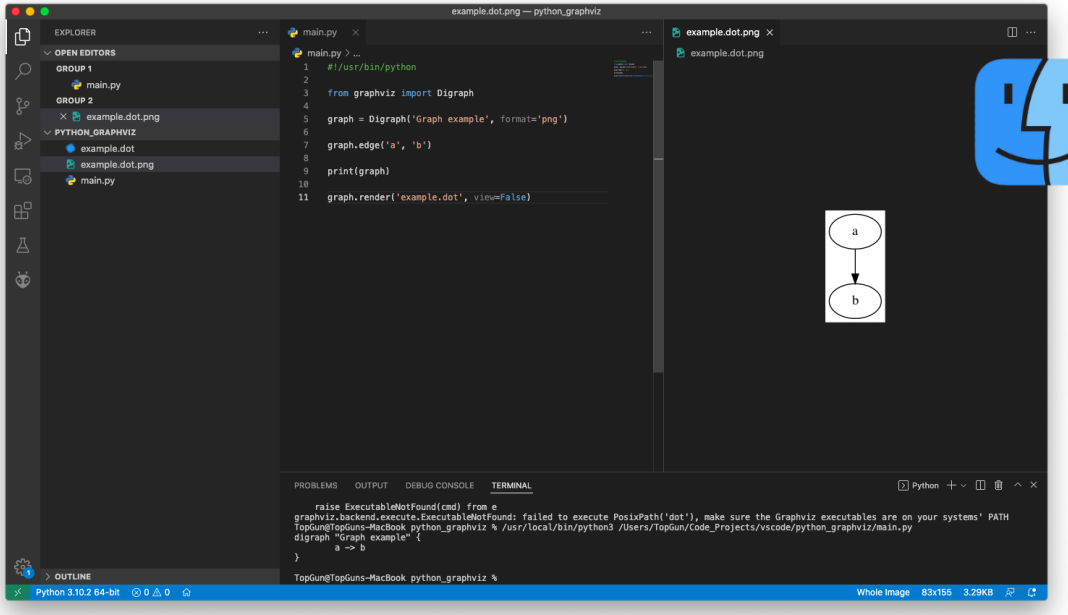
Запуск кода в Python

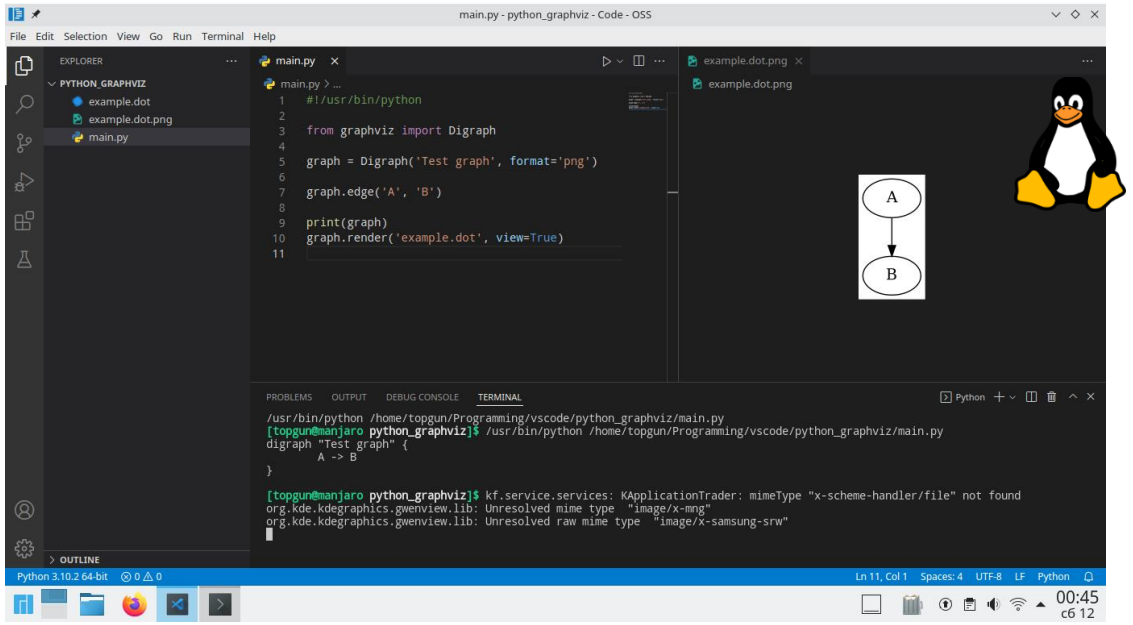


Написание кода в Python: Visual Studio Code

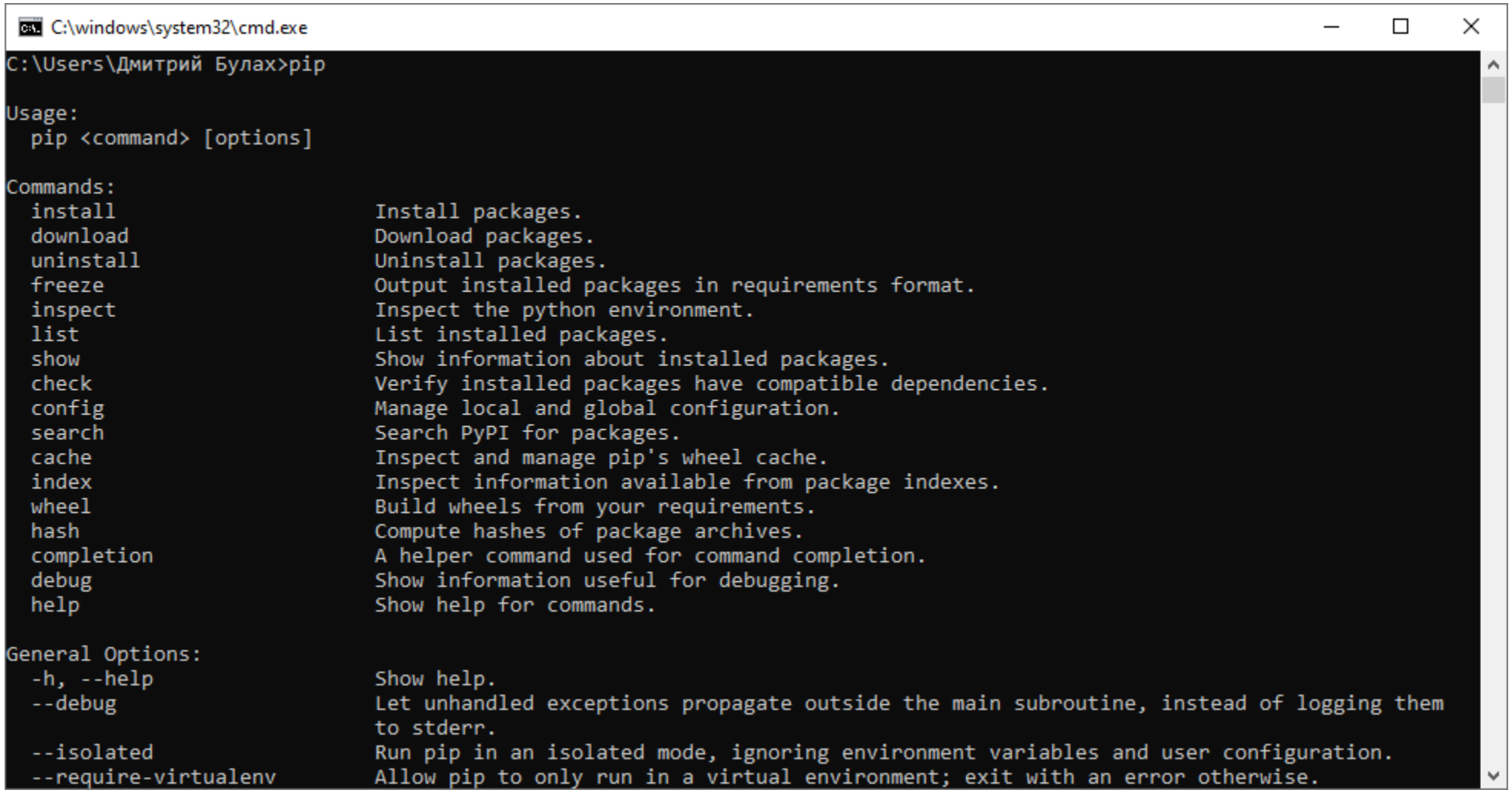
Visual Studio Code







Package Installer for Python (pip)

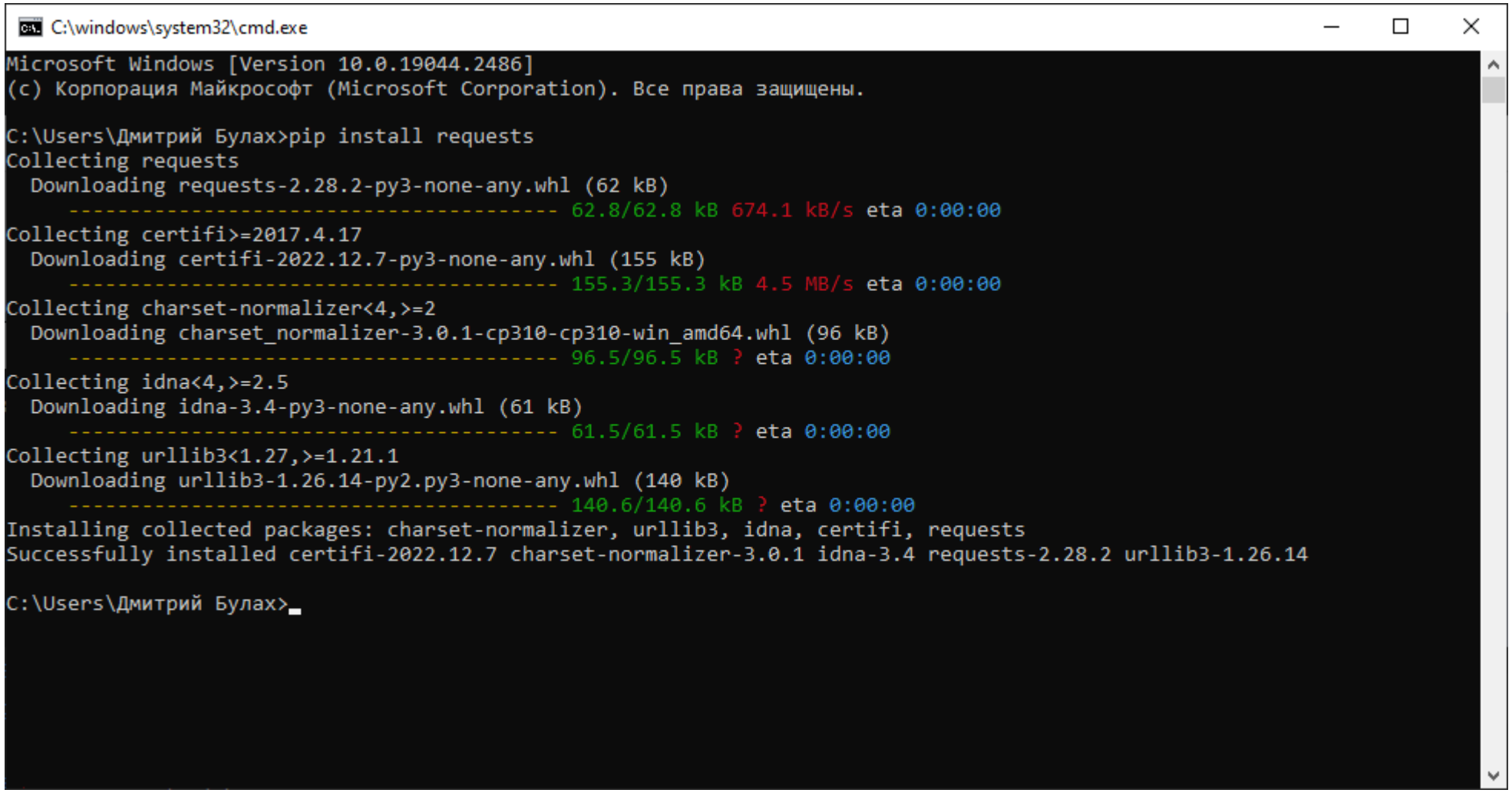


pip list: получение списка пакетов

```
C:\Users\Дмитрий Булах>pip list
Package            Version
------------------ ---------
bashplotlib        0.6.5
certifi            2022.12.7
charset-normalizer 3.0.1
click              8.1.3
colorama           0.4.6
cycler             0.11.0
fonttools          4.29.1
graphviz           0.19.1
idna               3.4
imgui              1.4.1
kiwisolver         1.3.2
matplotlib         3.5.1
mypy               0.931
mypy-extensions    0.4.3
numpy              1.22.2
opencv-python      4.5.5.64
packaging          21.3
Pillow             9.0.1
pip                22.3.1
progress           1.6
PTable             0.9.2
PyOpenGL           3.1.6
pyparsing          3.0.7
PyQt5              5.15.7
PyQt5-Qt5          5.15.2
PyQt5-sip          12.11.0
PySDL2             0.9.11
pysdl2-dll         2.0.20
PySimpleGUI        4.60.4
python-dateutil    2.8.2
requests           2.28.2
setuptools         58.1.0
six                1.16.0
tk                 0.1.0
tomli              2.0.1
typing_extensions  4.1.1
urllib3            1.26.14
```

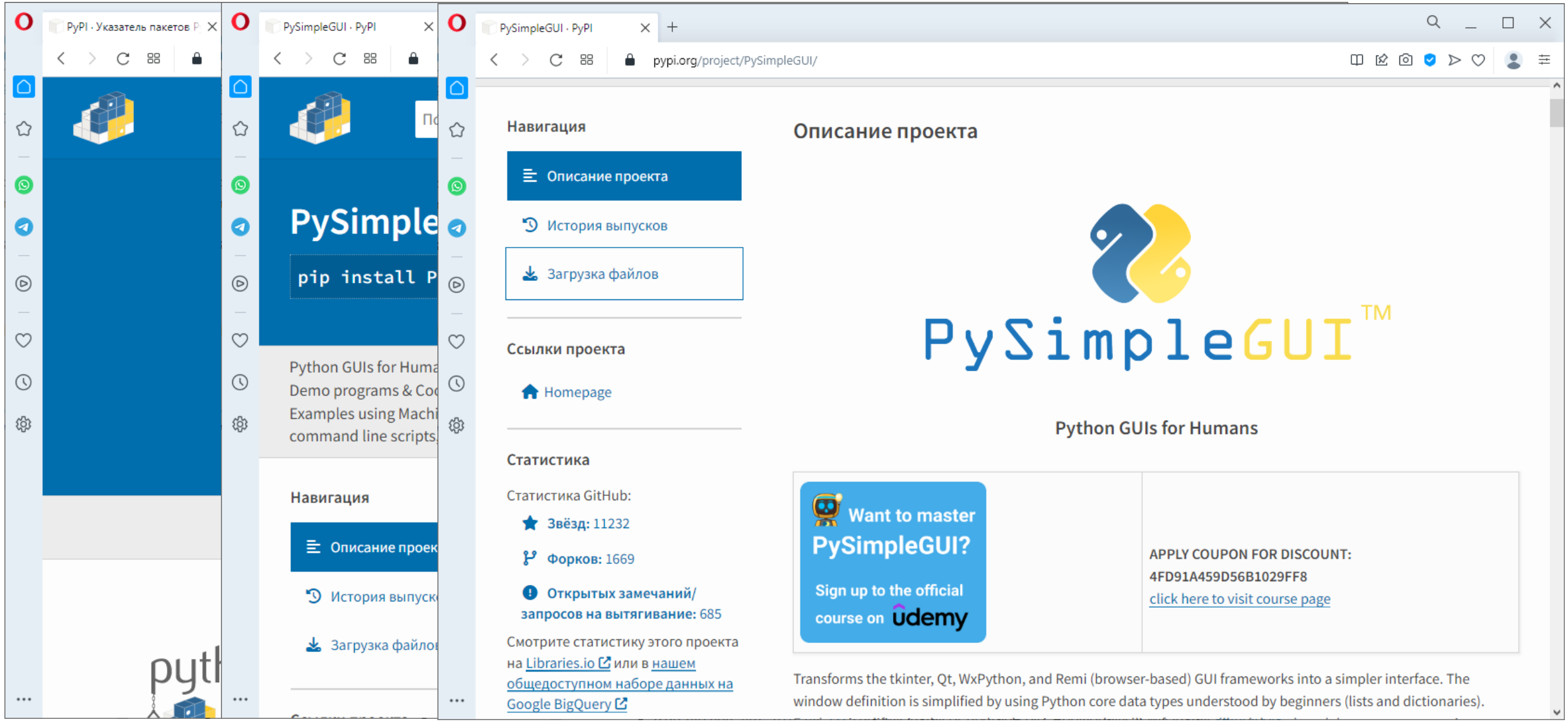
pip install: установка пакетов

```
>pip install <пакет1> [<пакет2>, ...]
```



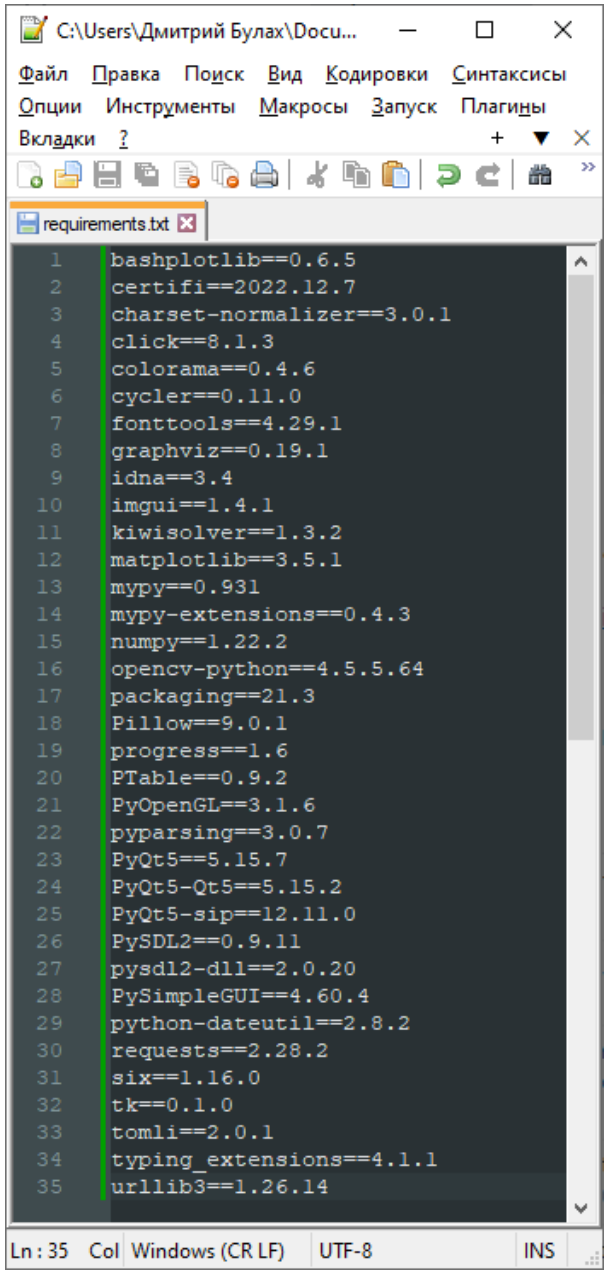
Откуда берётся информация о пакетах?

URL: https://pypi.org/



pip install: установка заданного перечня пакетов

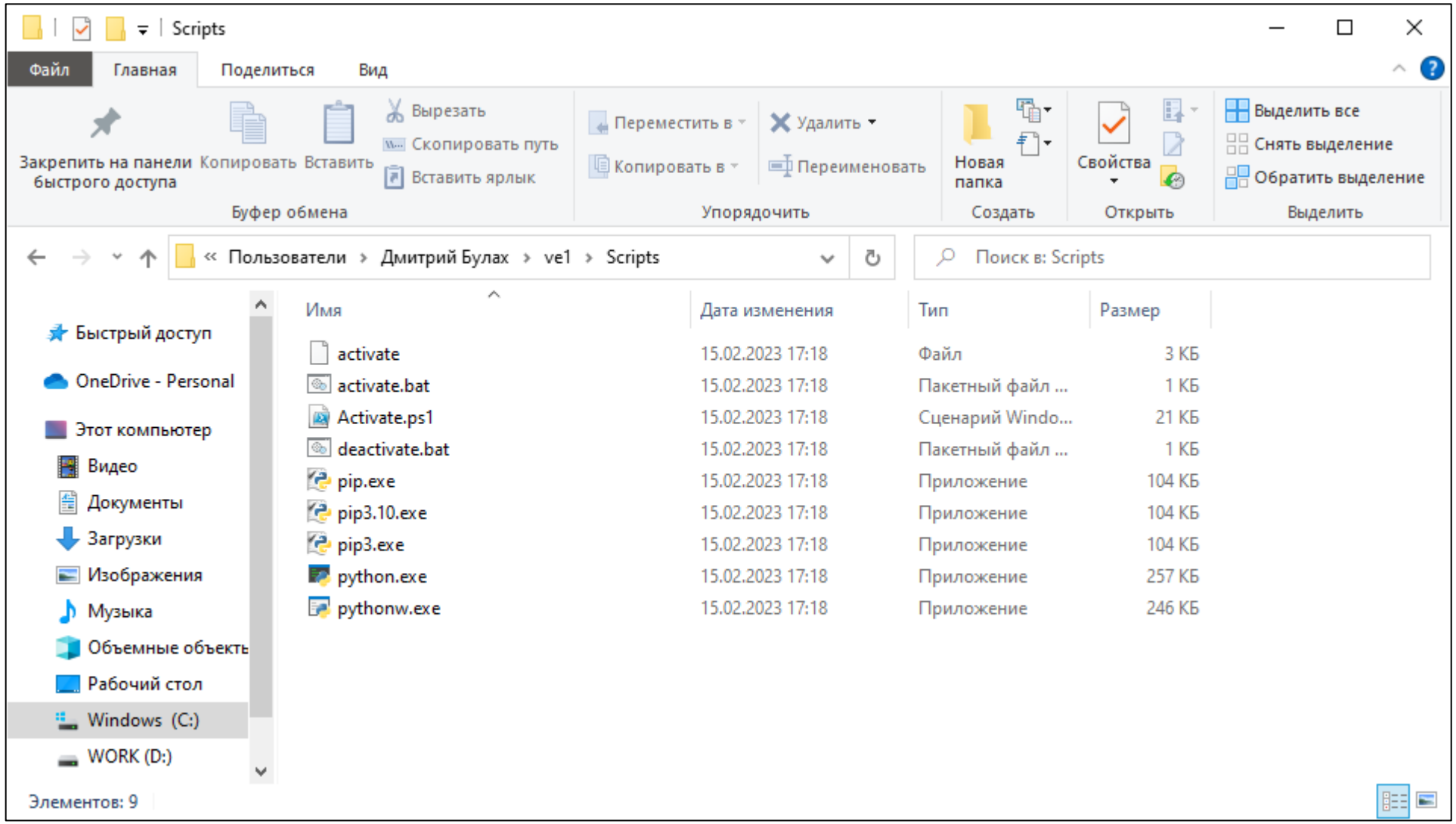
```
C:\Users\Дмитрий Булах>pip freeze > requirements.txt
```




```
C:\Users\Дмитрий Булах>pip install -r requirements.txt
```

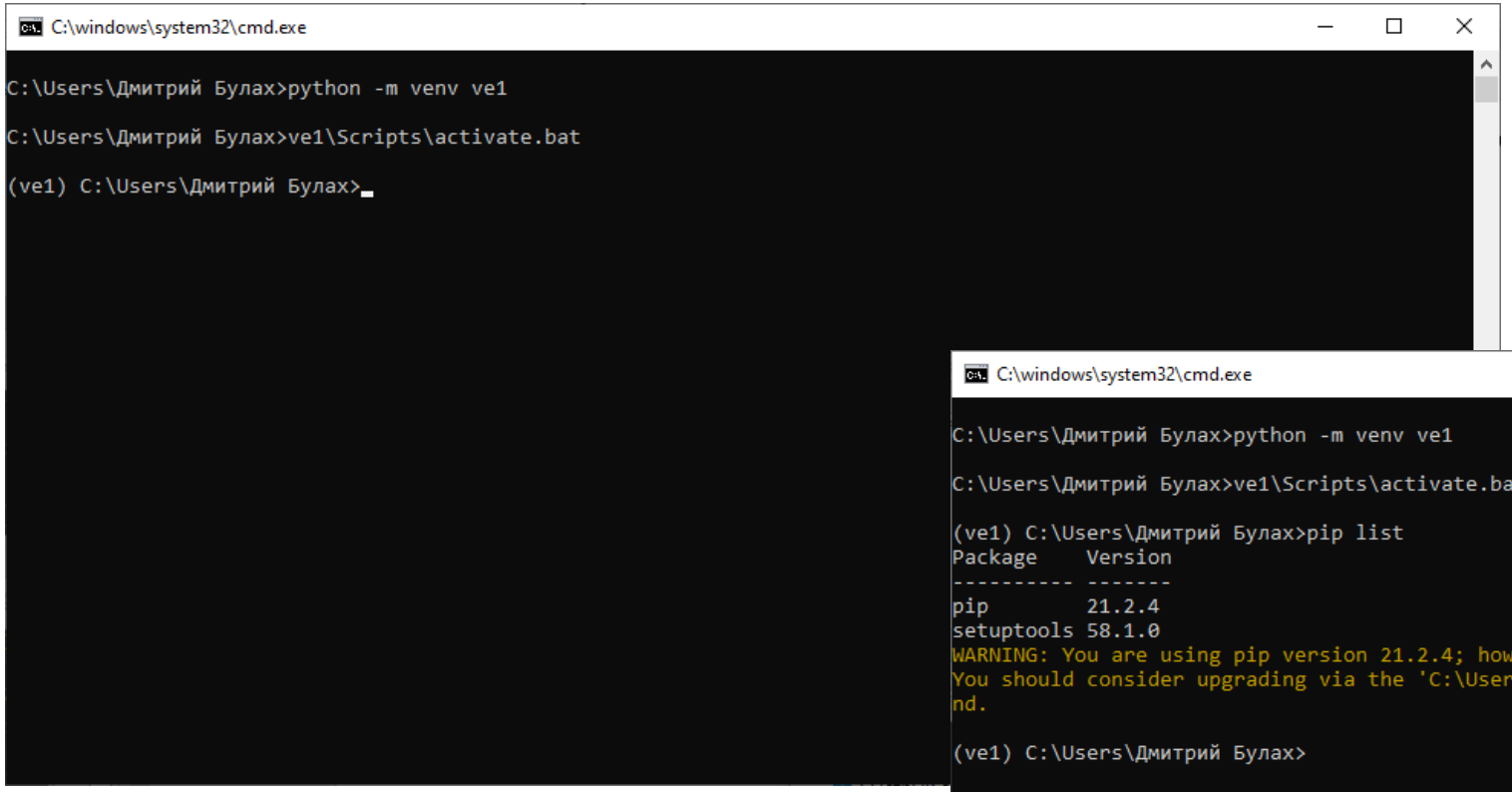
Виртуальное окружение в Python: создание

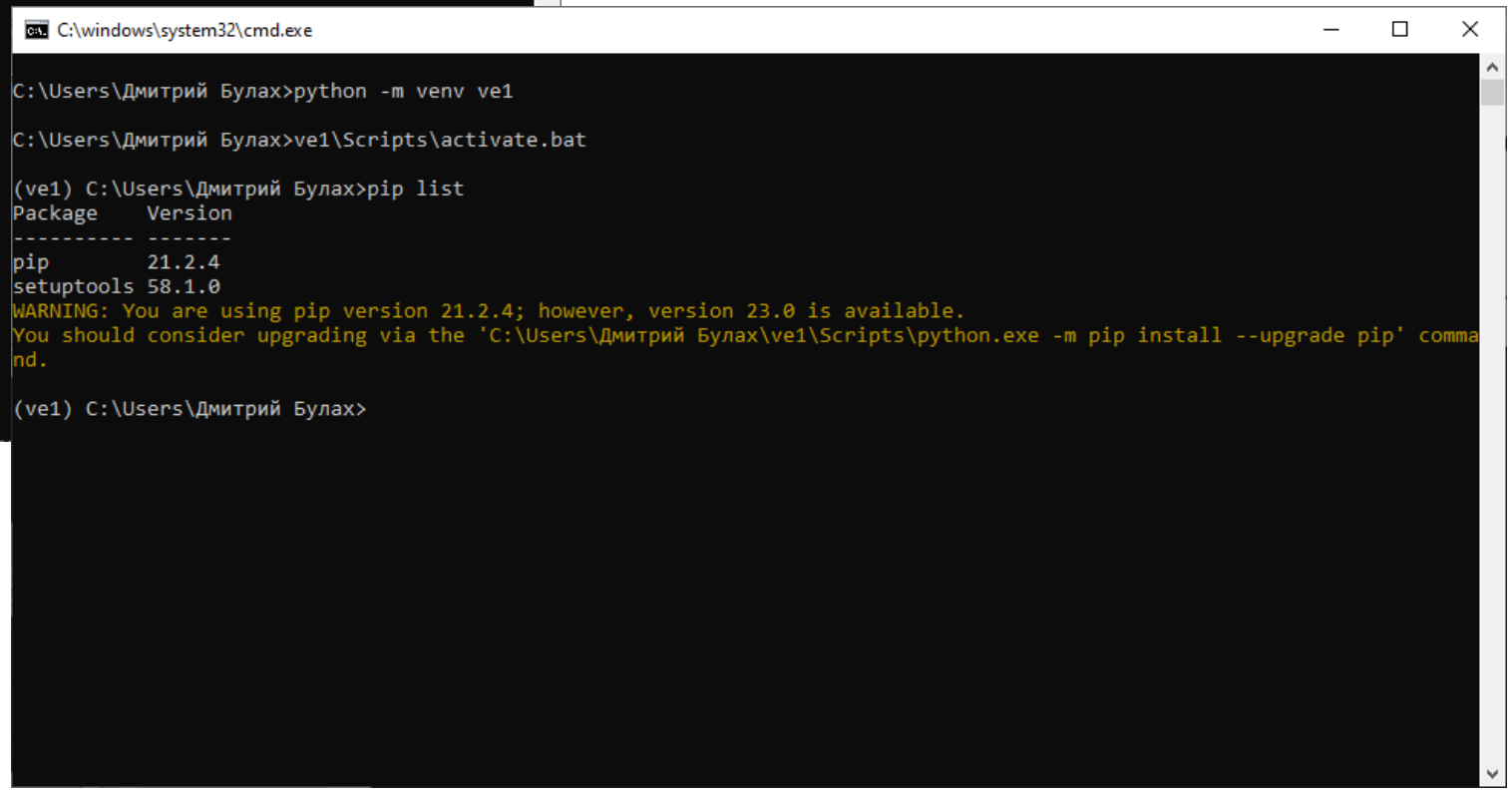
```
C:\Users\Дмитрий Булах>python -m venv ve1
```



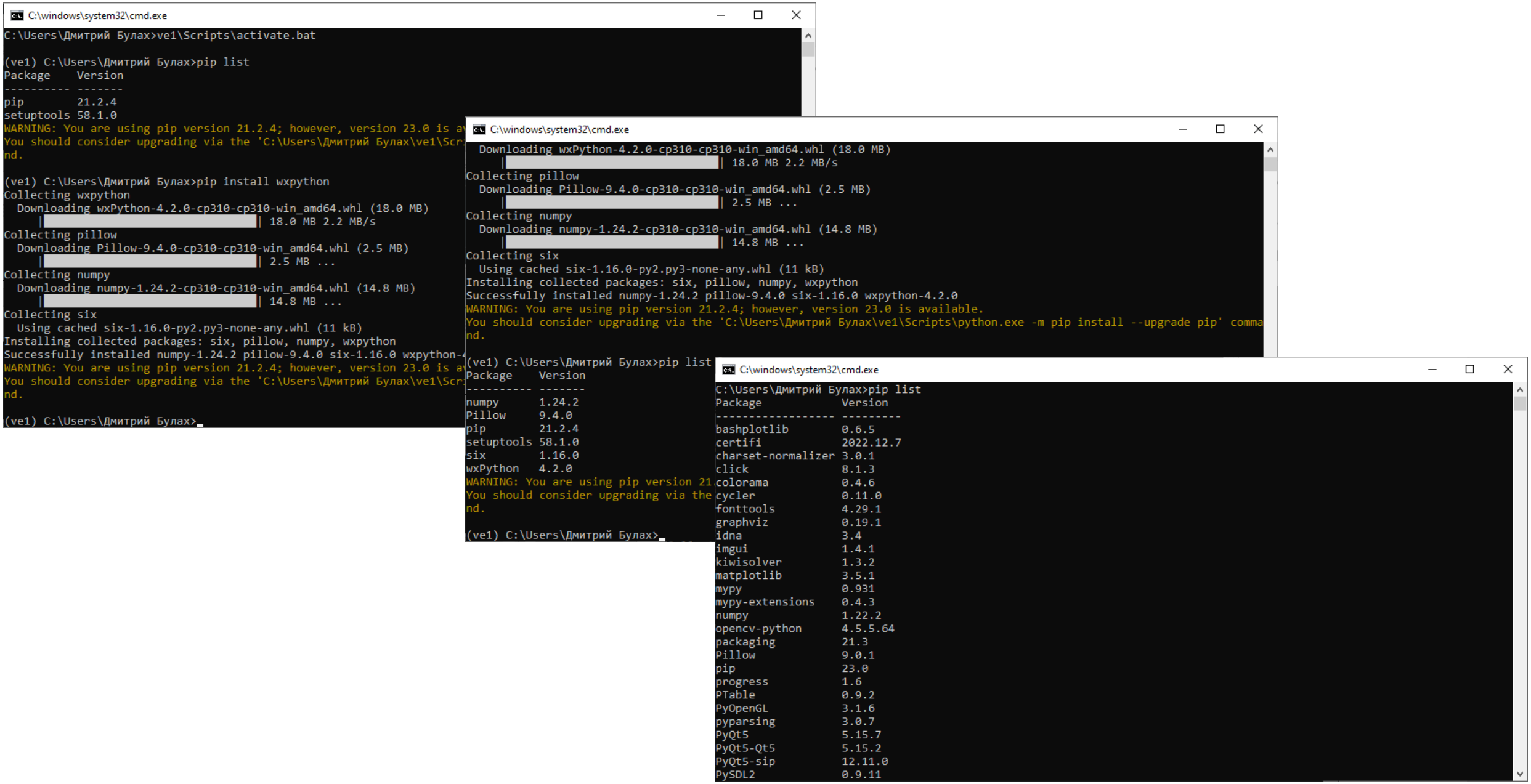
Виртуальное окружение в Python: активация

```
C:\Users\Дмитрий Булах>ve1\Scripts\activate.bat
```





Виртуальное окружение в Python: работа с venv



Синтаксис языка Python: вывод данных

```
print(*items, sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
x = 4
y = 2

print('x = {}, y = {}'.format(x, y))


print(f'x = {x}, y = {y}')


print(f'{{x}} = {x}, {{y}} = {y}')
```

Python2 и Python3

```
print "Hello, world!"

print 'Hello, world!'

s1 = 'A'
print "Value: %s" % s1

s1 = 'A'
s2 = 'B'
print "Values are: %(b)s and %(a)s" % {'a' : s1, 'b' : s2}
```

```
x = 4
y = 2

print('x = {}, y = {}'.format(x, y))

print(f'x = {x}, y = {y}')

print("Values: {1} {0}".format(x, y))
```

Строки : ASCII

Строки : Unicode

Целочисленное деление:
7/2 => 3

Целочисленное деление:
7/2 => 3.5

Типы данных в Python

В Python есть несколько стандартных типов данных:

- числа (Numbers)
  - целые (Integers)
  - вещественные (Real)
  - комплексные (Complex)
- строки (Strings)
- списки (Lists)
- множества (Sets)
- кортежи (Tuples)
- словари (Dictionaries)
- логический тип данных (Boolean)

Эти типы данных можно, в свою очередь, классифицировать по нескольким признакам:

- изменяемые (списки, словари и множества)
- неизменяемые (числа, строки и кортежи)
- упорядоченные (списки, кортежи, строки и словари)
- неупорядоченные (множества)

```
x = 42
print(f' value: {x} of type {type(x)}')
```

```
value: 42 of type <class 'int'>
```

Операции с целыми числами

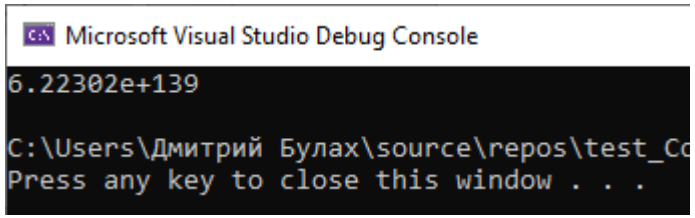
```
#!/usr/bin/python

x = 5
y = 3

print(5 / 3)
print(5 // 3)
print(5 % 3)
print(5 ** 3)
```

Операция	Результат
-x	Смена знака
x + y	Сумма двух чисел
x - y	Разность двух чисел
x * y	Произведение двух чисел
x / y	Частное чисел
x // y	Целая часть от деления
x % y	Остаток от деления
x ** y	Возведение x в степень y, x^y

```
1.6666666666666667
1
2
125
```

Длинная арифметика в Python

```cpp
#include <iostream>
#include <cmath>

int main() {
  int x = 5;
  int y = 200;
  std::cout << pow(x, y) << std::endl;
  return EXIT_SUCCESS;
}
```




```python
#!/usr/bin/python

x = 5
y = 200

print(x ** y)
```

622301527786114170714406405378012424059025216872116713310111661478969883403538344118394482312571361695696658955512248212471604347229003906 25

Работа с комплексными числами

```
#!/usr/bin/python

x = complex(4, 2)
y = 2 + 4j

print(x)
print(y)
print(x + y)

print(type(x))
print(type(y))
print(type(x + y))
```

```
PS C:\Users\...\numpy> & D:/Python/python.exe "c:/Users/.../numpy/main.py"
(4+2j)
(2+4j)
(6+6j)
<class 'complex'>
<class 'complex'>
<class 'complex'>
PS C:\Users\...\numpy>
```

Списки (Lists)

```
lst = [1, 2, 3, 4, 5]
print(lst)
```

```
[1, 2, 3, 4, 5]
```

```
print(len(lst))
```

```
5
```

```
lst = list('ПКИМС')
print(lst)
```

```
['П', 'К', 'И', 'М', 'С']
```

```
lst = [1, 2, 3, 'ПКИМС', 4]
print(lst)
```

```
[1, 2, 3, 'ПКИМС', 4]
```

```
print(type(lst))
```

```
<class 'list'>
```

```
lst = []
lst.append(1)
lst.append(2)
lst.append(3)
```

```
lst.remove(1)
```

```
lst.reverse()
```

```
lst.count(1)
```

```
lst.clear()
```

Множества (Sets) (1)

```
s = {1, 2, 3, 4, 5}
print(s)
```

```
{1, 2, 3, 4, 5}
```

```
print(len(s))
```

```
5
```

```
s = {1, 2, 1, 2, 3}
print(s)
```

```
{1, 2, 3}
```

```
l = [1, 2, 1, 2, 3]
print(l)
```

```
[1, 2, 1, 2, 3]
```

```
s = set(l)
print(s)
```

```
{1, 2, 3}
```

```
s.add(4)
s.add(8)
s.add(7)
```

```
{1, 2, 3, 4, 7, 8}
```

```
s.discard(2)
```

```
{1, 3, 4, 7, 8}
```

Множества (Sets) (2)

```
s1 = {1, 2, 3, 4, 5}
s2 = {4, 5, 6, 7, 8}
s3 = {4, 5}
print(s1.intersection(s2))
```

```
{4, 5}
```

```
print(s1.difference(s2))
```

```
{1, 2, 3}
```

```
print(s1.issubset(s3))
```

```
False
```

```
print(s3.issubset(s1))
```

```
True
```

```
print(s1.union(s2))
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

```
print(s1.issuperset(s3))
```

```
True
```

Кортежи (Tuples)

```
t = (1, 2, 3)
print(t)
```

```
(1, 2, 3)
```

```
lst = [1, 2, 3]
t = tuple(lst)
print(t)
```

```
(1, 2, 3)
```

```
print(len(t))
```

```
3
```

Словари (Dictionaries)

```
dict = {'name' : 'v1', 'type': 'source', 'value': 5}
print(dict)
```

```
{'name': 'v1', 'type': 'source', 'value': 5}
```

```
dict = {
    'v1' : { 'type': 'source', 'value': 5 },
    'r1' : { 'type': 'resistor', 'value': 1000 }
}
```

```
{'v1': {'type': 'source', 'value': 5}, 'r1': {'type': 'resistor', 'value': 1000}}
```

```
dict['c1'] = {'type': 'capacitor', 'value': 10**-12}
```

Упорядоченные и неупорядоченные типы данных

Список:

```
l = list('Hello, world')

print(l)
```

```
['H', 'e', 'l', 'l', 'o', ',', ' ', 'w', 'o', 'r', 'l', 'd']
```

Множество:

```
s = set('Hello, world')

print(s)
```

```
{'d', 'e', 'H', 'r', 'o', ' ', 'l', 'w', ','}
```

Синтаксис языка Python: условия

```
var1 = 4
var2 = 6

if var1 == var2:
    print("var1 = var2")
else:
    print("var1 != var2")
```

```
var1 = 4
var2 = 6

if var1 == var2:
    print("var1 = var2")
elif var1 < var2:
    print("var1 < var2")
else:
    print("var1 > var2")
```

Трёхместное выражение:

```
X = input()

if X == 'Y':
    A = True
else:
    A = False
```

```
A = True if X == 'Y' else False
```

Синтаксис языка Python: циклы

```
for i in range(10):
    print(i)


for i in range(1, 10):
    print(i)


for i in range(1, 10, 2):
    print(i)
```

```
lst = [1, 'two', 3, 'four', 5]

for i in lst:
    print(f'i={i} of type {type(i)}')
```

```
i=1 of type <class 'int'>
i=two of type <class 'str'>
i=3 of type <class 'int'>
i=four of type <class 'str'>
i=5 of type <class 'int'>
```

Python: работа с файлами

```
f = open("myfile.txt", 'r')

text = f.read()

print (text)

f.close()
```

```
f = open("myfile.txt", 'r')

for line in f:
  print (line)

f.close()
```

```
f = open("myfile.txt", 'r')

while True:
  line = f.readline()
  if line == "":
    break
  print (line)

f.close()
```

Синтаксис языка Python: функции (1)

```
def func1(a, b, strval):
  c = a+b
  print (strval)
  return c

sum = func1(4, 5, "String value!")
print (sum)
```

```
sum = func1(strval ="String", a=4, b=6)
print(sum)
```

Документирование программного кода в Python

```python
def RequestAirHT(sensorId : int) -> Tuple[float, float]:
    """
    Функция посылает запрос типа GET для получения значений температуры и
    влажности воздуха
    URL для запроса: https://dt.miet.ru/ppo_it/api/temp_hum/<number>
    Вход : (int) номер датчика в диапазоне [1..4]
    Выход: (int, int) значения температуры и влажности воздуха
    """
    ret = requests.get(f'https://dt.miet.ru/ppo_it/api/temp_hum/{sensorId}')

    print(f'Status: {ret.status_code} Answer: {ret.text}')

    json_code = json.loads(ret.text)

    return (float(json_code['temperature']), float(json_code['humidity']))
```

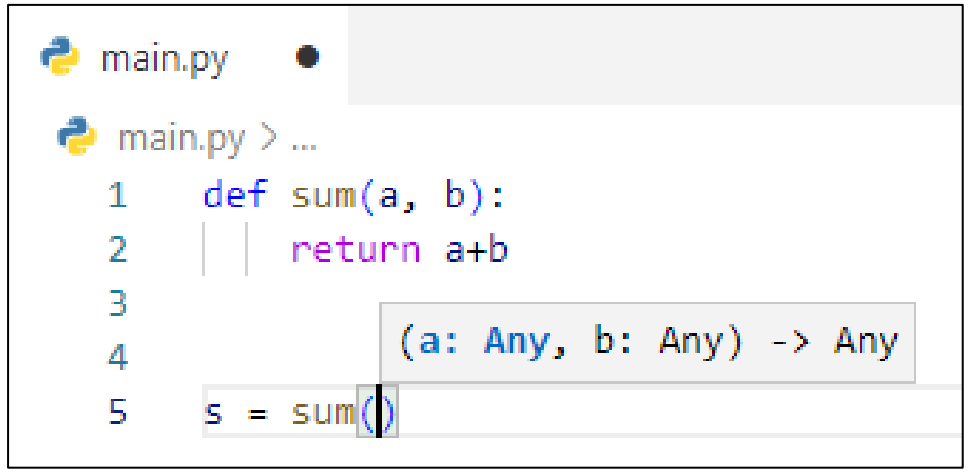
```python
print(help(RequestAirHT))
```

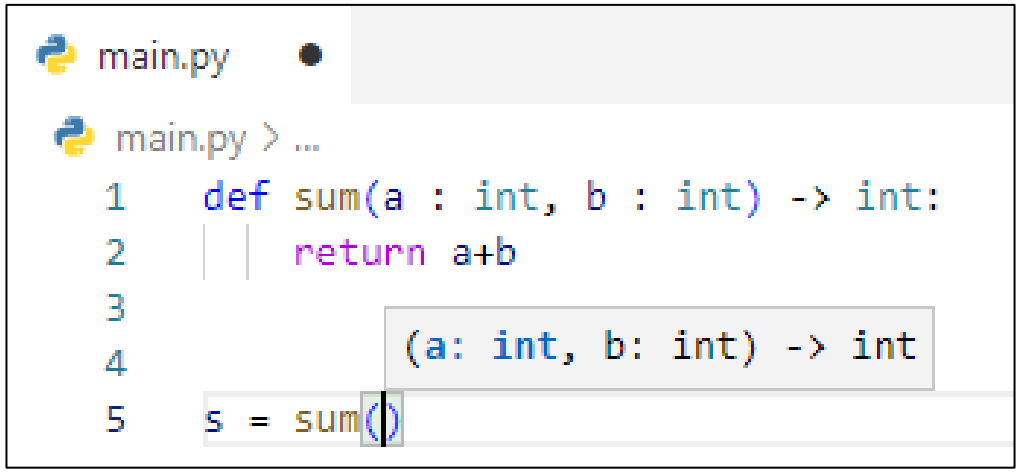
```python
AH1, AT1 = RequestAirHT(1)
```

```
Help on function RequestAirHT in module __main__:

RequestAirHT(sensorId: int)
    Функция посылает запрос типа GET для получения значений температуры и
     влажности воздуха
    URL для запроса: https://dt.miet.ru/ppo_it/api/temp_hum/<number>
    Вход : (int) номер датчика в диапазоне [1..4]
    Выход: (int, int) значения температуры и влажности воздуха

None
```

Синтаксис языка Python: аннотирование типов данных

```python
def RequestGroundH(sensorId : int) -> float:
    """
    Функция посылает запрос типа GETдля получения значения влажности почвы
    URL для запроса: https://dt.miet.ru/ppo_it/api/hum/<number>
    Вход : (int) номер датчика в диапазоне [1..6]
    Выход: (int, int) значения температуры и влажности воздуха
    """
    ret = requests.get(f'https://dt.miet.ru/ppo_it/api/hum/{sensorId}')
    print(f'Status: {ret.status_code} Answer: {ret.text}')
    json_code = json.loads(ret.text)
    return (float(json_code['humidity']))
```





Регулярные выражения в Python

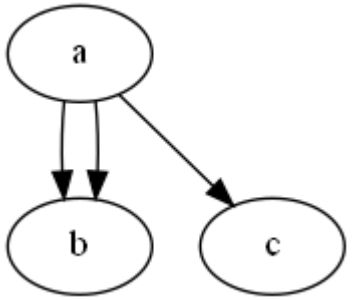
```
import re

ret = re.search(r'^\s*([Rr]\w+)\s+(\w+)\s+(\w+)', line)
```

```
import re

line = 'R1 1 2 1K'
ret = re.search(r'^\s*([Rr]\w+)\s+(\w+)\s+(\w+)', line)
if ret:
    print(ret.group(0))

    print(ret.group(1))

    print(ret.group(2))

    print(ret.group(3))
```

Модули в Python: graphviz

Язык DOT

```
digraph {
    a -> b
    a -> b
    a -> c
}
```



Язык Python и библиотека graphviz

```python
from graphviz import Digraph

graph = Digraph()

graph.edge('a', 'b')
graph.edge('a', 'b')
graph.edge('a', 'c')

graph.render('result.dot', format='png', view=True)
```

Модули в Python: numpy


```
import numpy as np
```

```
a = np.array([1, 2, 3])
```

Создание массива по диапазону:

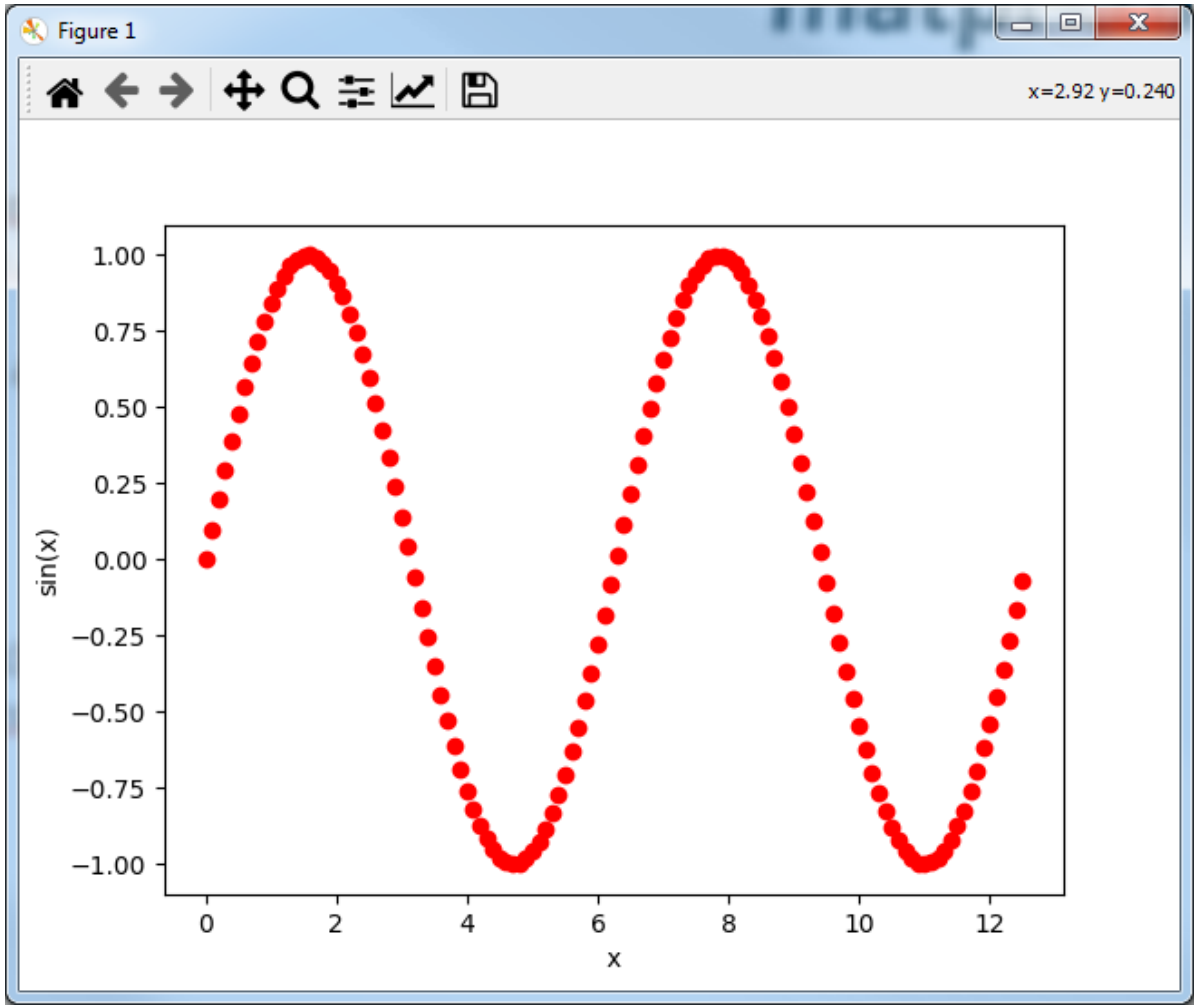
```
a = np.arange(0, 10, 1)
a = np.arange(0, 1, 0.1)
```

Создание массива по диапазону и числу разбиений:

```
a = np.linspace(0, 2, 7)
```

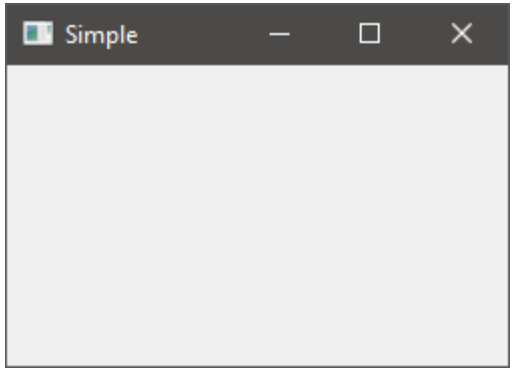
Модули в Python: matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 4*np.pi, 0.1)
y = np.sin(x)


plt.xlabel('x')
plt.ylabel('sin(x)')
plt.plot(x, y, 'ro')
plt.show()
```



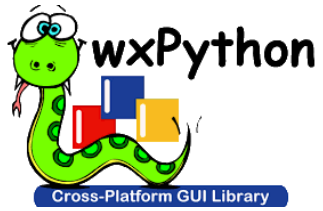
Модули в Python: PyQt5


```python
import sys
from PyQt5.QtWidgets import QApplication, QWidget

if __name__ == '__main__':

    app = QApplication(sys.argv)

    win = QWidget()
    win.resize(250, 150)
    win.move(300, 300)
    win.setWindowTitle('Simple')
    win.show()

    sys.exit(app.exec_())
```

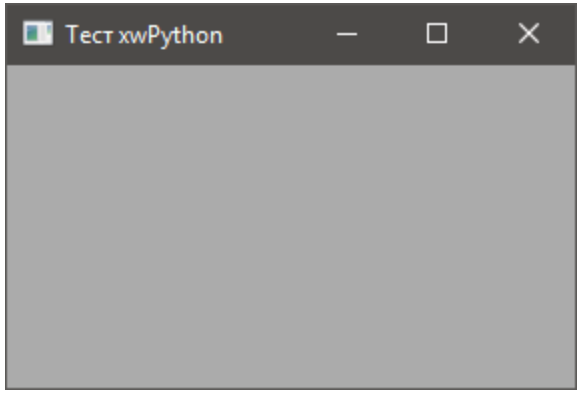



```cpp
#include <QApplication>
#include <QWidget>

int main(int argc, char *argv[]) {
  QApplication app(argc, argv);

  QWidget *win = new QWidget;
  win->resize(250, 150);
  win->move(300, 300);
  win->setWindowTitle("Simple");
  win->show();

  return app.exec();
}
```

Модули в Python: wxPython




```
import wx

class Example(wx.Frame):
    def __init__(self, title):
        super(Example, self).__init__(None, title=title,
                                      size=(300, 200))
        self.Move((800, 250))

def  main():
    app = wx.App()
    ex = Example(title='Тест xwPython')
    ex.Show()
    app.MainLoop()

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



Модули в Python: MyHDL (1)

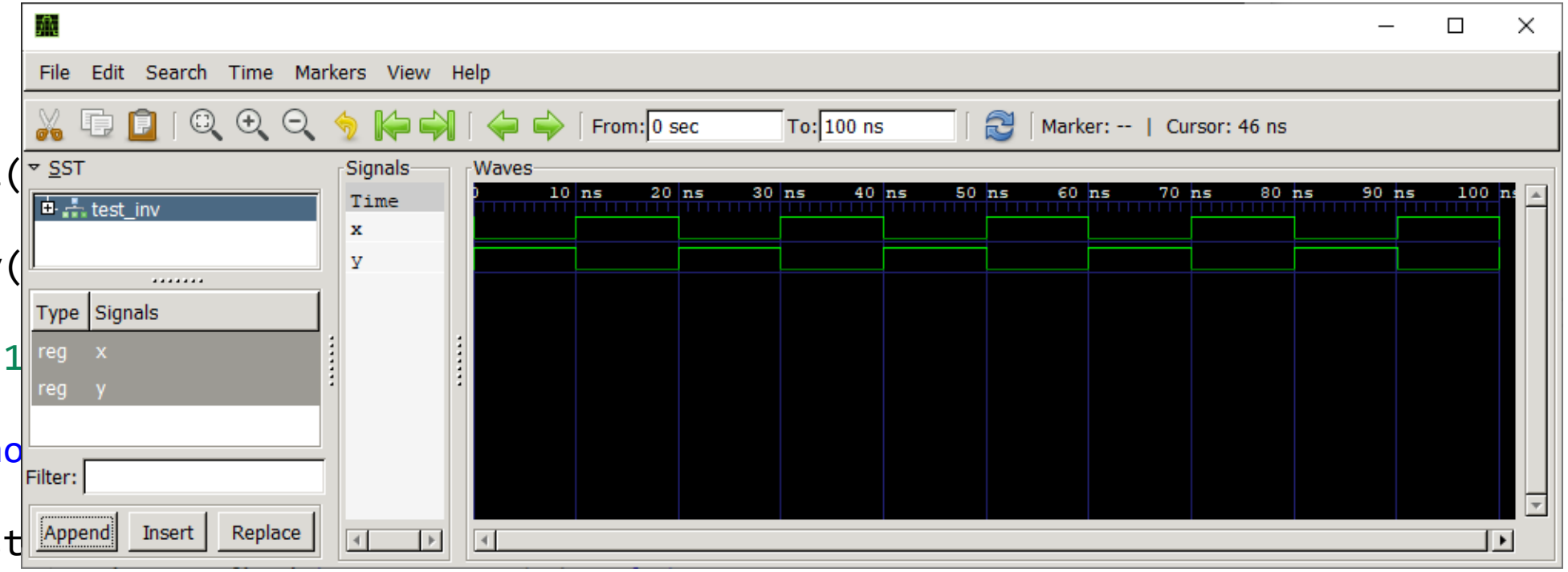
```
module inv(x, y);

  input  x;
  output y;

  always@(x)
    y = ~x;

endmodule
```

```
entity inv is
  port(x: in  bit;
       y: out bit);
end inv;

architecture BEH of inv is
begin
  process(x)
  begin
    y <= not x;
  end process;
end RTL;
```

```
from myhdl import *

@block
def inv(x, y):

    @always(x)
    def logic():
        y.next = not x

    return logic
```

Модули в Python: MyHDL (2)

```
@block
def test_inv():

    x, y = [Signal(

    inv_inst = inv(

    @always(delay(1
    def gen_x():
        x.next = no

    return inv_inst
```




```
def simulate(timesteps):
    simInst = test_inv()
    simInst.config_sim(trace=True, tracebackup=False)
    simInst.run_sim(timesteps, quiet=0)

simulate(100)
```