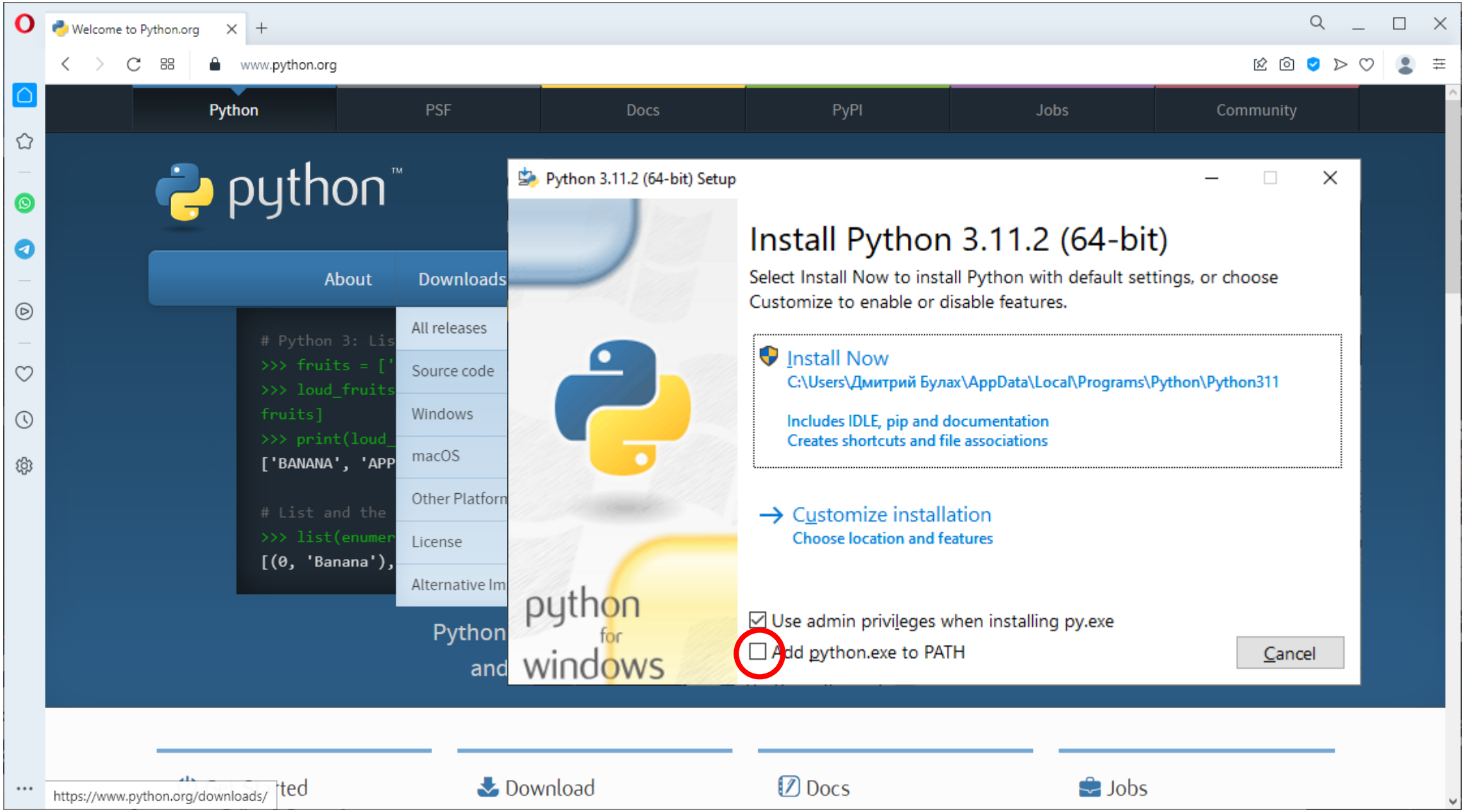
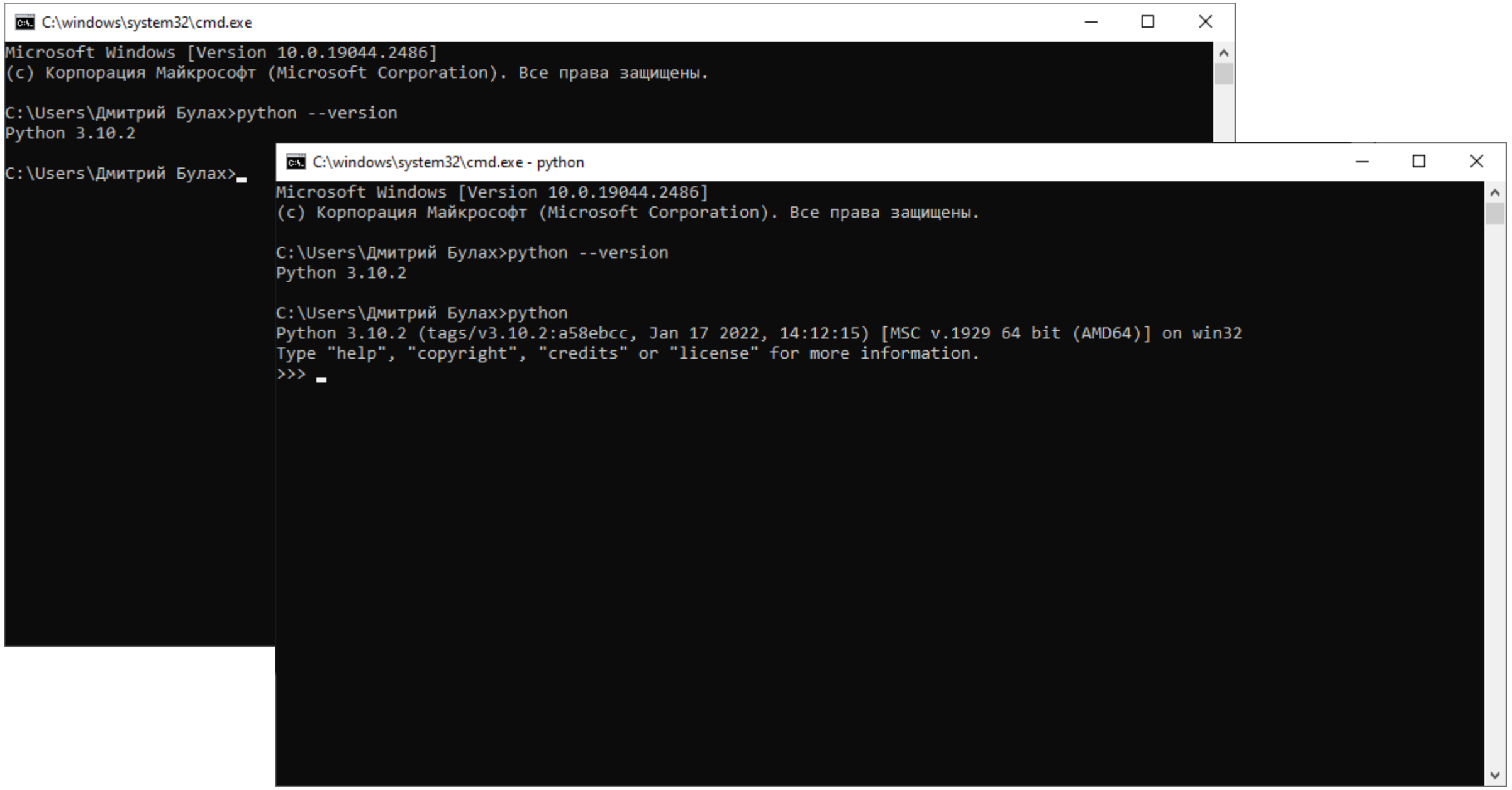
Разработка графических интерфейсов

Лекция 1

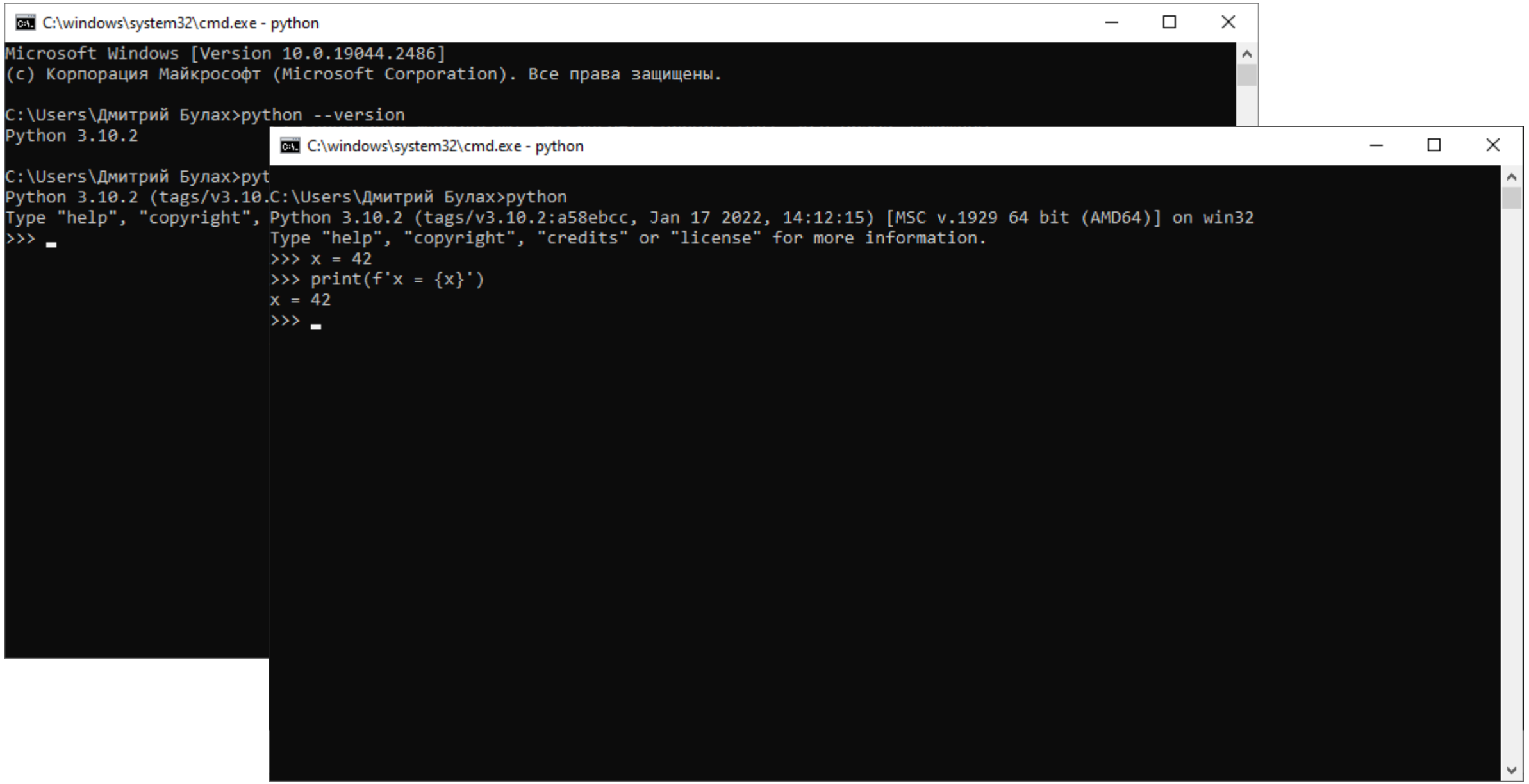
Установка Python



Запуск Python

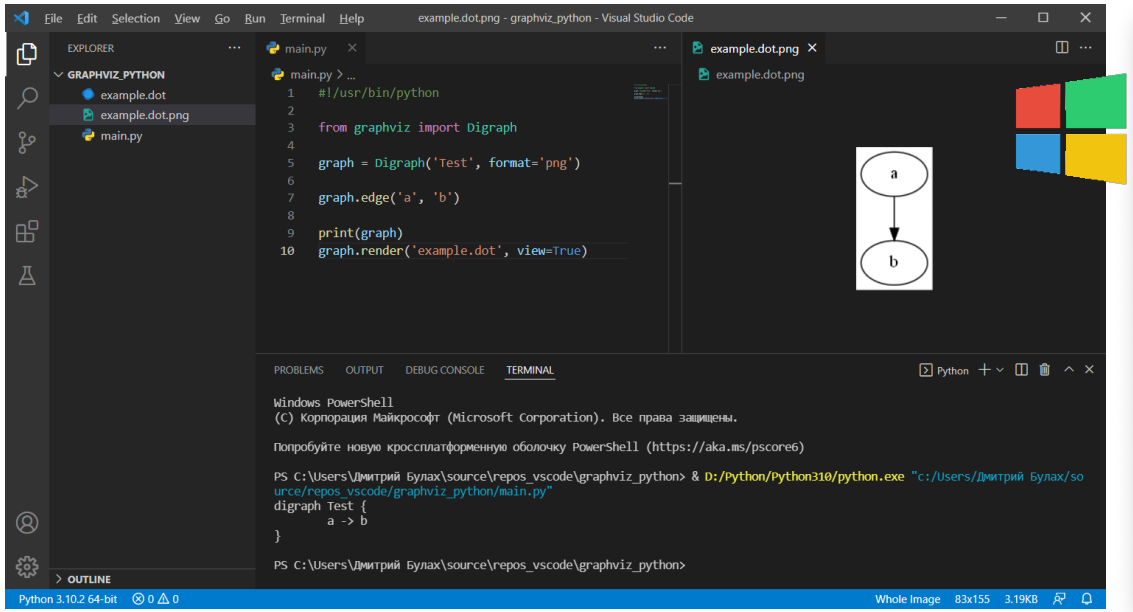


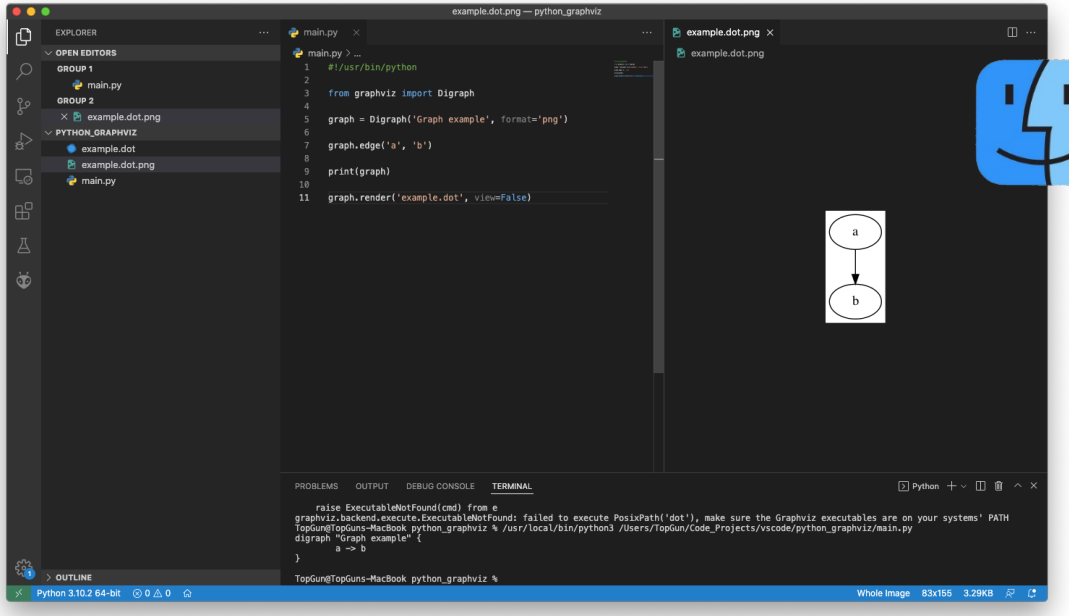
Написание кода в Python (1)

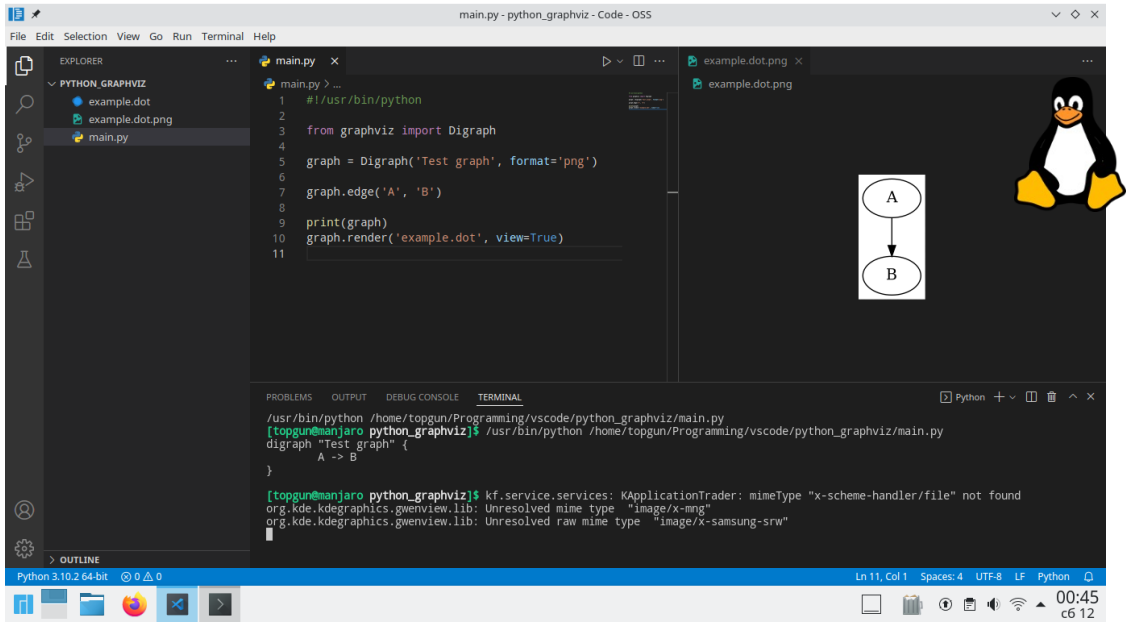


Написание кода в Python: Visual Studio Code

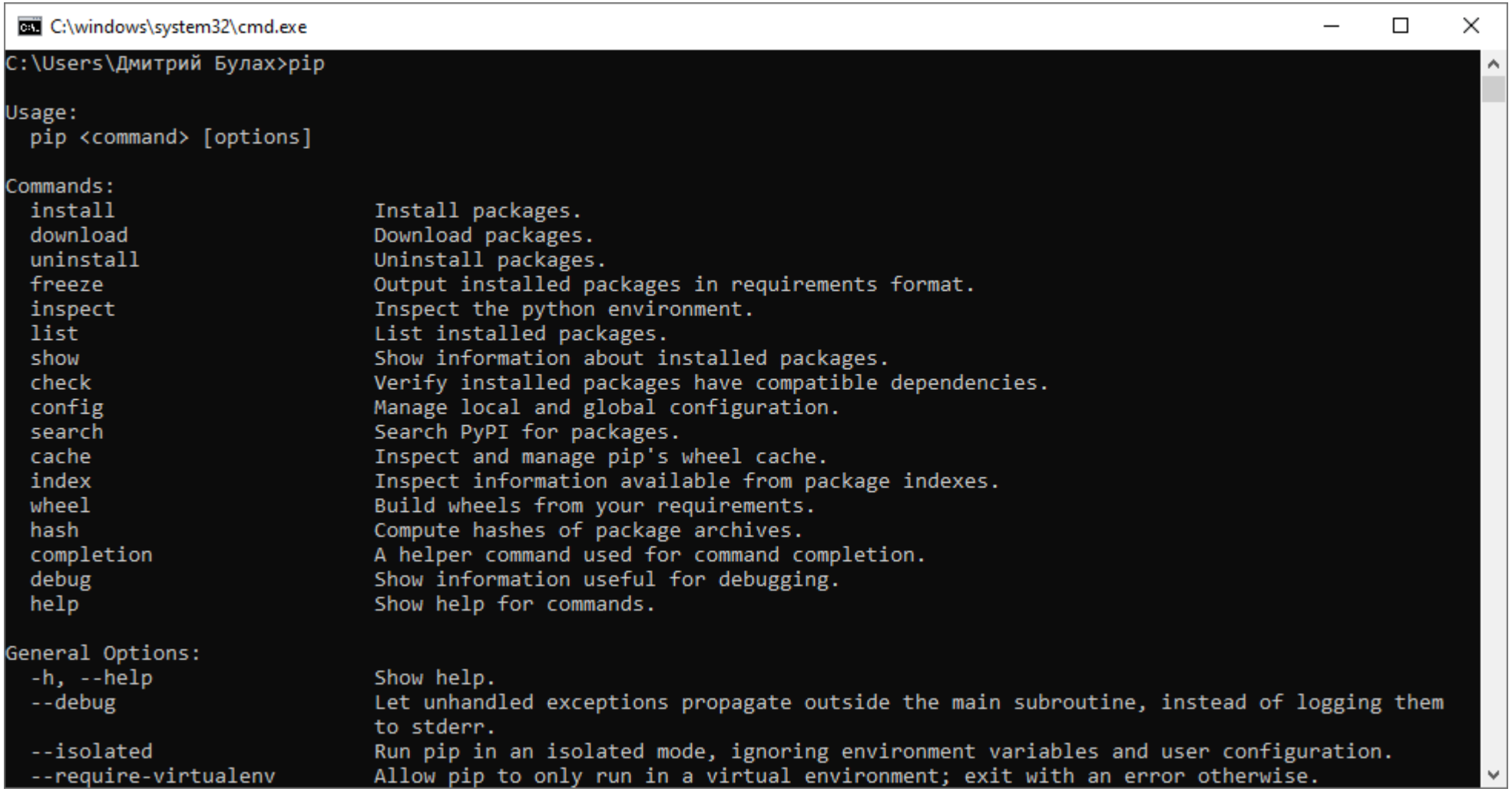
Visual Studio Code







Package Installer for Python (pip)

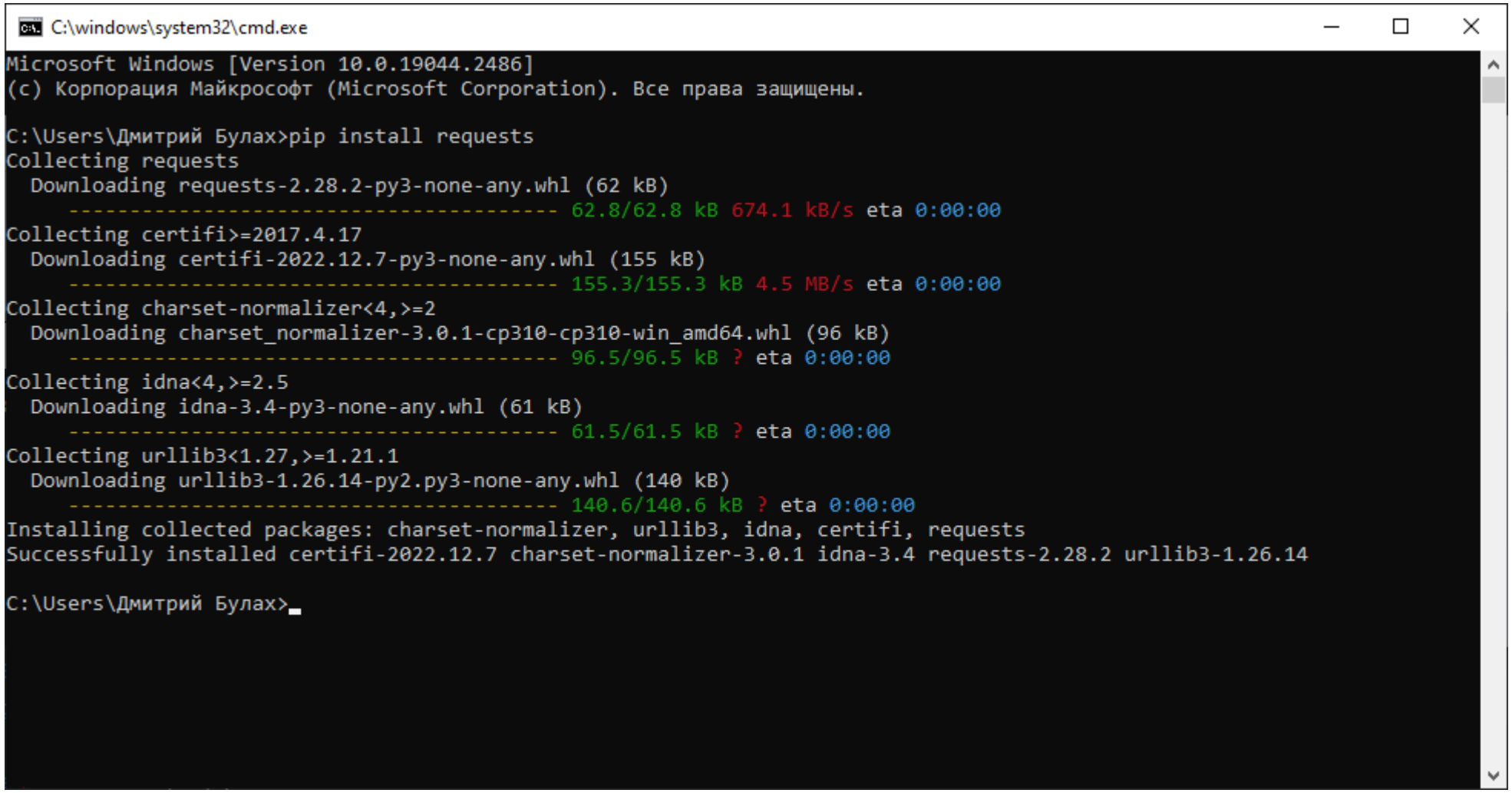


pip list: получение списка пакетов

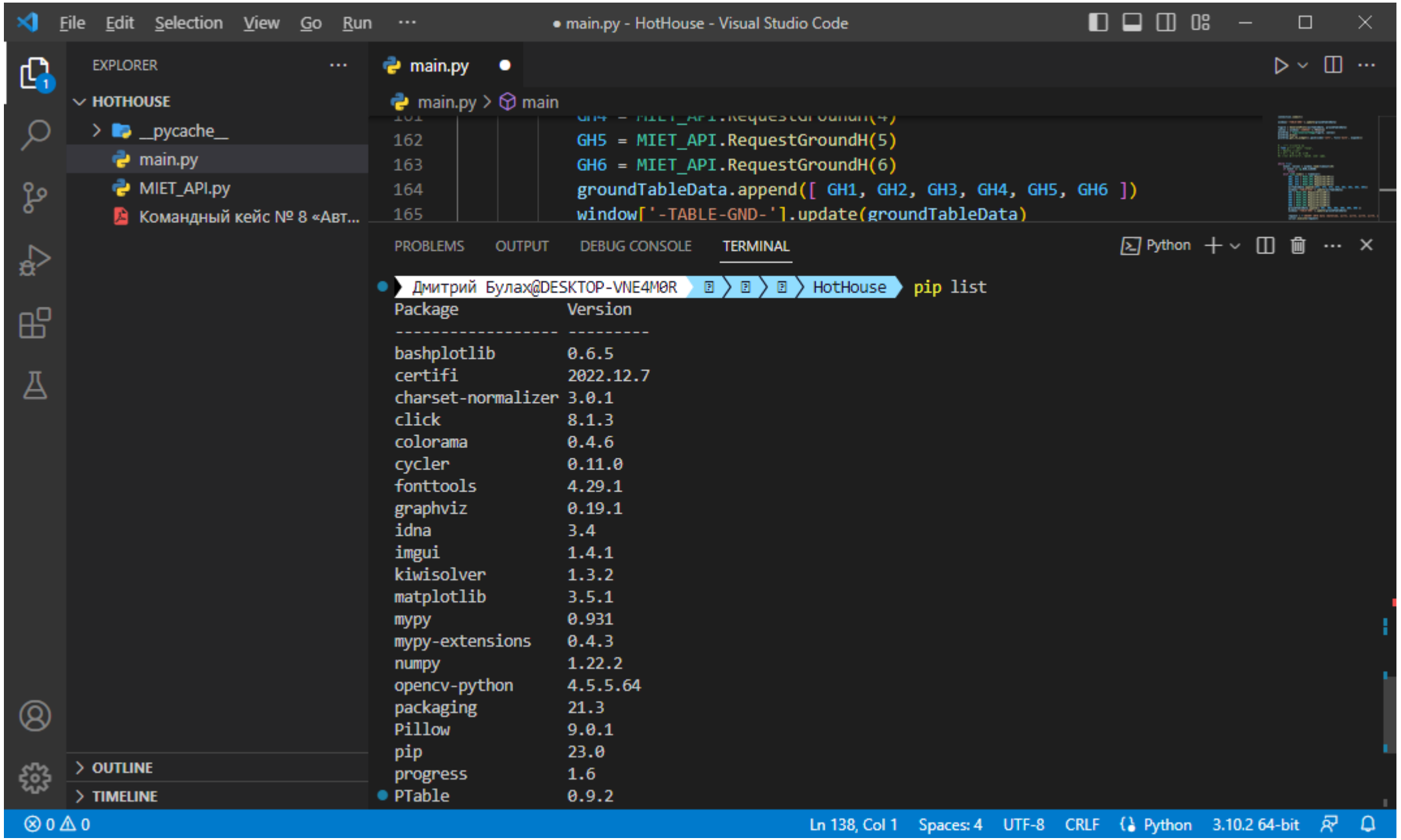
```
C:\Users\Дмитрий Булах>pip list
Package            Version
------------------ ---------
bashplotlib        0.6.5
certifi            2022.12.7
charset-normalizer 3.0.1
click              8.1.3
colorama           0.4.6
cycler             0.11.0
fonttools          4.29.1
graphviz           0.19.1
idna               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>, ...]
```



Работа с pip из Visual Studio Code



pip freeze: получение списка пакетов в формате «requirements»

```
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
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
six==1.16.0
tk==0.1.0
tomli==2.0.1
typing_extensions==4.1.1
urllib3==1.26.14
```

Синтаксис языка Python: вывод данных

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

```
x = 4
y = 2

print('x = {}, y = {}'.format(4, 2))


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


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

Синтаксис языка Python: ввод данных

```
x = input()
print(x)
```

```
x = input()
y = input()
print(x + y)
```

```
Дмитрий Булах@DESKTOP-VNE4M0R lab_python> & python.exe test_code.py
4
2
42
Дмитрий Булах@DESKTOP-VNE4M0R lab_python>
```

Синтаксис языка Python: приведение типов данных

Функции приведения типов:

```
str()
int()
long()
float()
```

```
x = input()
y = input()
print(int(x) + int(y))
```

```
Дмитрий Булах@DESKTOP-VNE4M0R lab_python> & python.exe test_code.py
4
2
6
Дмитрий Булах@DESKTOP-VNE4M0R lab_python>
```

```
x = input()
y = input()
print(int(x) + y)
```

```
Дмитрий Булах@DESKTOP-VNE4M0R & python.exe test_code.py
4
2
Traceback (most recent call last):
  File "test_code.py", line 4, in <module>
    print(int(x) + (y))
TypeError: unsupported operand type(s) for +: 'int' and 'str'
Дмитрий Булах@DESKTOP-VNE4M0R lab_python >
```

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'>
```

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', ','}
```

Изменяемый тип данных

Списки:

```
lst = []

lst.append(1)
lst.append(2)
lst.append(3)
```

Словари:

```
dict = {'a' : 1, 'b': 2, 'c': 5}

dict['d'] = 4;
```

Множества:

```
s = set()
s.add(1)
s.add(2)
s.add(1)
s.add(3)
s.add(2)
```

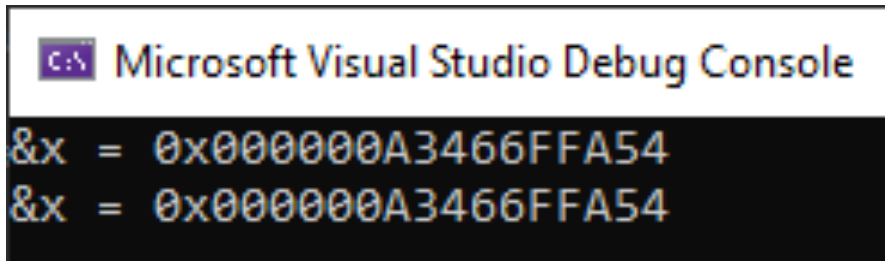
Неизменяемый тип данных

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

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

```python
x = 4
print(id(x))      140715232060296

x = 5
print(id(x))      140715232060328

y = 4
print(id(y))      140715232060296
```

- неизменяемые (числа, строки и кортежи)

```c
#include <cstdio>
#include <cstdlib>

int main() {
  int x = 4;
  printf("&x = 0x%p\n", &x);
  x = 5;
  printf("&x = 0x%p\n", &x);
  return EXIT_SUCCESS;
}
```



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)
```

Синтаксис языка Python: функции (2)

```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']))



AH1, AT1 = RequestAirHT(1)
```

Декораторы Python (1)

```
class Rectangle:
    def __init__(self, a, b):
        self.a = a
        self.b = b


    def area(self):
        return self.a * self.b

rect = Rectangle(5, 6)
print(rect.area())
```

```
class Rectangle:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    @property
    def area(self):
        return self.a * self.b

rect = Rectangle(5, 6)
print(rect.area)
```

```
print(rect.area)
```

```
<bound method Rectangle.area of <__main__.Rectangle object at 0x000022E074236D0>>
```

Декораторы Python (2)

```
def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print(f'Оборачиваемая функция: {func}')
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper
```

```
def hello_world():
     print('Hello world!')

hello_world()
```

```
@decorator_function
def hello_world():
     print('Hello world!')

hello_world()
```

```
Hello world!
```

```
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x000001D7E1CC89A0>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки
```

Документирование программного кода в Python

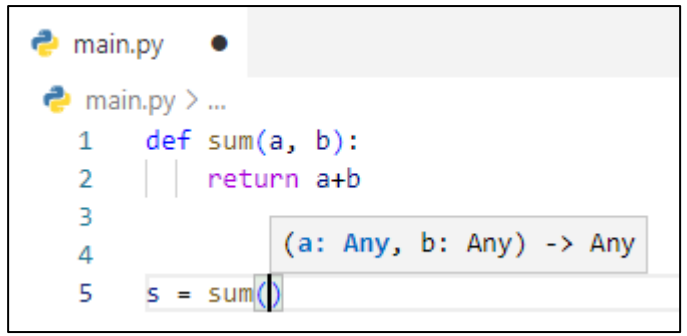
```python
def RequestAirHT(sensorId : int) -> Tuple[float, float]:
    """
    Функция посылает запрос типа GET для получения значений температуры и
    влажности воздуха
    URL для запроса: https://dt.miet.ru/ppo_it/api/temp_hum/<number>
    Вход : (int) номер датчика в диапазоне [1..4]
    Выход: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

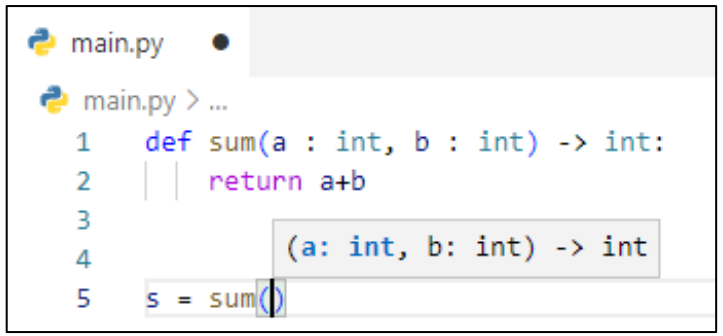
```python
print(help(RequestAirHT))
```

```
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

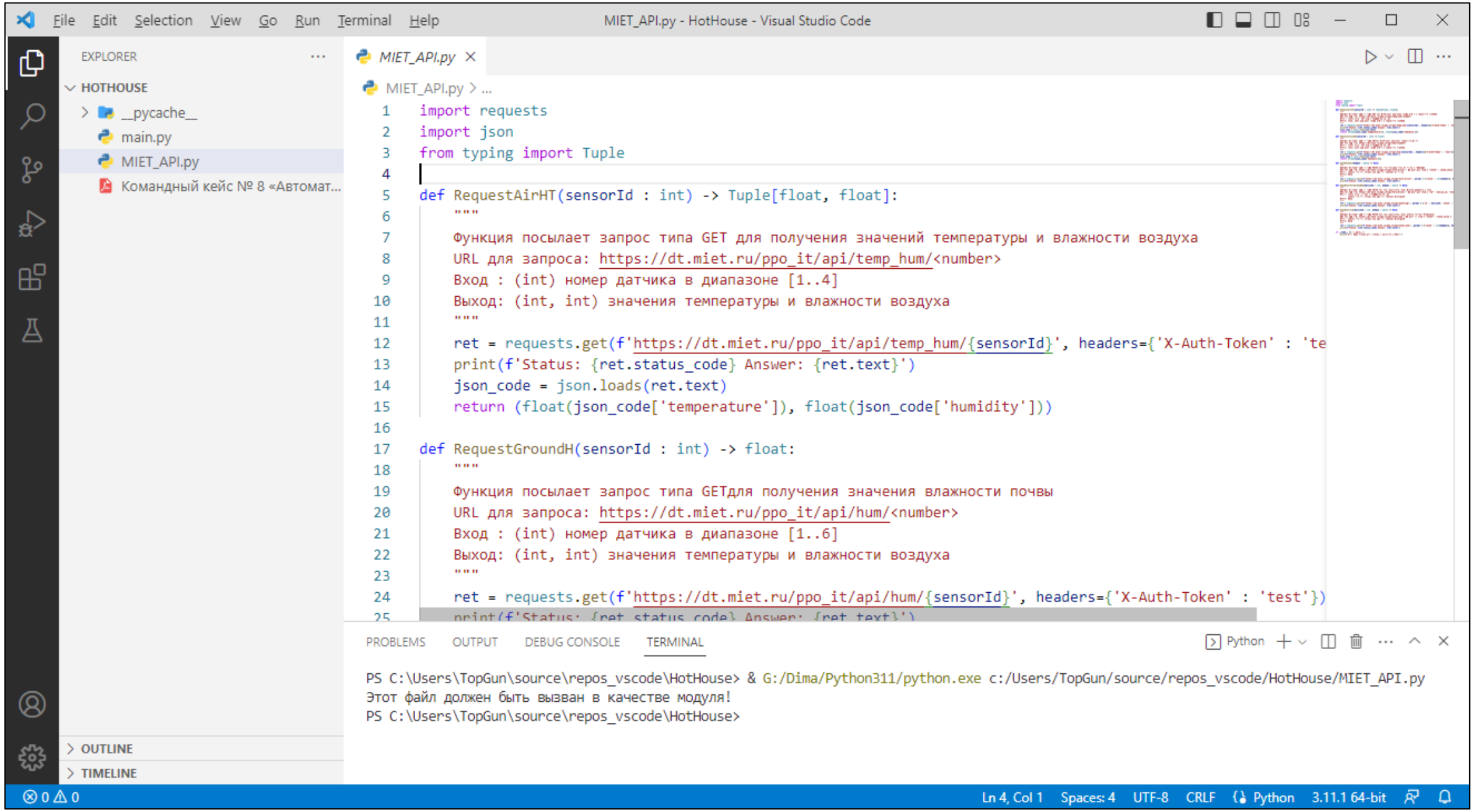
```python
def RequestGroundH(sensorId : int) -> float:
    """
    Функция посылает запрос типа GETдля получения значения влажности почвы
    URL для запроса: https://dt.miet.ru/ppo_it/api/hum/<number>
    Вход : (int) номер датчика в диапазоне [1..6]
    Выход: (int, int) значения температуры и влажности 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']))
```



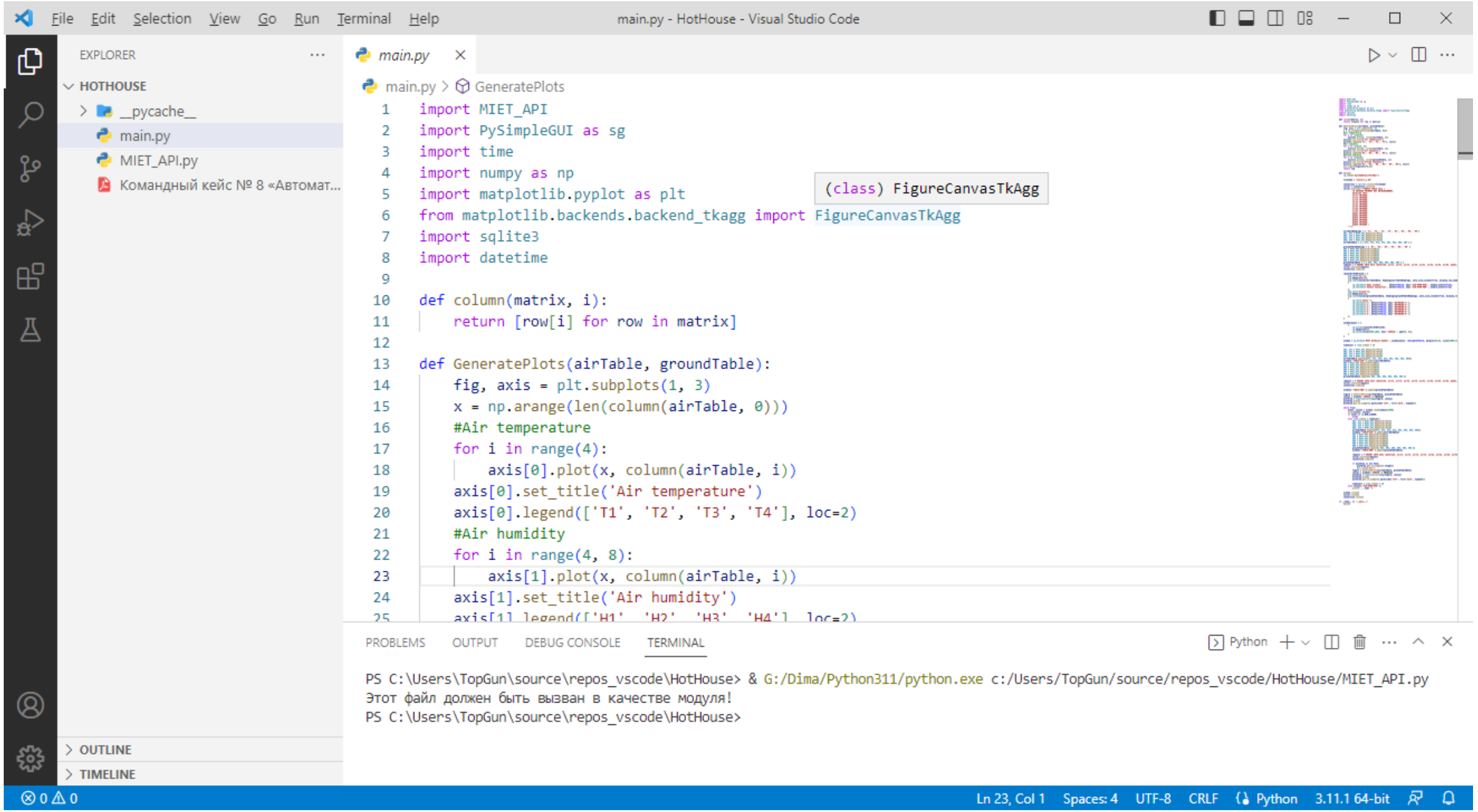



```python
s = sum('a', 'b')
```

Синтаксис языка Python: модули (1)



Синтаксис языка Python: модули (2)



Синтаксис языка Python: модули (3)

```
if __name__ == "__main__":
    print('Этот файл должен быть вызван в качестве модуля!')
```

```
import MIET_API

def main():
    filename = "sensors_1.db"

    connection = sqlite3.connect(filename)
    cursor = connection.cursor()
    cursor.execute("""CREATE TABLE data (
    ...
```

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

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))
```

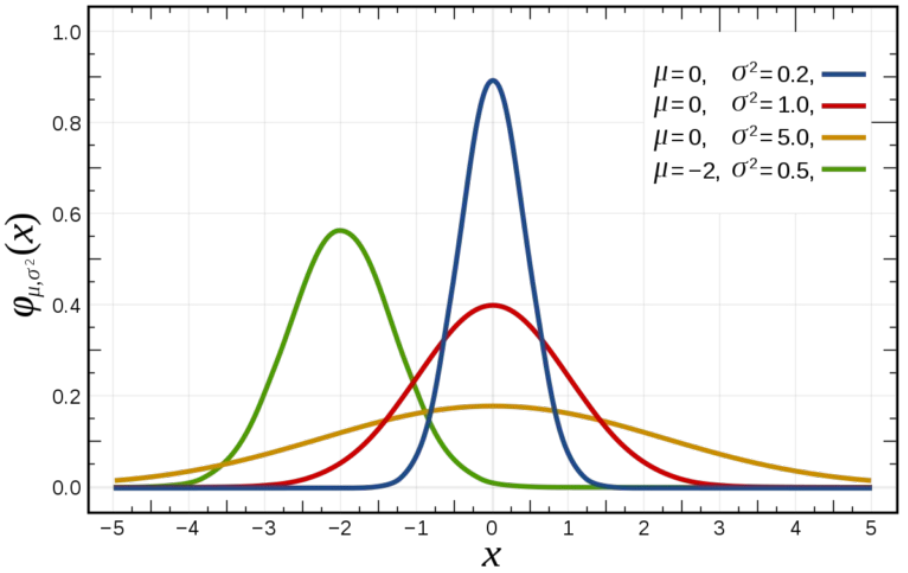
Модули в Python: random (1)

```python
import random

random.seed(4132)

print(f'random  = {random.random()}')

print(f'randint = {random.randint(0, 1000)}')

print(f'uniform = {random.uniform(4, 7)}')

print(f'range   = {random.randrange(5, 27, 3)}')

lst = [1, 2, 3, 4, 5]
random.shuffle(lst)
print(f'shuffe  = {lst}')

print(f'choice  = {random.choice(lst)}')

print(f'choises = {random.choices(lst, weights=None, k=2)}')
```

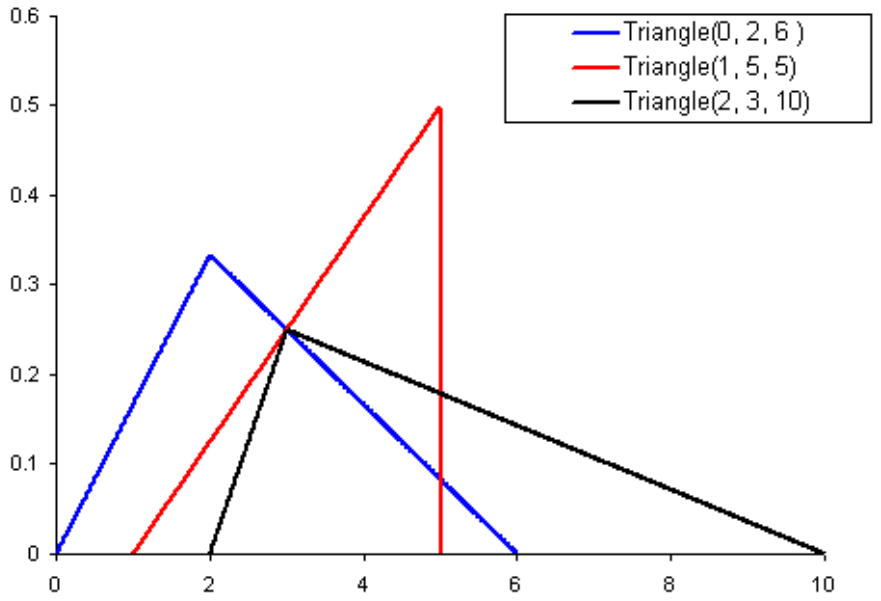
```
random  = 0.30167026468361413
randint = 213
uniform = 6.930483815609997
range   = 11
shuffe  = [3, 4, 2, 5, 1]
choice  = 2
choises = [4, 3]
```

Модули в Python: random (2)

```
random.gauss(mu, sigma)
```




```
random.triangular(low, high, mode)
```



Модули в Python: numpy


```
import numpy as np
```

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

Создание массива по диапазону:

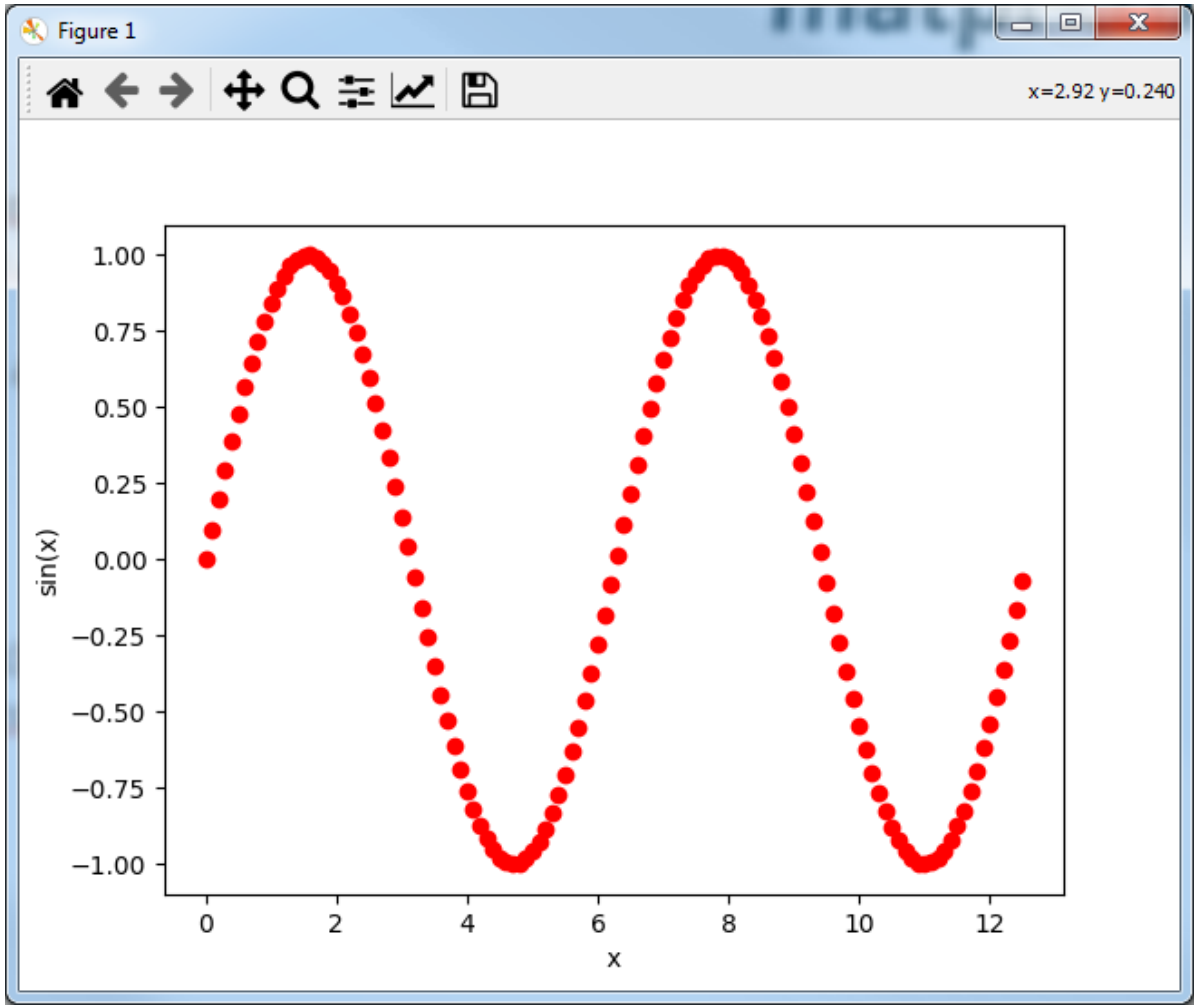
```
a = np.arange(0, 10, 1)
a = np.arange(0, 1, 0.1)
```

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: cython (1)

code_python.py:

```
def test(x):
    y = 1
    for i in range(1, x+1):
        y += i
    return y
```

code_cython.py:

```
cpdef int test(int x):
    cdef int y = 1
    cdef int i
    for i in range(1, x+1):
        y += i
    return y
```

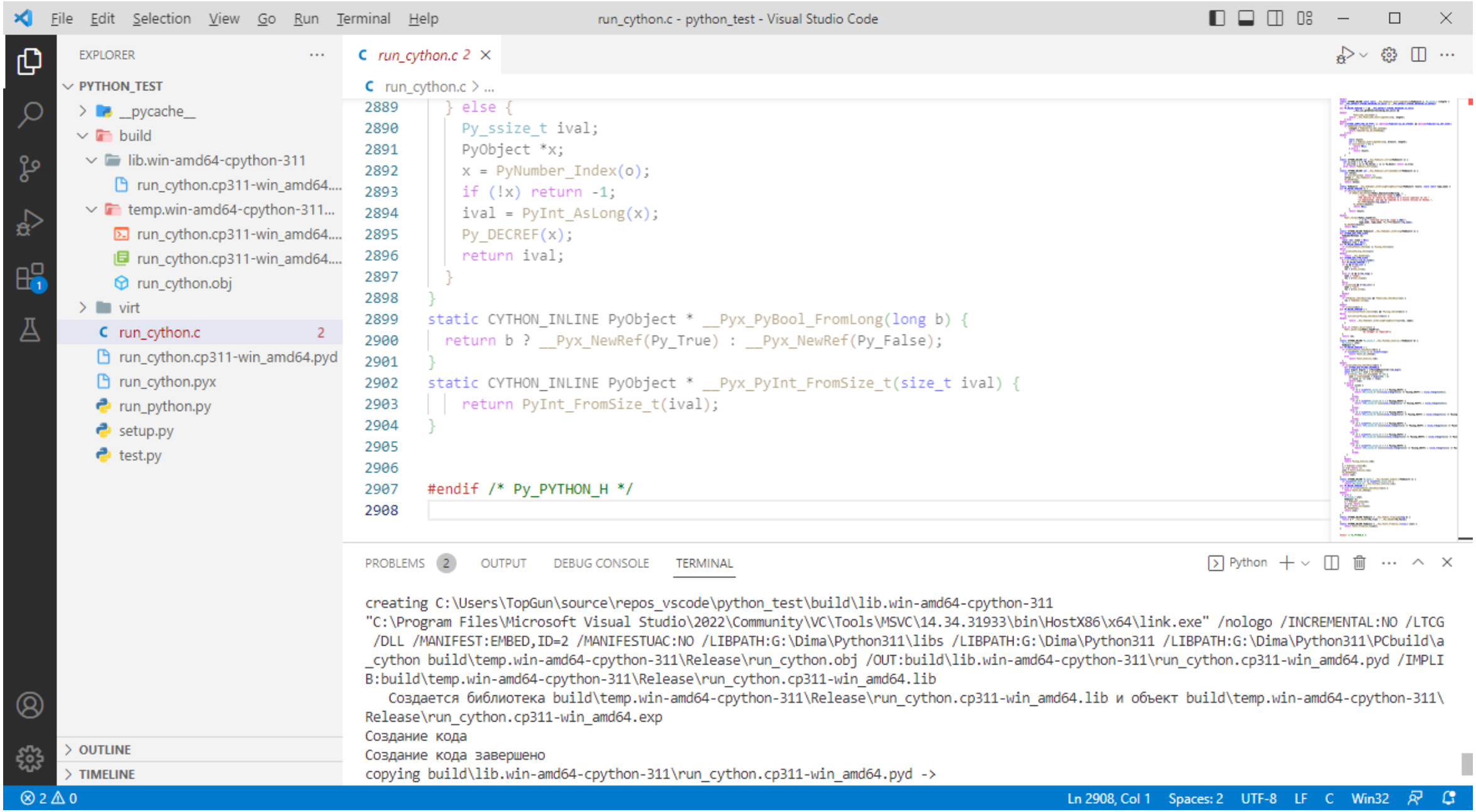
Компиляция кода:

```
from distutils.core import setup
from Cython.Build import cythonize

setup(ext_modules = cythonize(['*.pyx']))
```

```
python setup.py build_ext --inplace
```

*https://habr.com/ru/company/ruvds/blog/462487/

Модули в Python: cython (2)



Модули в Python: cython (3)

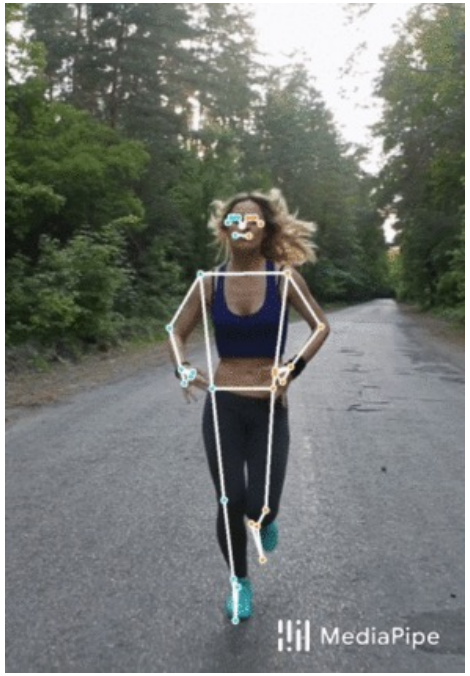
test.py:

```
import run_python
import run_cython
import time

number = 10_000_000

start = time.time()
run_python.test(number)
end =  time.time()

print("Python time = {}".format(end - start))

start = time.time()
run_cython.test(number)
end =  time.time()

print("Cython time = {}".format(end - start))

print("Speedup = {}".format(py_time / cy_time))
```

```
Python time = 0.7201635837554932
Cython time = 0.0028400421142578125
Speedup = 253.5749664204164
```

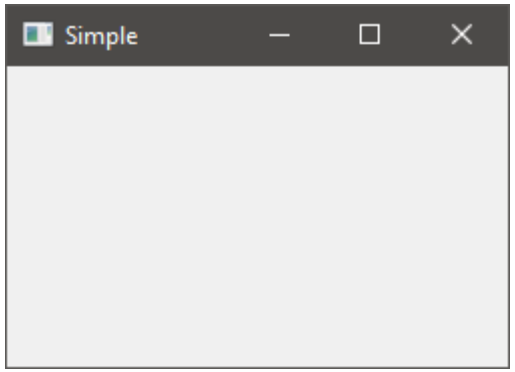
Модули в Python: mediapipe

MediaPipe



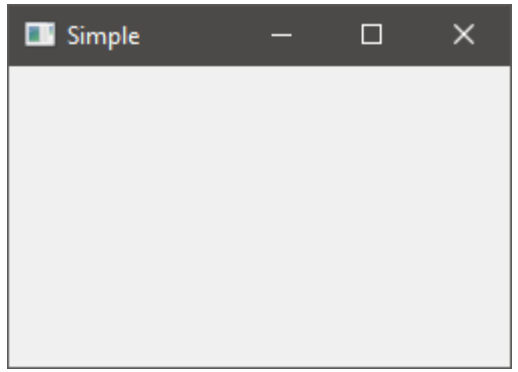
Модули в Python: PyQt5 (1)

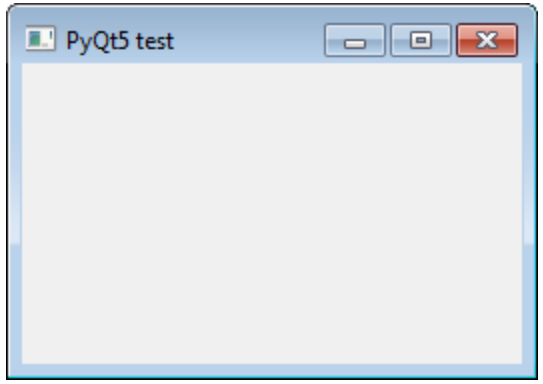

```python
import sys
from PyQt5.QtWidgets import QApplication, QWidget

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_())
```

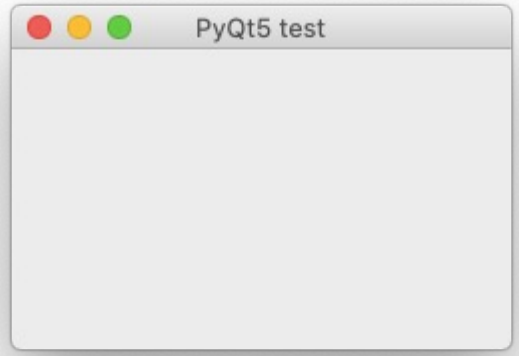



```cpp
#include <QApplication>
#include <QWidget>

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();
}
```

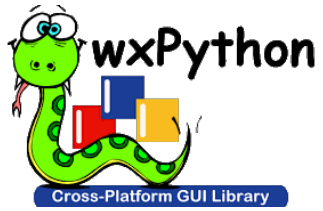
Модули в Python: PyQt5 (2)







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()
```