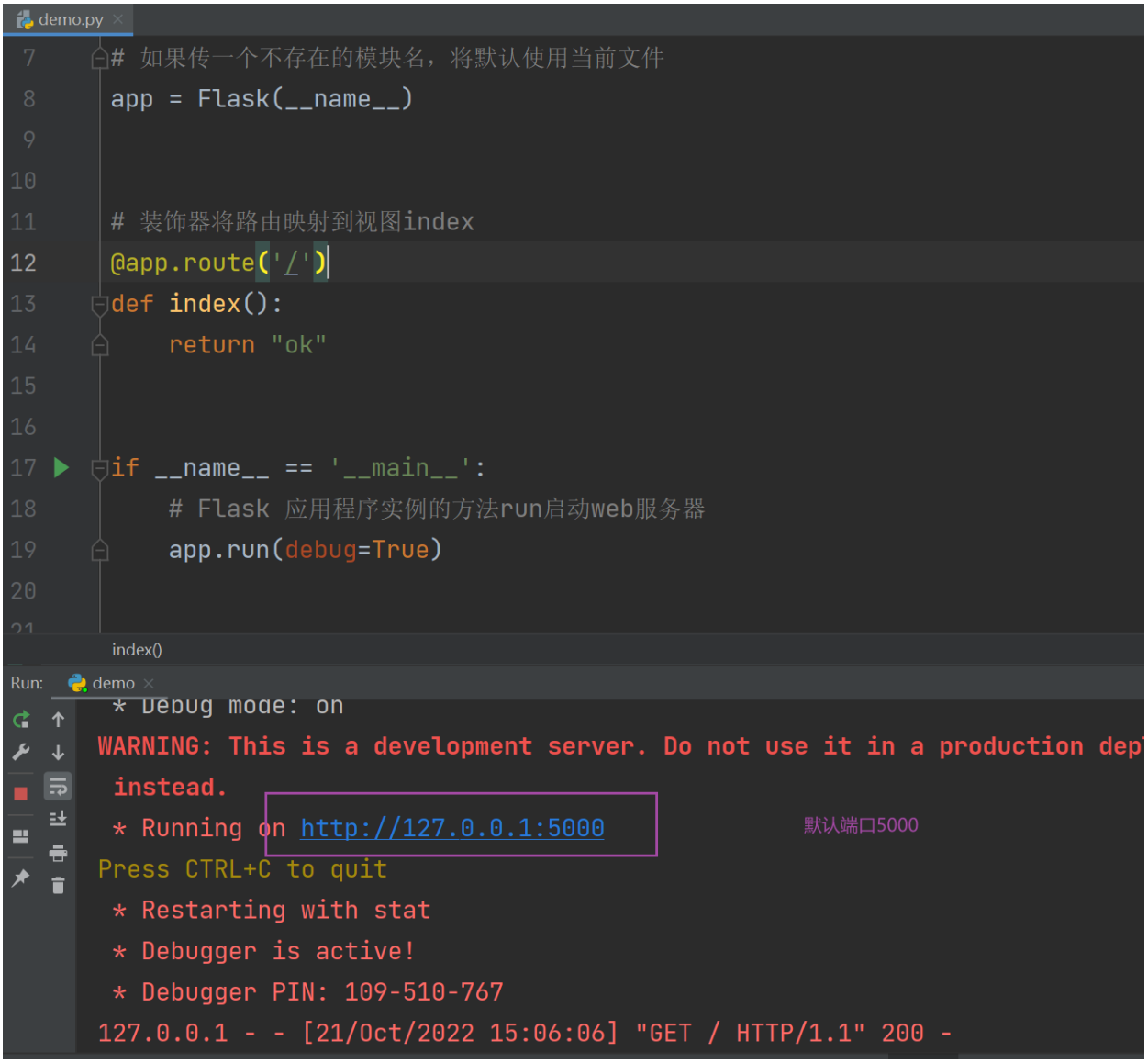
1.创建Flask工程

创建Flask项目没有命令快捷方式，只需要在项目文件夹下创建一个普通的py文件，导入Flask实例化一个Flask应用程序。

```
# coding=utf-8
# 导入Flask类
from flask import Flask

# Flask 接收一个参数 name ，当前模块的文件名
# Flask 在查找静态文件，或者模板时候默认以当前文件所在的目录去查找
# 如果传一个不存在的模块名，将默认使用当前文件
app = Flask(__name__)

# 装饰器将路由映射到视图index
@app.route('/')
def index():
    return "ok"


if __name__ == '__main__':
    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```


demo.py
7 # 如果传一个不存在的模块名，将默认使用当前文件
8 app = Flask(__name__)
9
10
11 # 装饰器将路由映射到视图index
12 @app.route('/')
13 def index():
14 return "ok"
15
16
17 if __name__ == '__main__':
18 # Flask 应用程序实例的方法run启动web服务器
19 app.run(debug=True)
20
index()
Run: demo
* Debug mode: on
WARNING: This is a development server. Do not use it in a production dep
instead.
* Running on http://127.0.0.1:5000
默认端口5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 109-510-767
127.0.0.1 - - [21/Oct/2022 15:06:06] "GET / HTTP/1.1" 200 -

ok

必须在项目中导入Flask模块。

Flask类的一个对象是我们的WSGI应用程序。

Flask构造函数使用当前模块（__name __）的名称作为参数。

Flask类的route()函数是一个装饰器，它告诉应用程序哪个URL应该调用相关的函数。

```
--- app.route(rule, options)
---     rule 参数表示与该函数的URL绑定。
---     options 是要转发给基础Rule对象的参数列表。

--- 在上面的示例中，'/ ' URL与index()函数绑定。
--- 因此，当在浏览器中打开web服务器的主页时，将呈现该函数的输出。
```

最后，Flask类的run()方法在本地开发服务器上运行应用程序。

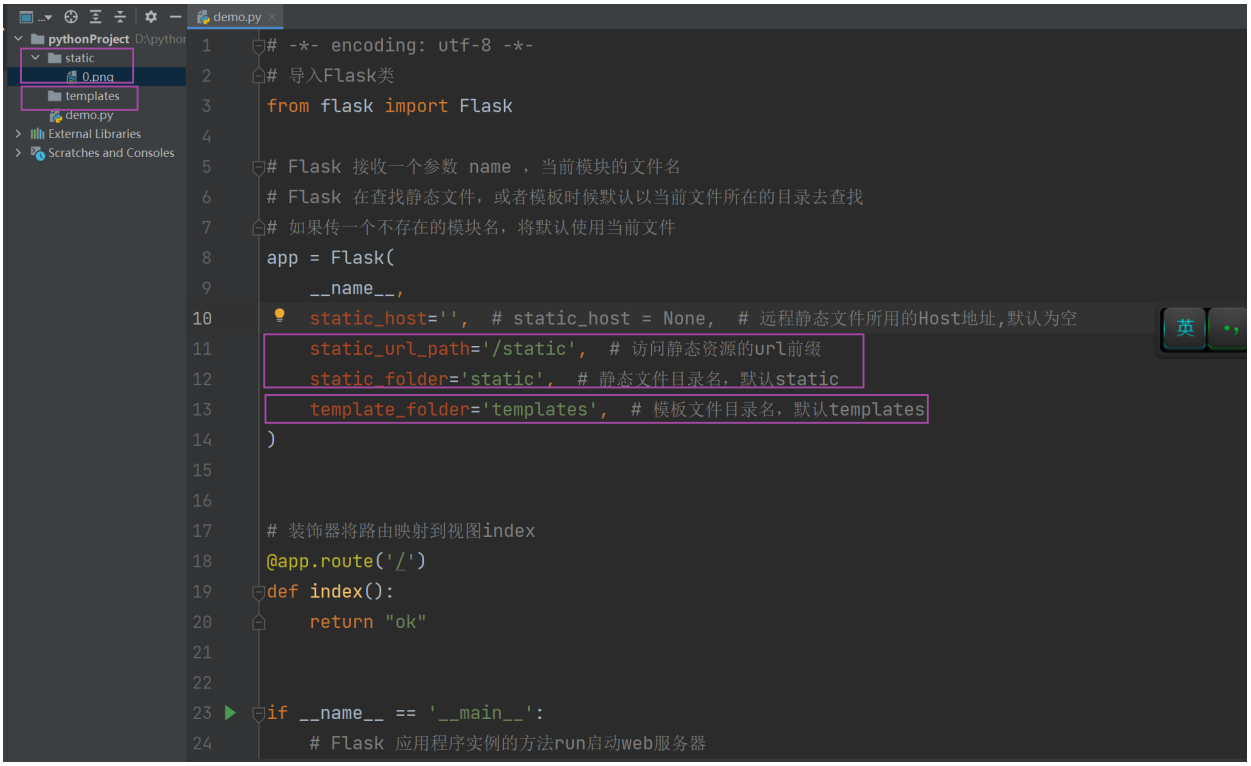
```
app.run(host, port, debug, options)
---     host
            要监听的主机名。 默认为127.0.0.1（localhost）。设置为“0.0.0.0”以使服务器在外部可用
---     port
            默认值为5000
---     debug
        默认为false。 如果设置为true，则提供调试信息
```

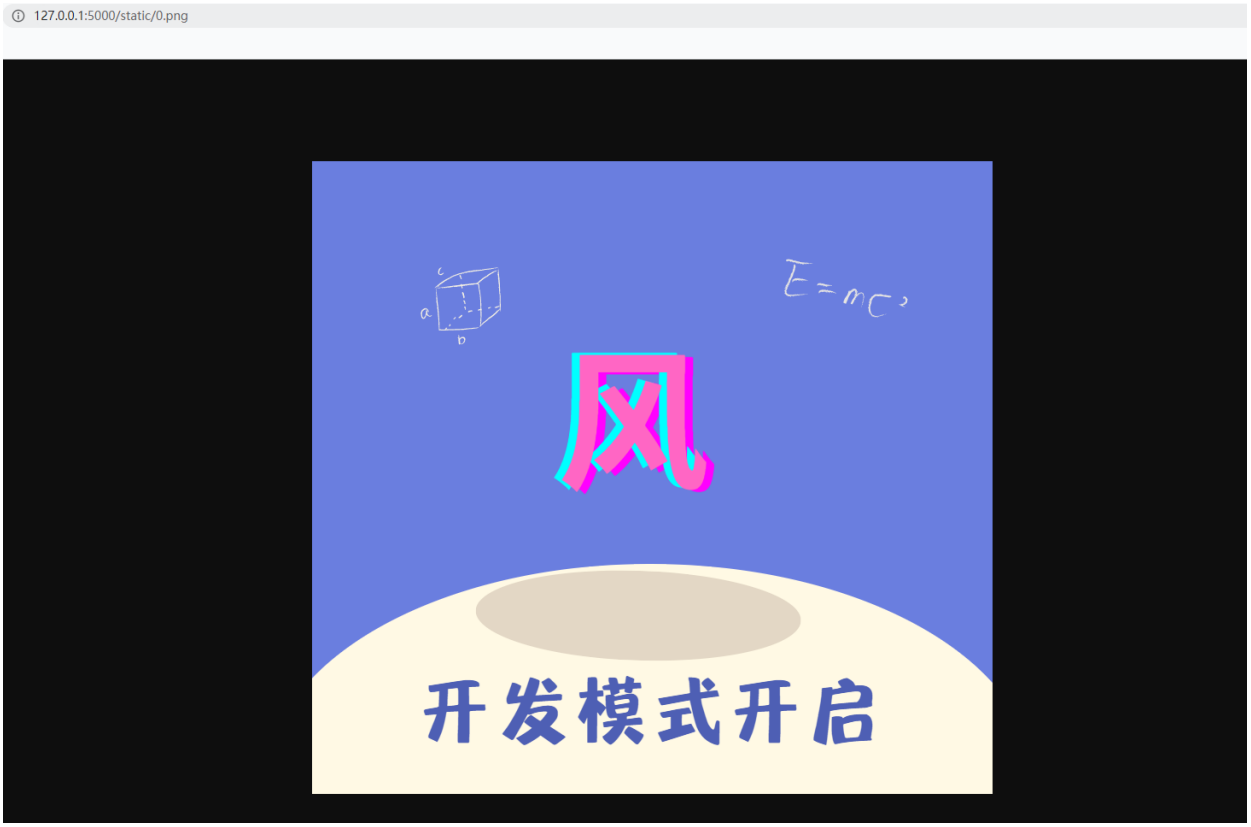
```
---     options
        要转发到底层的Werkzeug服务器(请求响应)。
```

2.Flask初始化参数

```
import_name: 导包的目录
static_path: 访问静态资源的路---已淘汰
static_url_path: 访问静态文件的url前缀
static_folder: 默认‘static’
template_folder: 默认‘templates’
```

实例化Flask对象之后，静态文件默认在Flask第一个参数指定的模块所在的目录下，静态文件使用**static**目录，模板使用templates目录。





3.配置

django项目中有一个setting.py的配置文件，但是Flask中没有，需要自己定义。
常见的两种方式：

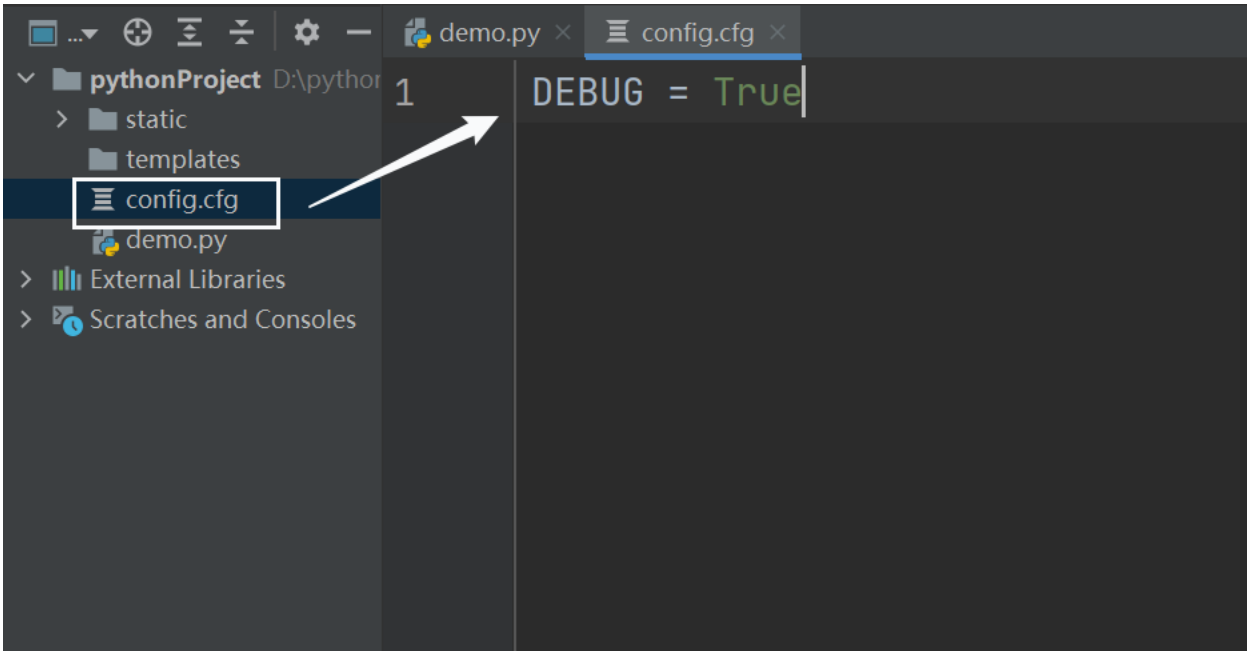
```
--- 从配置文件中读取配置
    app.config.from_pyfile('文件名')

--- 从类中读取配置信息
    app.config.from_object(类名)
```

Flask 默认是运行在线上模式，也就是说出错了是看不到错误堆栈信息的，为了好调试一般开发阶段会将调试打开。
django中默认开启调试DEBUG = True,Flask中开启调试同样只需要在配置中写入 DEBUG=True。

a.从配置文件中读取配置

注意文件类型cfg

大多数情况下，很多程序都要保存用户的设置，办法有很多：注册表，日志文件..... 而很多程序都使用了一个专用的文件。为了方便起见，常常命名为*.cfg，有时甚至直接命名为Config.cfg




```python
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask

# Flask 接收一个参数 name ，当前模块的文件名
# Flask 在查找静态文件，或者模板时候默认以当前文件所在的目录去查找
# 如果传一个不存在的模块名，将默认使用当前文件
app = Flask(
    __name__,
    static_host='',  # static_host = None,  # 远程静态文件所用的Host地址,默认为空
    static_url_path='/static',  # 访问静态资源的url前缀
    static_folder='static',  # 静态文件目录名，默认static
    template_folder='templates',  # 模板文件目录名，默认templates
)

# 设置配置信息获取方式，从配置文件中查找
app.config.from_pyfile('config.cfg')


# 装饰器将路由映射到视图index
@app.route('/')
def index():
    return "ok"


if __name__ == '__main__':
    # Flask 应用程序实例的方法run启动web服务器
    app.run()
```

注意：配置的变量名需要用大写。如果注册多个配置文件，后面注册的同名配置会覆盖掉前面的配置。一般使用一 个工程只用一个配置文件。

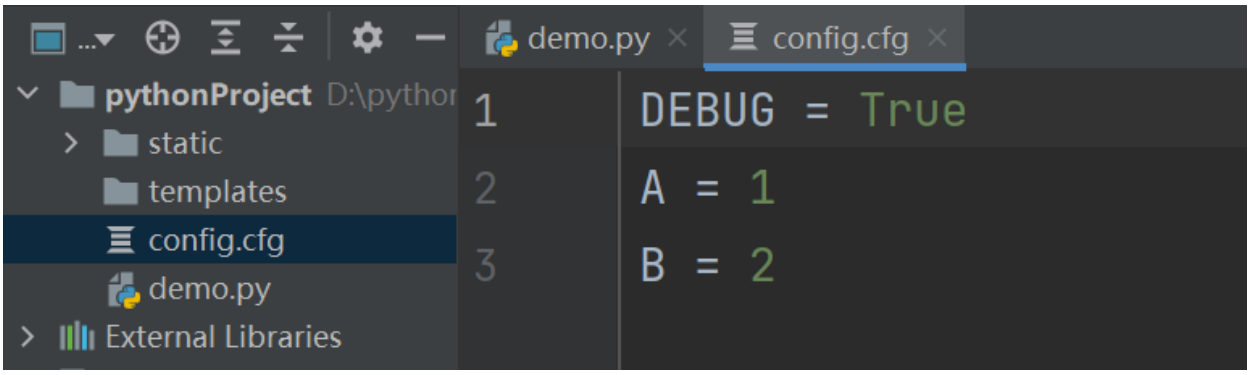
b.从类中读取配置信息

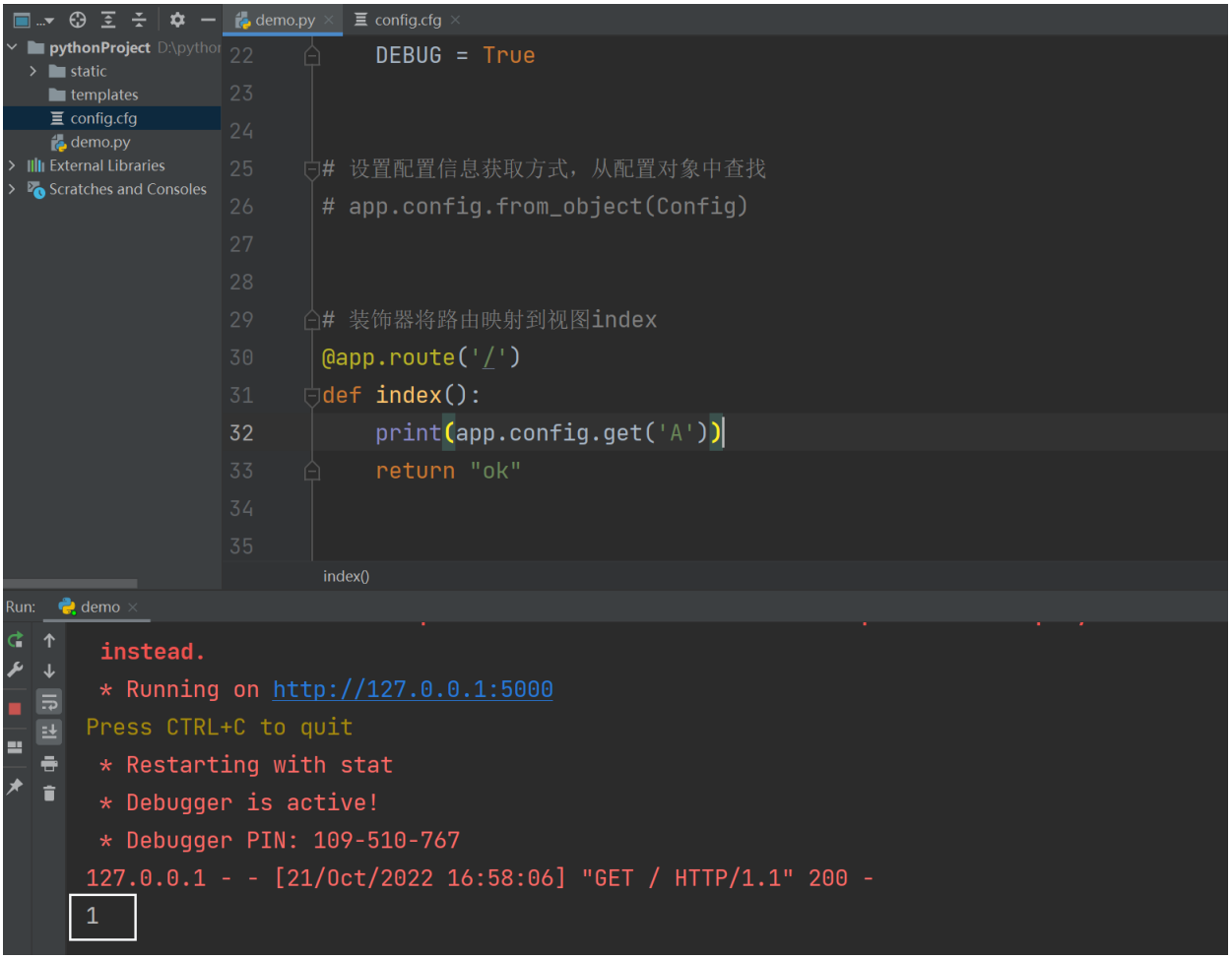
```python
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask

# Flask 接收一个参数 name ，当前模块的文件名
# Flask 在查找静态文件，或者模板时候默认以当前文件所在的目录去查找
# 如果传一个不存在的模块名，将默认使用当前文件
app = Flask(
    __name__,
    static_host='',  # static_host = None,  # 远程静态文件所用的Host地址,默认为空
    static_url_path='/static',  # 访问静态资源的url前缀
    static_folder='static',  # 静态文件目录名，默认static
    template_folder='templates',  # 模板文件目录名，默认templates
)

# 设置配置信息获取方式，从配置文件中查找
# app.config.from_pyfile('config.cfg')


# 配置类
class Config(object):
    DEBUG = True


# 设置配置信息获取方式，从配置对象中查找
app.config.from_object(Config)


# 装饰器将路由映射到视图index
@app.route('/')
def index():
    return "ok"


if __name__ == '__main__':
    # Flask 应用程序实例的方法run启动web服务器
    app.run()
```

4.读取配置参数

第一种方式：

在能访问到app对象的文件中可直接使用app.config.get方式获取配置参数 app.config.get('配置')

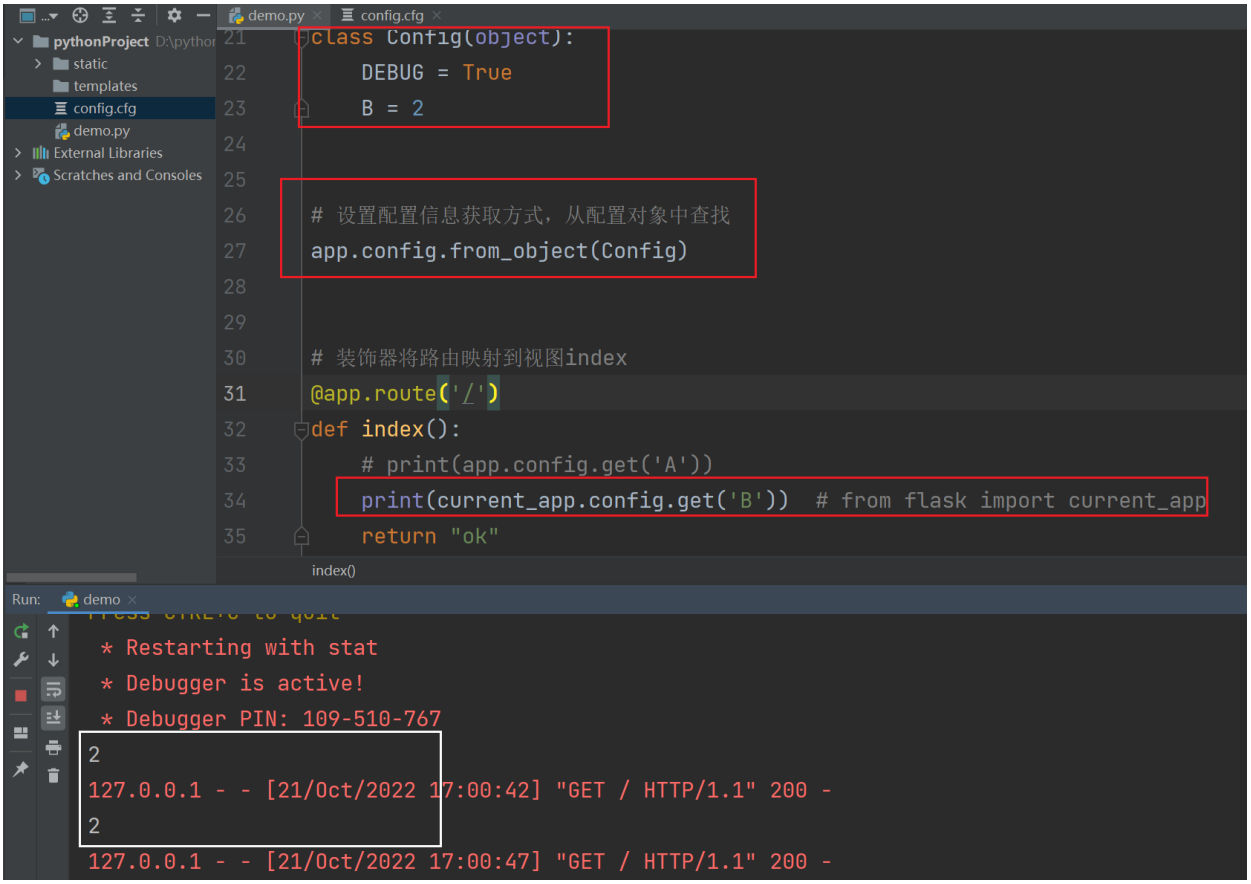




第二种方式：

需要从flask中导入current_app，在整个Flask项目中都可以使用，实际上也是app对象相到于一个别名。

```
current_app.config.get('配置名')
```



5.app.run参数

```
def run(
    self,
    host: t.Optional[str] = None,
    port: t.Optional[int] = None,
    debug: t.Optional[bool] = None,
    load_dotenv: bool = True,
    **options: t.Any,
) -> None:
    """Runs the application on a local development server.

    Do not use ``run()`` in a production setting. It is not intended to
    meet security and performance requirements for a production server.
    Instead, see :doc:`/deploying/index` for WSGI server recommendations.

    If the :attr:`debug` flag is set the server will automatically reload
    for code changes and show a debugger in case an exception happened.
```

app.py × config.cfg ×

```
        if not host:
            if sn_host:
                host = sn_host
            else:
                host = "127.0.0.1"

        if port or port == 0:
            port = int(port)
        elif sn_port:
            port = int(sn_port)
        else:
            port = 5000
```

```
    options.setdefault("use_reloader", self.debug)
    options.setdefault("use_debugger", self.debug)
    options.setdefault("threaded", True)

    cli.show_server_banner(self.debug, self.name)

    from werkzeug.serving import run_simple

    try:
        run_simple(t.cast(str, host), port, self, **options)
    finally:
        # reset the first request information if the development server
        # reset normally.  This makes it possible to restart the server
        # without reloader and that stuff from an interactive shell.
        self._got_first_request = False
```

从源码中可以看出run方法是启动一个werkzug工具箱提供的简易服务器。 参数：
    host 服务器主机地址，默认使用本机地址'127.0.0.1'
    port 端口号,默认5000

    debug 调试，参数是bool值 ,表示是否开启调试，True开启调试。默认Flase

```
app.run(host='0.0.0.0') # 这样可以监听本机所有ip地址
```