```
#Flask AppContext
#Request RequestContext


# ctx=app.app_context()
# ctx.push()
# #手动推入。。离线使用或者测试
#
# a=current_app
# d=current_app.config['DEBUG']
#
# ctx.pop()


#with:使用
#对实现了上下文协议的对象使用with
#上下文管理器
#__enter__,__exit__
#1.连数据库->sql->释放资源 try except finally:   python中用with
#文件读写用with  try: f=open(filename) finally:f.close
#with open('r',filename) as xxx: f.read()
#as后位__enter__对象副本
```

```
#主线程
#多线程最大化利用CPU
#异步编程
#多核并行执行程序.每个核心运行不同程序
#python无法充分运用多核优势
#python的GIL（全局解释器锁）=》为了线程安全
#wekzeug  local字典    实现线程隔离
new_t = threading.Thread(target=worker)
new_t.start()
t = threading.current_thread()
t1=threading.Thread(target=worker)
t1.start()
print(t.getName())
```

```
    print('thread')
    t=threading.current_thread()
    import time
    time.sleep(10)
    print(t.getName())


#主线程
#多线程最大化利用CPU
#异步编程
#多核并行执行程序.每个核心运行不同程序
#python无法充分运用多核优势
#python的GIL（全局解释器锁）=》为了线程安全
#wekzeug  local字典   实现线程隔离
new_t = threading.Thread(target=worker)
new_t.start()
t = threading.current_thread()
t1=threading.Thread(target=worker)
t1.start()
print(t.getName())
```

```
#IO线程 严重依赖CPU计算 CPU密集型

#IO密集型程序   查询数据库   请求网络资源 读写文件

#python不适合cpu密集型


#flask web框架

#请求线程

#IO 请求   flask开启多少个线程来处理请求

#webserver

#调试的时候是单进程，单线程；app.run(threaded)可以开启多线程模式
```

```python
#local使用字典实现线程隔离
#localstack线程隔离的栈结构
#local可以使用.方法直接使用

from werkzeug.local import Local


class A:
    b=1

myobj = Local()
myobj.b=1
def worker():
    #新线程
    myobj.b=2
    print(' in new thread b is myobj:'+str(myobj.b))

new_t = threading.Thread(target=worker,name='sss thread')
new_t.start()
```

```python
import threading


import time
from werkzeug.local import LocalStack
#push pop top
#local使用字典实现线程隔离
#localstack线程隔离的栈结构
#local可以使用.方法直接使用
#localstack线程隔离(每个线程一个栈)
my_stack = LocalStack()
my_stack.push(1)
print('in main thread after push  value:'+str(my_stack.top))
def worker():
    print('new thread before push:'+str(my_stack.top))
    my_stack.push(2)
    print('new Thread after push:',str(my_stack.top))
mythread=threading.Thread(target=worker)
mythread.start()
time.sleep(2)
```

```python
@contextmanager
def make_myresource():
    print('connect to resource')
    yield MyResource() #将其返回，实例化，然后调用实例方法
    print ('Close resource connection')
#yield生成器.return 退出函数。yield中断函数，然后执行下一步
with make_myresource() as r:
    r.query()
```