



ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Модуль pdb, отладчик Python

 mango-office.ru

РЕКЛАМА



Виртуальная АТС Расширенная

2 000 ₽

Узнать больше

Статьи

О том, как использовать отладчик pdb, написанных на Python

Модуль pdb, отладчик Python



Активный отладчик исходного кода для программ Python. Он поддерживает установку условных точек останова, выполнение на уровне строки исходного кода, проверку стековых фреймов, листинг исходного кода Python в контексте любого стекового фрейма. Он также поддерживает посмертную отладку и управление программой.

Отладчик является расширяемым - он фактически определяется как [класс pdb.Pdb\(\)](#).

Типичное использование для запуска программы под управлением отладчика:

```
>>> import pdb
>>> import mymodule
>>> pdb.run('mymodule.test()')
# > <string>(0)?()
# (Pdb) continue
# > <string>(1)?()
# (Pdb) continue
# NameError: 'spam'
# > <string>(1)?()
# (Pdb)
```

[Модуль pdb](#) также может быть вызван как скрипт для отладки других скриптов.

```
python3 -m pdb myscript.py
```

При вызове в виде сценария модуль pdb автоматически запускает post-mortem отладку, если отлаживаемая программа завершает работу ненормально. После post-mortem отладки или после обычного выхода из программы, модуль pdb перезапустит программу. Автоматический перезапуск сохраняет состояние pdb, например точки останова и в большинстве случаев более полезен, чем выход из отладчика при выходе из программы.

Типичное использование [pdb.set_trace\(\)](#) для взлома отладчика из запущенной программы:

```
import pdb

class MyClass():
    def __init__(self, loops):
        self.count = loops

    def start(self):
        for i in range(self.count):
            pdb.set_trace()
            print(i)
        return

if __name__ == '__main__':
    MyClass(5).start()
```

Затем вы можете пройти по коду после [pdb.set_trace\(\)](#) и продолжить работу без отладчика, используя [команду continue](#).

Вверх

Для команд и аргументов команд доступно завершение табуляции через [модуль readline](#), например в качестве аргументов [команды p](#) предлагаются текущие глобальные и локальные имена.

```
$ python3 test.py
> .РЕКЛАМА docs-python/test.py(10)start()
-> print(i)
(Pdb) p MyClass
<class '__main__.MyClass'>
(Pdb) p MyClass.start
<function MyClass.start at 0x7f623f0502f0>
(Pdb) p MyClass(2).start()
0
1
None
(Pdb)
```

Встроенная [функция breakpoint\(\)](#) при вызове со значениями по умолчанию может использоваться вместо `import pdb; pdb.set_trace()`.

Типичное использование для проверки сбойной программы:

```
>>> import pdb
>>> import mymodule
>>> mymodule.test()
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
#   File "./mymodule.py", line 4, in test
#     test2()
#   File "./mymodule.py", line 3, in test2
#     print(spam)
# NameError: spam
>>> pdb.pm()
# > ./mymodule.py(3)test2()
# -> print(spam)
# (Pdb)
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Команды отладчика модуля pdb](#)
- [Функция run\(\) модуля pdb](#)
- [Функция runeval\(\) модуля pdb](#)
- [Функция set_trace\(\) модуля pdb](#)
- [Функция runcall\(\) модуля pdb](#)
- [Функция post_mortem\(\) модуля pdb](#)
- [Класс Pdb\(\) модуля pdb](#)