

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Модуль multipledispatch в Python

skysmart

УЗНАЙТЕ
ВСЕ ПЛЮСЫ
ОБУЧЕНИЯ НА
БЕСПЛАТНОМ
УРОКЕ

skysmart.ru

РЕКЛАМА

Программирование для детей и подростков в Skysmart

Узнать больше

для вас
промокод! → 30-dlyaVAS

cutwoodshop.ru

3D карта мира из дерева
по специальной цене!

Отличный подарок вам и вашим
близким. Заказывайте на сайте
используя промокод!

Ручная работа

Гарантия качества

Подсветка

Купить

Модуль multipledispatch в Python

Установка модуля multipledispatch в виртуальное окружение:

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# ставим модуль multipledispatch
(VirtualEnv):~$ python3 -m pip install -U multipledispatch
```

Содержание:

- Пример использования модуля multipledispatch.
- Разрешение перегруженных методов/функций.
 - Реализация метода/функции для нескольких типов.
 - Использование абстрактных типов.
 - Выбор конкретной реализации.
- Множественная отправка.
 - Множественная вариативная отправка.
 - Неоднозначность выбора реализации перегруженной функции.

Пример использования модуля multipledispatch.

Пример перегрузки функций:

```
from multipledispatch import dispatch

@dispatch(object, object)
# базовая реализация функции `add()`
def add(x, y):
    return f"{x} + {y}"

@dispatch(int, int)
# реализация функции `add()` для целых чисел
def add(x, y):
    return x + y
```

```
>>> add(1, 2)
# 3
>>> add(1, 'hello')
```



а:

patch

`.add()`

для целых чисел

енных методов/функций.

Известно, что диспетчеризация выбирает метод/функцию, на основе анализа типов входящих аргументов.

```
from multipledispatch import dispatch

@dispatch(int)
# реализация функции `f()` для целого числа
def f(x):
    # увеличит целое число
    return x + 1

@dispatch(float)
# реализация функции `f()` для вещественного числа
def f(x):
    # уменьшит вещественное число
    return x - 1

>>> f(1)
# 2
>>> f(1.0)
# 0.0
```

Реализация метода/функции для нескольких типов.

Аналогично встроенной [функции isinstance\(\)](#), в декораторе @dispatch() указываются несколько допустимых типов с помощью кортежа. Приведенная ниже реализация f() для (list, tuple) применит реализацию f() для целого числа к каждому элементу в списке или кортеже, переданному в качестве аргумента.

```
@dispatch((list, tuple))
# реализация функции `f()` для типов `list` и `tuple`
def f(x):
    """ Применит `f(y: int)` к каждому элементу в списке или кортеже """
    return [f(y) for y in x]


>>> f([1, 2, 3])
# [2, 3, 4]
>>> f((1, 2, 3))
# [2, 3, 4]
```

Вверх

Использование абстрактных типов.

В декораторе `@dispatch()` можно также использовать абстрактные классы, такие как `Iterable` и `Number`, вместо [типов объединения](#), таких как `(list, tuple)` или `(int, float)` соответственно.

РЕКЛАМА



для вас промокод!

→ 30-dlyaVAS

cutwoodshop.ru

3D карта мира из дерева по специальной цене!

%

Отличный подарок вам и вашим близким. Заказывайте на сайте используя промокод!

Ручная работа

Гарантия качества

Подсветка

Купить

```
>>> flatten([1, [2], 3])
# [1, 2, 3]
>>> flatten([1, 2, (3, 4), [[5]], [(6, 7), (8, 9)]])
# [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Так как строки итерируемы, то они тоже будут "сглажены":

```
>>> flatten([1, 'hello', 3])
# [1, 'h', 'e', 'l', 'l', 'o', 3]
```

Нужно избегать этого, конкретизируя реализацию `flatten()` до типа [str](#). Поскольку тип `str` более конкретен, чем `Iterable`, то следующая реализация имеет приоритет для типа `str`.

```
@dispatch(str)
def flatten(s):
    return s

>>> flatten([1, 'hello', 3])
# [1, 'hello', 3]
```

Модуль `multipledispatch` зависит от механизма работы [функции `issubclass\(\)`](#), который определяет, какие типы более специфичны, чем другие.

Множественная отправка.

Все эти правила применяются, когда метод/функция принимает несколько аргументов в качестве входных данных.

```
from multipledispatch import dispatch

@dispatch(object, object)
def f(x, y):
    return x + y


@dispatch(object, float)
def f(x, y):
    """ Квадрат второго аргумента, если это `float` """
    return x + y**2

>>> f(1, 10)
```

Вверх

```
# 11
>>> f(1.0, 10.0)
# 101.0
```

РЕКЛАМА



для вас промокод! → 30-dlyaVAS

cutwoodshop.ru

3D карта мира из дерева по специальной цене!

%

Отличный подарок вам и вашим близким. Заказывайте на сайте используя промокод!

Ручная работа

Гарантия качества

Подсветка

Купить

многоэлементная отправка.

Множественная отправка (включая поддержку типов объединения) в качестве последнего аргумента функции.

Множественная отправка принимает список, содержащий тип аргументов, которые принимает функция.

Множественная отправка принимает число с плавающей запятой, за которым следует любое число (включая 0) либо int, либо

patch

```
args):
    ...)
```

реализации перегруженной функции.

Множественная отправка, когда разные реализации функции одинаково допустимы.

patch

```
@dispatch(object, object)
def f(x, y):
    return x + y

@dispatch(object, float)
def f(x, y):
    """ Квадрат второго аргумента, если он `float` """
    return x + y**2

@dispatch(float, object)
def f(x, y):
    """ Квадрат первого аргумента, если он `float` """
    return x**2 + y

>>> f(2.0, 10.0)
# ?
```

Какого результата ждать: $2,0 \cdot 2 + 10,0$ или $2,0 + 10,0 \cdot 2$? Типы входных данных удовлетворяют трем различным реализациям, две из которых имеют одинаковую силу.

input types:	float, float
Вариант 1:	object, object
Вариант 2:	object, float
Вариант 3:	float, object

Вариант 1 - строго менее конкретен, чем варианты 2 или 3, поэтому он отбрасывается. Но варианты 2 и 3 одинаково специфичны, поэтому неясно, какой из них использовать.

Чтобы решить такие проблемы, как эта, множественная отправка проверяет предоставленные ей сигнатуры типов и ищет неоднозначности. Затем появляется предупреждение, подобное следующему:

```
multipledispatch/dispatcher.py:27: AmbiguityWarning:
Ambiguities exist in dispatched function f

The following signatures may result in ambiguous behavior:
    [object, float], [float, object]


Сделайте следующее:
```

Вверх

```
@dispatch(float, float)
def f(...)
    warn(warning_text(dispatcher.name, ambiguities), AmbiguityWarning)
14.0
```

Когда в перегруженном методе/функции есть двусмысленность, а так же помогает создать
лучше функция с сигнатурой (float, float) более конкретна, чем варианты 2 или 3, и таким
избежать этого предупреждения, необходимо создать эту реализацию раньше других.



 cutwoodshop.ru

3D карта мира из дерева по специальной цене!

%

Отличный подарок вам и вашим близким. Заказывайте на сайте используя промокод!

Ручная работа >

Гарантия качества >

[DOCS-Python.ru](#)™, 2023 г. —

Подсветка >

Купить

ь, то одна из конкурирующих функций будет выбрана псевдослучайно. По умолчанию, выбор
г согласован во время сеанса интерпретатора, но может меняться от сеанса к сеансу.

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs_python_ru](#)