Python

План

- Практика
- Синтаксический сахар
- Принципы проектирования

Синтаксический сахар

Списковые выражения (list comprehensions)

- Позволяют создавать списки "в одну строку"
- Позволяют создавать не только списки

Практика

Файл "comprehensions.py"

lambda

- Анонимные функции, позволяют создавать функцию только для одного раза
- Могут занимать только одну строку
- Если сомневаетесь, используйте def

Формализмы

Лямбда-выражение *в программировании — специальный синтаксис для определения функциональных объектов, заимствованный из λ-исчисления. Применяется как правило для объявления анонимных функций по месту их использования, и обычно допускает замыкание на лексический контекст, в котором это выражение использовано. Используя лямбда-выражения, можно объявлять функции в любом месте кода.*

Практика

Файл "lambdas.py"

Полезные концепции из функционального программирования

- map - выполняет функцию для каждого объекта из массива входных параметров

- filter - оставляет из массива только что удовлетворяет условию

- reduce - выполняет функцию для каждого объекта из массива входных параметров, оперируя остатком

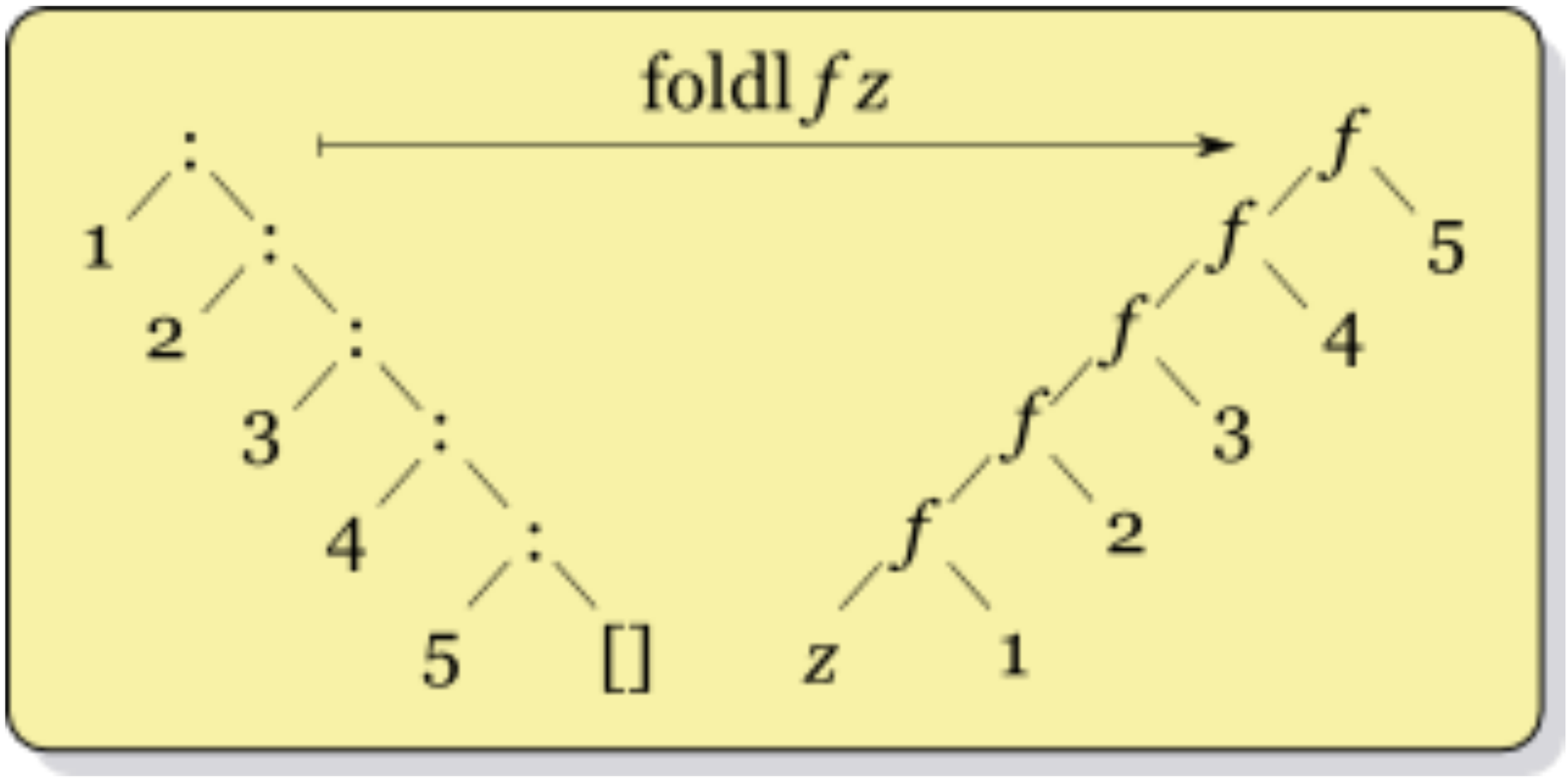
Формализмы

`map` *— функция высшего порядка, используемая во многих языках программирования, которая применяет данную функцию к каждому элементу списка, возвращая список результатов. При рассмотрении в функциональной форме она часто называется «применить-ко-всем».*

Формализмы

Свёртка списка *(англ. folding, также известна как reduce, accumulate) в программировании — функция высшего порядка, которая производит преобразование структуры данных к единственному атомарному значению при помощи заданной функции. Операция свёртки часто используется в функциональном программированиипри обработке списков.*

Свертка списка (foldl/reduce)



Практика

Файл "functions/fp.py"

Декораторы

Декораторы - очень простые

- Декоратор - такая функция
- Которая принимает функцию
- И возвращает функцию

Практика

Файл "decorators/*.py"

Принципы проектирования

Для всего кода

Принципы проектирования ПО

- DRY - Don't repeat yourself (principles/dry.py)
- KISS - Keep it short and simple (https://github.com/sobolevn/python-code-disasters/blob/master/python/create_objects.py)
- YAGNI - You Ain't Gonna Need It. Не создавайте что-либо, пока оно не понадобится.
- SOLID - фундаментальный принцип построения ПО

Интерфейс

Интерфе́йс *(англ. interface) — программная/синтаксическая структура, определяющая отношение между объектами, которые разделяют определённое поведенческое множество и не связаны никак иначе. При проектировании классов, разработка интерфейса тождественна разработке спецификации (множества методов, которые каждый класс, использующий интерфейс, должен реализовывать).*

SOLID

- Single responsibility principle: На каждый класс должна быть возложена одна-единственная обязанность.

- Open/closed principle: Программные сущности должны быть открыты для расширения, но закрыты для изменения.

- Liskov substitution principle: Функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом.

- Interface segregation principle: Много специализированных интерфейсов лучше, чем один универсальный.

- Dependency inversion principle: Зависимости внутри системы строятся на основе абстракций. Модули верхнего уровня не зависят от модулей нижнего уровня. Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций.

Закон Деметры

Закон Деметры *(англ. Law of Demeter, LoD) — набор правил проектирования при разработке программного обеспечения, в частности объектно-ориентированных программ, накладывающий ограничения на взаимодействия объектов (модулей). Обобщенно, закон Деметры является специальным случаем слабого зацепления(англ. loose coupling). Правила были предложены в конце 1987 в северо-восточном Университете (Бостон, Массачусетс, США).*

Говоря упрощённо, каждый программный модуль:

- *должен обладать ограниченным знанием о других модулях: знать о модулях, которые имеют «непосредственное» отношение к этому модулю.*
- *должен взаимодействовать только с известными ему модулями «друзьями», не взаимодействовать с незнакомцами.*
- *обращаться только к непосредственным «друзьям».*

Аналогия из жизни: Если Вы хотите, чтобы собака побежала, глупо командовать её лапами, лучше отдать команду собаке, а она уже разберётся со своими лапами сама.

Основной идеей является то, что объект должен иметь как можно меньше представления о структуре и свойствах чего угодно (включая собственные подкомпоненты).

Но Python отличается от обычных ООП языков

https://github.com/faif/python-patterns

Шаблоны проектирования родились, когда разработчики решали свои проблемы

https://github.com/kamranahmedse/design-patterns-for-humans