

FP В PYTHON

это проще, чем вы думали

September 10, 2016

ПРИЧИНЫ?

Версия #1

Почему это важно, понимать функциональный подход к программированию?

Это позволяет по-новому взглянуть на ваш код, сделать его более модульным и надежным



ПРИЧИНЫ?



ПРИЧИНЫ?

- FP - это парадигма, не привязанная к языку
- Python - мультипарадигменный язык, позволяющий писать функционально
- Это позволяет увидеть преимущества FP в вашем коде уже завтра

КРАТКОЕ ВВЕДЕНИЕ В FR

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

- Неизменяемость данных (immutability)
- Чистые функции и side effects
- Функции высшего порядка
- Монады (?)

Функциональные программы обычно оперируют неизменяемыми данными. Вместо изменения оригинального значения, изменяется его копия. Та часть данных, которая сохранилась и в оригинальном объекте и в его копии, переиспользуется, что предотвращает излишний расход памяти.

ЧИСТЫЕ ФУНКЦИИ И SIDE EFFECTS

A function or expression is said to have a side effect if it modifies some state or has an interaction with the outside world. For example, a particular function might modify a global variable, modify one of its arguments, raise an exception, perform IO operation, or call other side-effecting functions.

Abstract Data Type (ADT)

FUNCTION COMPOSITION





Function composition is the act of pipelining the result of one function, to the input of another, creating an entirely new function.

FP PARTS IN PYTHON

WHAT WE CAN USE IN PYTHON

- Immutable data structures:
 - string
 - tuple/namedtuple
 - frozenset
- Higher-order functions
- List comprehension
- Generators (another way to rewrite recursion)
- itertools
- functools

WHAT WE CAN'T

- Tail recursion
- Pure functions
- Pattern matching 
- Automatic currying 
- Monads 
- ADT 

- Использование «чистого» Python
- Собственные надстройки
- Сторонние библиотеки

EXAMPLES

```

def get_item_data(self):
    data = {}
    if self.entity_id:
        entity = course_cache.entities[self.entity_id]
        if entity.group_id:
            data['section_name'] = cache.groups[entity.group_id].title
        if self.step_id:
            step = entity.step_by_id(self.step_id)
            if step:
                data["entity_prompt"] = step.prompt()
            else:
                logger.warning()
    else:
        if self.test_item_id:
            # do something with data[]
        else:
            # do something with data[]
        data["task_questions"] = process_questions()
        data["task_answer_choices"] = process_choices()
        # do something
    return data

```

```

def noop(*args, **kwargs):
    pass

def get_item_data(self, entity_id, test_item_id):
    def common_part():
        data["task_questions"] = process_questions()
        data["task_answer_choices"] = process_choices()
    data = {}
    if entity_id:
        entity = course_cache.entities[self.entity_id]
        group = cache.groups.get(entity.group_id)
        step = entity.step_by_id(self.step_id)
        data['section_name'] = getattr(group, "title", None)
        data['entity_prompt'] = getattr(step, "prompt", noop)()
    if test_item_id:
        # do something with data []
        common_part()
    if entity_id is None and test_item_id is None:
        # do something with data []
        common_part()
    return data





```

PATTERN MATCHING

MAYBE

БИБЛИОТЕКИ

СПИСОК FP БИБЛИОТЕК ДЛЯ PYTHON

- Tail recursion
- Pure functions
- Pattern matching 
- Automatic currying 
- Monads 
- ADT 

QUESTIONS?