Лабы

Главная » Python

Python. Лабораторная работа №5

> Функции высшего порядка и лямбда-выражения

Задания для самостоятельного выполнения

Скачайте архив. Запуск тестирования аналогичен ЛР№3.

WWPD (What Would Python Display?)

Для всех вопросов данного типа нужно ввести верный ответ. Для остальных случаев есть простые правила - если вы считаете, что:

- ответом будет `<function...>` - введите `Function`,
- интерпретатор выдаст ошибку - введите `Error`,
- ничего не будет показано - введите `Nothing`.

lambda_curry2

Решение должно занимать **ровно одну строчку кода**.

count_cond

Рассмотрите следующие реализации функций `count_factors` и `count_primes`:

```
def count_factors(n):
    """Возвращает число положительных делителей числа n.
    >>> count_factors(6)
    4   # 1, 2, 3, 6
```

```
    >>> count_factors(4)
    3   # 1, 2, 4
    """
    i, count = 1, 0
    while i <= n:
        if n % i == 0:
            count += 1
        i += 1
    return count

def count_primes(n):
    """Возвращает число простых чисел до n включительно.
    >>> count_primes(6)
    3   # 2, 3, 5
    >>> count_primes(13)
    6   # 2, 3, 5, 7, 11, 13
    """
    i, count = 1, 0
    while i <= n:
        if is_prime(i):
            count += 1
        i += 1
    return count

def is_prime(n):
    return count_factors(n) == 2 # делится только на 1 и на себя
```

Эти функции довольно похожи - обобщите их логику в функции `count_cond`.

composite_identity

Используйте функцию `compose1`.

cycle (I Heard You Liked Functions…). Необязательное задание

Напишите функцию, которая принимает в качестве аргументов 3 функции `f1, f2, f3` , и возвращает функцию, которая принимает целочисленный аргумент `n` и возвращает ещё одну функцию. Эта последняя функция должна принимать аргумент `x` и последовательно применять к нему функции `f1, f2, f3` `n` раз. Например:

- n = 0 - вернуть x
- n = 1 - вернуть f1(x)
- n = 2 - f2(f1(x))
- n = 3 - f3(f2(f1(x)))
- n = 4 - начать следующий цикл, вернув f1(f3(f2(f1(x)))) И т.д.

> Подсказка: большая часть кода будет в наиболее вложенной функции.

Построение контекстных диаграмм

make_adder

Постройте контекстную диаграмму следующего кода:

```
n = 9
def make_adder(n):
    return lambda k: k + n
add_ten = make_adder(n+1)
result = add_ten(n)
```

Всего у вас будет 3 фрейма, включая глобальный. Ответьте на следующие вопросы:

- В глобальном фрейме имя `add_ten` указывает на функцию. Какое внутреннее имя у этой функции и какой фрейм будет для ней родительским?
- Какое имя у фрейма `f2` (`add_ten` или λ)? Какой фрейм является родительским для него?
- С каким значением связана переменная `result` ?

Можно подглядеть ответ тут

lambda

Попробуйте предсказать, что выведет интерпретатор и постройте контекстную диаграмму.

```
a = lambda x: x * 2 + 1
def b(b, x):
    return b(x + a(x))
x = 3
b(a, x)
```

Проверьте себя тут

← ПРЕДЫДУЩАЯ
Python. Лабораторная работа №4

СЛЕДУЮЩАЯ →
Python. Лабораторная работа №6