Разбор домашнего задания

Роман Булгаков
Спикер курса

Skillbox

Задача «Односвязный список»

Связный список — это структура данных, которая состоит из элементов, которые называются **узлами**. В узлах хранятся данные, а между собой узлы соединены связями. Связь — это ссылка на следующий или предыдущий элемент списка.

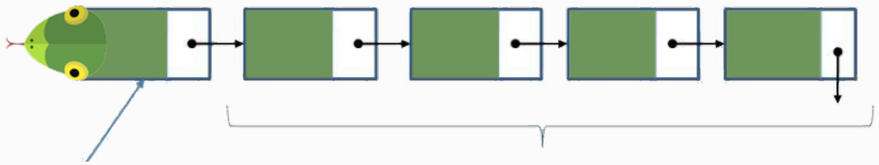
класс «Узел»

Атрибуты класса «Узел»:

- значение
- ссылка на следующий узел

класс «ОднСписок»

Атрибуты класса «ОднСписок»:
указатель на первый (головной) узел.



Голова **Хвост**

Функция как объект. Функции высшего порядка

Роман Булгаков
Спикер курса

Skillbox

Задача «Таймер»

Условие задачи: функция.

Выходные данные: время работы функции.

```python
def squares_sum():
    number = 100
    result = 0
    for _ in range(number + 1):
        result += sum([i_num**2 for i_num in range(10000)])

    return result
```

Функции

```python
def squares_sum():
    number = 100
    result = 0
    for _ in range(number + 1):
        result += sum([i_num**2 for i_num in range(10000)])

    return result
```

Функция (объект) первого класса

Передаётся и используется в качестве аргумента для других функций.

```python
def timer(func):
    started_at = time.time()
    result = func()
    ended_at = time.time()
    run_time = round(ended_at - started_at, 4)
    print('Функция работала {} секунд(ы)'.format(run_time))

    return result
```

Функция высшего порядка

Принимает в качестве аргумента другую функцию **и/или возвращает** функцию как результат работы.

Декораторы

Роман Булгаков
Спикер курса

Skillbox

Реализация декоратора

```
def decorator(func):
    def wrapped_func(*args, **kwargs):
        # Код до вызова функции
        value = func(*args, **kwargs)
        # Код после вызова функции
        return value
    return wrapped_func
```

Декоратор является одним из паттернов проектирования.

```
def decorator(func):
    def wrapped_func(*args, **kwargs):
        # Код до вызова функции
        value = func(*args, **kwargs)
        # Код после вызова функции
        return value
    return wrapped_func


@decorator
def some_function(*args):
    pass
```

Некоторые особенности использования декораторов

Роман Булгаков
Спикер курса

Skillbox

Задача «Плагины»

Условие задачи:

- функции
- нужен декоратор, «регистрирующий» функции как плагины

Выходные данные:

- PLUGINS — зарегистрированные плагины

Модуль functools.
Декоратор functools.wraps()

Роман Булгаков
Спикер курса

Skillbox

Итоги модуля

- def timer(func):
  . . . . .
  my_func = squares_sum
- timer — функция высшего порядка
- def timer(func):
  def wrapper(*args, **kwargs):
  . . . . . .
  return wrapper
- @timer
  @logging
- @functools.wraps(func)