Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Якимова Татьяна ИУ5-35Б

Парадигмы и конструкции языков программирования

ОТЧЁТ ПО
Лабораторной работе №6

Москва
2023

Задание.

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно располагаться в отдель-ном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соот-ветствующего задания.

- field.py: необходимо реализовать генератор field. Генератор field последо-вательно выдает значения ключей словаря.

- gen_random.py: необходимо реализовать генератор gen_random (количе-ство, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до мак-симума, включая границы диапазона.

- unique.py: необходимо реализовать итератор Unique(данные), который прини-мает на вход массив или генератор и итерируется по элементам, пропуская дуб-ликаты.

- sort.py: дан массив 1, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив 2, кото-рые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сорти-ровку необходимо осуществлять с помощью функции sorted.

- print_result.py: необходимо реализовать декоратор print_result, который выво-дит на экран результат выполнения функции.

- cm_timer.py: необходимо написать контекстные менеджеры cm_timer_1 и cm_timer_2, которые считают время работы блока кода и выводят его на экран.

- process_data.py: необходимо реализовать 4 функции - f1, f2, f3, f4. Каждая функция вызывается, принимая на вход результат работы предыдущей. За счет декоратора @print_result печатается результат, а контекстный мене-джер cm_timer_1 выводит время работы цепочки функций.

Текст программы.

cm_timer.py

```
import time

class cm_timer_1:
    def __enter__(self):      #менеджер контекст
        self.start_time = time.time()

    def __exit__(self, *args):
        self.end_time = time.time()
        execution_time = self.end_time - self.start_time
        print(f"time: {execution_time}")

import time
```

```
from contextlib import contextmanager
@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    execution_time = end_time - start_time
    print(f"time: {execution_time}")

with cm_timer_1():
    time.sleep(5.5)
with cm_timer_2():
    time.sleep(5.5)
```

Файл gen_random.py

```
import random
def gen_random(num_count, begin, end):
    for i in range(num_count):
        yield random.randint(begin,end)

for num in gen_random(5,1,3):
    print(num)
```

Файл field.py

```
def field(items, *args):
    for item in items:
        if len(args) == 1:
            field_value = item.get(args[0])
            if field_value is not None:
                yield field_value
        else:
            filtered_item = {key: item.get(key) for key in args if item.get(key)
is not None}
            if filtered_item:
                yield filtered_item


goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'color': 'black'}
]


for title in field(goods, 'title'):
    print(title)


for item in field(goods, 'title', 'price'):
    print(item)
```

файл print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
```

```
        print(f"Выполнение функции {func.__name__}:")
        if isinstance(result, list):
            for item in result:
                print(item)
        elif isinstance(result, dict):
            for key, value in result.items():
                print(f"{key} = {value}")
        else:
            print(result)
        return result
    return wrapper

@print_result
def add(a, b):
    return a + b
@print_result
def multiply(a, b):
    return a * b
@print_result
def get_list():
    return ['a','c' ]
@print_result
def get_dict():
    return {'a': 1, 'b': 2, 'c': 3}
add(2, 3)
multiply(4, 5)
get_list()
get_dict()
```

Файл procces_data.py

```
import json
from datetime import datetime
from contextlib import contextmanager

def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(result)
        return result
    return wrapper

@contextmanager
    start_time = datetime.now()
    yield
    end_time = datetime.now()
    print("Выполнение заняло: {}".format(end_time - start_time))

def f1(data):
    return sorted(set(item['Профессия'] for item in data), key=lambda x:
x.lower())

def f2(data):
    return list(filter(lambda x: x.startswith('программист'), data))

def f3(data):
```

```
    return list(map(lambda x: x + ', с опытом Python', data))

def f4(data):
    salaries = range(100000, 200001)
    return [f'{profession}, зарплата {salary} руб.' for profession, salary in
zip(data, salaries)]

with open('data_light.json') as file:
    data = json.load(file)

with cm_timer_1():
    f4_result = f4(f3(f2(f1(data))))
```

Файл sort.py

```
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
sorted_data = sorted(data, key=lambda x: abs(x), reverse=True)
print(sorted_data)

data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
sorted_data = sorted(data, key=abs, reverse=True)
print(sorted_data)
```

Файл Unique.py

```
class Unique:
    def __init__(self, data, ignore_case=False):
        self.data = data
        self.ignore_case = ignore_case
        self.seen = set()
    def __iter__(self):
        return self
    def __next__(self):
        for item in self.data:
            if self.ignore_case and isinstance(item, str):
                item_key = item.lower()
            else:
                item_key = item
            if item_key not in self.seen:
                self.seen.add(item_key)
                return item
        raise StopIteration
data_list = [1, 2, 2, 3, 4, 4, 5]
unique_list = Unique(data_list)
for item in unique_list:
    print(item)

data_generator = (x for x in range(5))
unique_generator = Unique(data_generator)
for item in unique_generator:
    print(item)

data_list = ['apple', 'banana', 'Apple', 'cherry']
unique_list_ignore_case = Unique(data_list, ignore_case=True)
for item in unique_list_ignore_case:
    print(item)
```