

**Московский государственный технический
университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по лабораторной работе №3
«Функциональные возможности языка Python»

Выполнил:
Аляев Д. В. ИУ5-51

Проверил:
Гапанюк Ю. Е.
Балашов А. М.

Москва, 2021 г.

Цель работы:

Изучение возможностей функционального программирования в языке Python.

Задание:

Задание лабораторной работы состоит из решения нескольких задач. Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fr. Решение каждой задачи должно располагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

Выполнение:

Задача 1:

Необходимо реализовать генератор field. Генератор field последовательно выдает значения ключей словаря.

Реализация:

```
def field(items, *args):
    assert len(args) > 0
    for val in items:
        result = {}
        for arg in args:
            if arg in val:
                result[arg] = val[arg]
        if result:
            yield result

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

field_gen = field(goods, 'title', 'price')
for i in field_gen:
    print(i)
```

Результат:

```
/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fr/field.py
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
{'title': 'отдыха'}

Process finished with exit code 0
```

Задача 2:

Необходимо реализовать генератор gen_random(количество, минимум, максимум), который последовательно выдает заданное количество случайных чисел в заданном диапазоне от минимума до максимума, включая границы

диапазона.

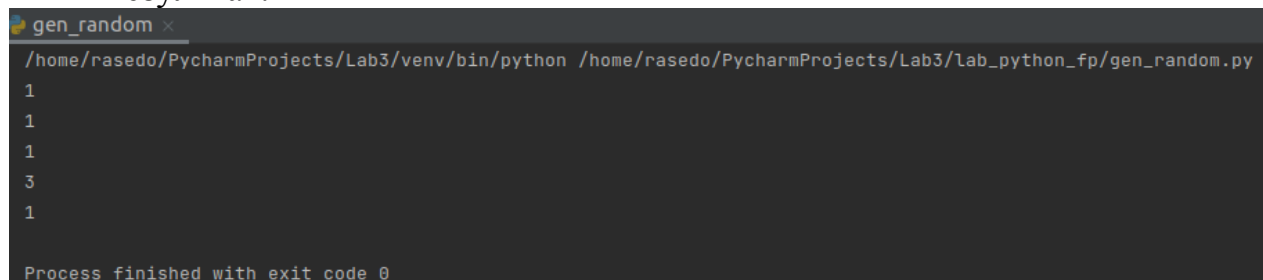
Реализация:

```
from random import randint

def gen_random(num_count, begin, end):
    for i in range(0, num_count):
        yield randint(begin, end)

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

Результат:



The screenshot shows a terminal window titled 'gen_random x'. The command executed is `/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fp/gen_random.py`. The output consists of five lines of random integers: 1, 1, 1, 3, and 1. At the bottom, it says 'Process finished with exit code 0'.

Задача 3:

Необходимо реализовать итератор Unique(данные), который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.

Реализация:

```
import types

class Unique(object):
    def __init__(self, items, **kwargs):
        self.used_elements = set()
        self.items = items
        self.index = 0
        if 'ignore_case' in kwargs:
            self.ignore_case = kwargs['ignore_case']
        else:
            self.ignore_case = False

    def __next__(self):
        if isinstance(self.items, types.GeneratorType):
            while True:
                current = self.items.__next__()
                if isinstance(current, dict):
                    current = current['job-name']
                if self.ignore_case and isinstance(current, str):
                    current = current.lower()
                self.index = self.index + 1
                if current not in self.used_elements:
                    self.used_elements.add(current)
```

```

        return current
    else:
        while True:
            if self.index >= len(self.items):
                raise StopIteration
            else:
                if self.ignore_case:
                    current = self.items[self.index].lower()
                else:
                    current = self.items[self.index]
                self.index = self.index + 1
                if current not in self.used_elements:
                    self.used_elements.add(current)
                    return current

def __iter__(self):
    return self

lst1 = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
lst2 = [1, 3, 2, 3, 2, 1, 4, 3, 3, 3]
result = ""
for i in Unique((i for i in lst1), ignore_case=False):
    result += str(i) + ' '
print(f'Generator: {result}')
result = ""
for i in Unique(lst2):
    result += str(i) + ' '
print(f'List: {result}')

```

Результат:

```

/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fp/unique.py
Generator: a A b B
List: 1 3 2 4

Process finished with exit code 0

```

Задача 4:

Дан массив 1, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив 2, которые содержит значения массива 1, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции sorted. Пример:

```
data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
```

Вывод: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Необходимо решить задачу двумя способами:

- С использованием lambda-функции.
- Без использования lambda-функции.

Реализация:

```

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]

if __name__ == '__main__':

```

```
result = sorted(data, key=abs, reverse=True)
print(result)

result_with_lambda = (lambda val: sorted(val, key=abs, reverse=True))(data)
print(result_with_lambda)
```

Результат:

```
/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fp/sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

Задача 5:

Необходимо реализовать декоратор `print_result`, который выводит на экран результат выполнения функции.

Декоратор должен принимать на вход функцию, вызывать её, печатать в консоль имя функции и результат выполнения, после чего возвращать результат выполнения.

Если функция вернула список (`list`), то значения элементов списка должны выводиться в столбик.

Если функция вернула словарь (`dict`), то ключи и значения должны выводиться в столбик через знак равенства.

Реализация:

```
def print_result(func):
    def decorated_func(*args, **kwargs):
        print(func.__name__)
        if isinstance(func(*args, **kwargs), dict):
            for key in func(*args, **kwargs):
                print(key, '=', func(*args, **kwargs)[key])
        elif isinstance(func(*args, **kwargs), list):
            for item in func(*args, **kwargs):
                print(item)
        else:
            print(func(*args, **kwargs))
        return func(*args, **kwargs)

    return decorated_func

@print_result
def test_1():
    return 1

@print_result
def test_2():
    return 'iu5'

@print_result
def test_3():
    return {'a': 1, 'b': 2}
```

```

@print_result
def test_4():
    return [1, 2]

if __name__ == '__main__':
    print('!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()

```

Результат:

```

/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fp/print_result.py
!!!!!!!
test_1
1
test_2
iv5
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0

```

Задача 6:

Необходимо написать контекстные менеджеры `cm_timer_1` и `cm_timer_2`, которые считают время работы блока кода и выводят его на экран.

Реализация:

```

from time import sleep, time
from contextlib import contextmanager

class cm_timer_1:
    def __init__(self):
        self.timer = time()

    def __enter__(self):
        pass

    def __exit__(self, exp_type, exp_value, traceback):
        self.timer = time() - self.timer
        print(self.timer)

@contextmanager
def cm_timer_2():
    timer = time()
    yield
    timer = time() - timer

```

```
print(timer)

with cm_timer_1():
    sleep(2)
with cm_timer_2():
    sleep(3)
```

Результат:

```
cm_timer x
/home/rasedo/PycharmProjects/Lab3/venv/bin/python /home/rasedo/PycharmProjects/Lab3/lab_python_fp/cm_timer.py
2.002106189727783
3.0012989044189453

Process finished with exit code 0
```

Задача 7:

- В файле data_light.json содержится фрагмент списка вакансий.
- Структура данных представляет собой список словарей с множеством полей: название работы, место, уровень зарплаты и т.д.
- 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 менеджер cm_timer_1 выводит время работы цепочки функций.
- Предполагается, что функции f1, f2, f3 будут реализованы в одну строку. В реализации функции f4 может быть до 3 строк.
- Функция f1 должна вывести отсортированный список профессий без повторений (строки в разном регистре считать равными). Сортировка должна игнорировать регистр. Используйте наработки из предыдущих задач.
- Функция f2 должна фильтровать входной массив и возвращать только те элементы, которые начинаются со слова “программист”. Для фильтрации используйте функцию filter.
- Функция f3 должна модифицировать каждый элемент массива, добавив строку “с опытом Python” (все программисты должны быть знакомы с Python). Пример: Программист C# с опытом Python. Для модификации используйте функцию map.
- Функция f4 должна сгенерировать для каждой специальности зарплату от 100 000 до 200 000 рублей и присоединить её к названию специальности. Пример: Программист C# с опытом Python, зарплата 137287 руб. Используйте zip для обработки пары специальность — зарплата.

Реализация:

```
import json
from time import sleep
from cm_timer import SimpleContextManager
import json
from time import sleep
```

```

from cm_timer import cm_timer_1
from field import field
from unique import Unique
from print_result import print_result
from gen_random import gen_random

@print_result
def f1(data):
    return sorted(Unique(field(data, 'job-name'), ignore_case=True))

@print_result
def f2(data):
    return list(filter(lambda val: 'программист' in val[:11], data))

@print_result
def f3(data):
    return list(map(lambda val: val + " с опытом Python", data))

@print_result
def f4(data):
    return list(zip(data, list(map(lambda val: "Зарплата " + val + " руб", map(str, (gen_random(len(data),
10000, 20000)))))))

def main():
    with open('data_light.json', 'r', encoding='utf8') as temp:
        data = json.load(temp)
    with cm_timer_1():
        sleep(1.57)
        f4(f3(f2(f1(data))))

```

Результат:

process_data x

```
юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
f2
программист
программист / senior developer
программист 1с
программист с#
программист с++
программист с++/с#/java
программист/ junior developer
программист/ технический специалист
программист-разработчик информационных систем
f3
программист с опытом Python
программист / senior developer с опытом Python
программист 1с с опытом Python
программист с# с опытом Python
программист с++ с опытом Python
программист с++/с#/java с опытом Python
программист/ junior developer с опытом Python
программист/ технический специалист с опытом Python
программист-разработчик информационных систем с опытом Python
f4
('программист с опытом Python', 'Зарплата 1928 руб')
('программист / senior developer с опытом Python', 'Зарплата 1959 руб')
('программист 1с с опытом Python', 'Зарплата 1401 руб')
('программист с# с опытом Python', 'Зарплата 1640 руб')
('программист с++ с опытом Python', 'Зарплата 1959 руб')
('программист с++/с#/java с опытом Python', 'Зарплата 1508 руб')
('программист/ junior developer с опытом Python', 'Зарплата 1331 руб')
('программист/ технический специалист с опытом Python', 'Зарплата 1678 руб')
('программист-разработчик информационных систем с опытом Python', 'Зарплата 1715 руб')
time: 1.6

Process finished with exit code 0
```