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

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «БКИТ»
Отчет по лабораторной работе №3-4
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-35Б
Самойлов Константин
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Нардид А.Н.
Подпись и дата:

Москва, 2022 г.

## Описание задания

Задание лабораторной работы состоит из решения нескольких задач.

Файлы, содержащие решения отдельных задач, должны располагаться в пакете lab_python_fp. Решение каждой задачи должно раполагаться в отдельном файле.

При запуске каждого файла выдаются тестовые результаты выполнения соответствующего задания.

## Задача 1 (файл field.py)

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

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

`field(goods, 'title')` должен выдавать `'Ковер', 'Диван для отдыха'`

`field(goods, 'title', 'price')` должен выдавать `{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}`

- В качестве первого аргумента генератор принимает список словарей, дальше через *args генератор принимает неограниченное количествово аргументов.
- Если передан один аргумент, генератор последовательно выдает только значения полей, если значение поля равно None, то элемент пропускается.
- Если передано несколько аргументов, то последовательно выдаются словари, содержащие данные элементы. Если поле равно None, то оно пропускается. Если все поля содержат значения None, то пропускается элемент целиком.

Шаблон для реализации генератора:

```
# Пример:
# goods = [
#    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
#    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
# ]
# field(goods, 'title') должен выдавать 'Ковер', 'Диван для отдыха'
# field(goods, 'title', 'price') должен выдавать {'title': 'Ковер', 'price': 2000},
{'title': 'Диван для отдыха', 'price': 5300}

def field(items, *args):
    assert len(args) > 0
    # Необходимо реализовать генератор
```

## Задача 2 (файл gen_random.py)

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. Пример:
`gen_random(5, 1, 3)` должен выдать 5 случайных чисел в диапазоне от 1 до 3, например `2, 2, 3, 2, 1`
Шаблон для реализации генератора:

```
# Пример:
# gen_random(5, 1, 3) должен выдать выдать 5 случайных чисел
# в диапазоне от 1 до 3, например 2, 2, 3, 2, 1
# Hint: типовая реализация занимает 2 строки
def gen_random(num_count, begin, end):
    pass
    # Необходимо реализовать генератор
```

## Задача 3 (файл unique.py)

- Необходимо реализовать итератор `Unique(данные)`, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
- Конструктор итератора также принимает на вход именованный bool-параметр ignore_case, в зависимости от значения которого будут считаться одинаковыми строки в разном регистре. По умолчанию этот параметр равен False.
- При реализации необходимо использовать конструкцию **kwargs.
- Итератор должен поддерживать работу как со списками, так и с генераторами.
- Итератор не должен модифицировать возвращаемые значения.

Пример:

`data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]`
`Unique(data)` будет последовательно возвращать только 1 и 2.
`data = gen_random(10, 1, 3)`
`Unique(data)` будет последовательно возвращать только 1, 2 и 3.
`data = [‘a’, ‘A’, ‘b’, ‘B’, ‘a’, ‘A’, ‘b’, ‘B’]`
`Unique(data)` будет последовательно возвращать только a, A, b, B.
`Unique(data, ignore_case=True)` будет последовательно возвращать только a, b.
Шаблон для реализации класса-итератора:

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, **kwargs):
        # Нужно реализовать конструктор
```

```
        # В качестве ключевого аргумента, конструктор должен принимать bool-параметр
ignore_case,
        # в зависимости от значения которого будут считаться одинаковыми строки в разном
регистре
        # Например: ignore_case = True, Абв и АБВ - разные строки
        #           ignore_case = False, Абв и АБВ - одинаковые строки, одна из которых
удалится
        # По-умолчанию ignore_case = False
        pass

    def __next__(self):
        # Нужно реализовать __next__
        pass

    def __iter__(self):
        return self
```

## Задача 4 (файл sort.py)

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:

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

Шаблон реализации:

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

if __name__ == '__main__':
    result = ...
    print(result)

    result_with_lambda = ...
    print(result_with_lambda)
```

## Задача 5 (файл print_result.py)

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 выводить в столбик через знак равенства.

Шаблон реализации:

```
# Здесь должна быть реализация декоратора

@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 выполнения:

```
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
```

## Задача 6 (файл 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. Пример:

```
with cm_timer_1():
    sleep(5.5)
```

После завершения блока кода в консоль должно вывестись `time: 5.5` (реальное время может несколько отличаться).

`cm_timer_1` и `cm_timer_2` реализуют одинаковую функциональность, но должны быть реализованы двумя различными способами (на основе класса и с использованием библиотеки contextlib).

## Задача 7 (файл process_data.py)

- В предыдущих задачах были написаны все требуемые инструменты для работы с данными. Применим их на реальном примере.
- 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.

Шаблон реализации:

```
import json
import sys
# Сделаем другие необходимые импорты

path = None

# Необходимо в переменную path сохранить путь к файлу, который был передан при запуске
сценария

with open(path) as f:
    data = json.load(f)

# Далее необходимо реализовать все функции по заданию, заменив `raise NotImplemented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg):
    raise NotImplemented


@print_result
def f2(arg):
    raise NotImplemented


@print_result
def f3(arg):
    raise NotImplemented


@print_result
def f4(arg):
    raise NotImplemented


if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
```

# Текст программы

## Файлы пакета lab_python_fp:

### field.py

```
def field(items, *args):
    assert len(args) > 0, 'The parameter "args" is empty!'
    if len(args) == 1:
        return (item[el] for item in items for el in item if el == args[0] and
item[el] is not None)
    else:
        return {el: item[el] for item in items for el in item for argument in args
if
                el == argument and item[argument] is not None}


def num1():
    print('Задача №1')

    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    # должен выдавать 'Ковер', 'Диван для отдыха'.
    res = (field(goods, 'title'))
    for el in res:
        print(el)
    # должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отды-
ха', 'price': 5300}.
    print(field(goods, 'title', 'price'))


if __name__ == '__main__':
    num1()
```

### gen_random.py

```
from random import randint


def get_random(num_count, begin, end):
    """
    Рандомные числа
    :param num_count: число случайных чисел
    :param begin: с какого числа
    :param end: по какое
    :return: картеж чисел
    """
    return (randint(begin, end) for _ in range(num_count))


def num2():
    print('\nЗадача №2')
    numbers = get_random(4, 0, 5)
    for el in numbers:
        print(el, end=' ')
    print()
```

```
if __name__ == '__main__':
    num2()
```

**unique.py**

```
# Итератор для удаления дубликатов
class Unique(object):
    def __init__(self, items, ignore_case=False, **kwargs):
        self._data = items
        self._ignore_case = ignore_case
        self.__used_data = set()
        self.__index = 0

    def __next__(self):
        if self._ignore_case:
            for counter, el in enumerate(self._data):
                if type(el) is str:
                    self._data[counter] = el.lower()

        while True:
            if self.__index >= len(self._data):
                raise StopIteration
            else:
                current = self._data[self.__index]
                self.__index += 1
                if current not in self.__used_data:
                    self.__used_data.add(current)
                    return current

    def __iter__(self):
        return self


def num3():
    print('\nЗадача №3')
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    it = Unique(data, ignore_case=False)
    try:
        while True:
            print(it.__next__())
    except StopIteration:
        print('StopIteration error.')


if __name__ == '__main__':
    num3()
```

**sort.py**

```
def num4():
    data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
    result = sorted(data, key=abs, reverse=True)
    print(result)

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

```
if __name__ == '__main__':
    num4()
```

**print_result.py**

```
def print_result(func):
    def wrapper(lst=[], *args, **kwargs):
        print(func.__name__)

        if len(lst) == 0:
            result = func(*args, **kwargs)
        else:
            result = func(lst, *args, **kwargs)

        if type(result) is dict:
            for key, el in result.items():
                print(f'{key} = {el}')

        elif type(result) is list:
            print('\n'.join(map(str, result)))

        else:
            print(result)

        return result

    return wrapper


@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]


def num5():
    print('!!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()


if __name__ == '__main__':
    num5()
```

**main.py**

```
    from lab_python_fp.field import num1
from lab_python_fp.gen_random import num2
from lab_python_fp.unique import num3
from lab_python_fp.sort import num4
from lab_python_fp.print_result import num5
from lab_python_fp.cm_timer import num6
from lab_python_fp.process_data import num7


def main():
    num1()
    num2()
    num3()
    print("Задача №4")
    num4()
    print("Задача №5")
    num5()
    print("Задача №6")
    num6()
    print("Задача №7")
    num7()


if __name__ == '__main__':
    main()
```

**cm_timer.py**

```
from time import sleep, time
from contextlib import contextmanager


class cm_timer_1:
    def __int__(self):
        self.__start = 0
        self.__finish = 0

    def __enter__(self):
        self.__start = time()

    def __exit__(self, exp_type, exp_value, traceback):
        self.__finish = time()
        print(f'Time of work: {self.__finish - self.__start}')


@contextmanager
def cm_timer_2():
    st = time()
    yield None
    en = time()
    print(f'Time of work: {en - st}')


def num6():
    with cm_timer_1():
        sleep(2.5)

    with cm_timer_2():
```

```
        sleep(2.5)


if __name__ == '__main__':
    num6()
```

**Финальная задача**
**process_data.py**

```
from lab_python_fp.print_result import print_result
from lab_python_fp.cm_timer import cm_timer_1
from lab_python_fp.gen_random import get_random
import json
import sys
# Сделаем другие необходимые импорты

try:
    path = sys.argv[1]
    print(path)
except:
    path = 'data_light.json'

with open(path, "rb") as f:
    data = json.load(f)


# Далее необходимо реализовать все функции по заданию, заменив `raise NotImple-
mented`
# Предполагается, что функции f1, f2, f3 будут реализованы в одну строку
# В реализации функции f4 может быть до 3 строк

@print_result
def f1(arg) -> list:
    return sorted(list(set([el['job-name'] for el in arg])), key=lambda x:
x.lower())


# Функция f2 должна фильтровать входной массив и возвращать только те элементы,
# которые начинаются со слова “программист”. Для фильтрации используйте функцию
filter.

@print_result
def f2(arg) -> list:
    return list(filter(lambda text: (text.split())[0].lower() == 'программист',
arg))


# 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.

@print_result
def f3(arg) -> list:
    return list(map(lambda lst: lst + ' с опытом Python', arg))


# 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.
```

```
@print_result
def f4(arg) -> list:
    return dict(zip(arg, ['зарплата ' + str(el) + ' руб.' for el in get_ran-
dom(len(arg), 100000, 200000)]))

def num7():
    with cm_timer_1():
        f4(f3(f2(f1(data))))

if __name__ == '__main__':
    num7()
```

## Экранные формы

```
C:\Users\kroll\PycharmProjects\BKIT\lab3-4\venv\Scripts\python.exe C:\Users\kroll\PycharmProjects\BKIT\lab3-4\main.py
Задача №1
Ковер
Диван для отдыха
{'title': 'Диван для отдыха', 'price': 5300}

Задача №2
4 5 4 2

Задача №3
a
A
b
B
StopIteration error.
Задача №4
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Задача №5
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
Задача №6
Time of work: 2.513941764831543
Time of work: 2.503169536590576
Задача №7
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
Гхимик-эксперт
```

ASIC специалист
JavaScript разработчик
RTL специалист
Web-программист
Web-разработчик
web-разработчик
Автожестянщик
Автоинструктор
Автомаляр
автомойщик
Автомойщик
Автор студенческих работ по различным дисциплинам
Автослесарь
автослесарь
Автослесарь - моторист
Автоэлектрик
Агент
Агент банка
Агент нпф
агент по гос. закупкам недвижимости
Агент по гос. закупкам недвижимости
Агент по недвижимости
Агент по недвижимости (стажер)
Агент по недвижимости / Риэлтор
Агент по привлечению юридических лиц
Агент по продажам (интернет, ТВ, телефония) в ПАО Ростелеком в населенных пунктах Амурской области: г. Благовещенск, г. Белогорск, г. Свободный, г. Шимановск, г. Зея, г. Тында
Агент торговый
агрегатчик-топливник KOMATSU
агроном
Агроном
Агроном по защите растений