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

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

Курс «Парадигмы и конструкции языков программирования»
Отчет по лабораторной работе №2
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-36Б
Ковалев Е.А.

Проверил:
Преподаватель каф. ИУ5
Нардид А.Н.

2024 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.

## 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`

## 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.

## 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.

## 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.

## 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.

# Текст программы

```python
#field.py

def field(items, *args):
    assert len(args) > 0

    for item in items:
        if len(args) == 1:
            res = item.get(args[0])
            if res != None:
                yield res
        else:
            result = {arg: item.get(arg) for arg in args if item.get(arg) != None}
            if result:
                yield result

if __name__ == '__main__':
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    print(list(field(goods, 'title')))
    print(list(field(goods, 'title', 'price')))
```

```python
#gen_random.py

import random

def gen_random(count, min, max):
    for i in range(count):
        yield random.randint(min, max)

if __name__ == '__main__':
    for number in gen_random(5, 1, 3):
        print(number)
```

```python
#unique.py

from gen_random import gen_random

class Unique:
    def __init__(self, items, **kwargs):
        self.items = items
        self.values = set()
        self.ignore_case = kwargs.get('ignore_case', False)

    def __iter__(self):
        return self
```

```
    def __next__(self):
        for item in self.items:
            if (not self.ignore_case and item not in self.values) or (self.ignore_case
and str(item).lower() not in self.values):
                self.values.add(item if not self.ignore_case else str(item).lower())
                return item
        raise StopIteration

if __name__ == '__main__':
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(list(Unique(data)))

    data = gen_random(10, 1, 3)
    print(list(Unique(data)))

    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print(list(Unique(data)))
    print(list(Unique(data, ignore_case=True)))
```

```
#sort.py

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

```
#print_result.py

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

```
#cm_timer.py

import time
from contextlib import contextmanager

class cm_timer_1:
    def __enter__(self):
        self.start = time.time()
        return self

    def __exit__(self, type, value, traceback):
        self.end = time.time()
        print(f"time: {self.end - self.start:.2f}")

@contextmanager
def cm_timer_2():
    start = time.time()
    yield
    end = time.time()
    print(f"time: {end - start:.2f}")

if __name__ == '__main__':
    with cm_timer_1():
        time.sleep(5.5)
    with cm_timer_2():
        time.sleep(5.5)
```

```python
#process_data.py

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

path = "data_light.json"

with open(path, encoding='utf-8') as f:
    data = json.load(f)

@print_result
def f1(arg):
    return sorted(list(Unique(field(arg, 'job-name'))), key = str.lower)

@print_result
def f2(arg):
    return list(filter(lambda x: (x.lower()).startswith("программист"), arg))

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

@print_result
def f4(arg):
    salaries = gen_random(len(arg), 100000, 200000)
    return [f"{profession}, зарплата {salary} руб." for profession, salary in zip(arg,
salaries)]

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

# Экранные формы с примерами выполнения программы

```
C:\Users\111\Desktop\lab_python_fp>python3 field.py
['Ковер', 'Диван для отдыха']
[{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}

C:\Users\111\Desktop\lab_python_fp>python3 gen_random.py
3
2
2
2
1

C:\Users\111\Desktop\lab_python_fp>python3 unique.py
[1, 2]
[2, 1, 3]
['a', 'A', 'b', 'B']
['a', 'b']

C:\Users\111\Desktop\lab_python_fp>python3 sort.py
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]

C:\Users\111\Desktop\lab_python_fp>python3 print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

C:\Users\111\Desktop\lab_python_fp>python3 cm_timer.py
time: 5.50
time: 5.50

C:\Users\111\Desktop\lab_python_fp>python3 process_data.py
f1
1С программист
2-ой механик
3-ий механик
4-ый механик
4-ый электромеханик
[химик-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
```

```
Электромонтер по эксплуатации и ремонту оборудования
электромонтер станционного телевизионного оборудования
Электронщик
электросварщик
Электросварщик  на полуавтомат
Электросварщик на автоматических и полуавтоматических машинах
электросварщик на автоматических и полуавтоматических машинах
Электросварщик ручной сварки
Электросварщики ручной сварки
Электрослесарь (слесарь) дежурный и по ремонту оборудования, старший
Электрослесарь по ремонту и обслуживанию автоматики и средств измерений электростанций
Электрослесарь по ремонту оборудования в карьере
Электроэрозионист
Эндокринолог
Энергетик
Энергетик литейного производства
энтомолог
Юрисконсульт
юрисконсульт
юрисконсульт 2 категории
Юрисконсульт. Контрактный управляющий
Юрист
юрист
Юрист (специалист по сопровождению международных договоров, английский - разговорный)
Юрист волонтер
Юристконсульт
f2
Программист
программист
Программист / Senior Developer
Программист 1С
программист 1С
Программист C#
Программист C++
Программист C++/C#/Java
Программист/ Junior Developer
Программист/ технический специалист
Программистр-разработчик  информационных систем
f3
Программист с опытом Python
программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1С с опытом Python
программист 1С с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
Программист/ Junior Developer с опытом Python
Программист/ технический специалист с опытом Python
Программистр-разработчик  информационных систем с опытом Python
f4
Программист с опытом Python, зарплата 184319 руб.
программист с опытом Python, зарплата 109575 руб.
Программист / Senior Developer с опытом Python, зарплата 180293 руб.
Программист 1С с опытом Python, зарплата 166244 руб.
программист 1С с опытом Python, зарплата 179188 руб.
Программист C# с опытом Python, зарплата 197440 руб.
Программист C++ с опытом Python, зарплата 134111 руб.
Программист C++/C#/Java с опытом Python, зарплата 184649 руб.
Программист/ Junior Developer с опытом Python, зарплата 100745 руб.
Программист/ технический специалист с опытом Python, зарплата 169223 руб.
Программистр-разработчик  информационных систем с опытом Python, зарплата 187151 руб.
time: 0.89
```