

**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 ИУ5-35Б  
Шакиров Тимур  
Подпись и дата:

Проверил:  
преподаватель каф. ИУ5  
Гапанюк Юрий Евгеньевич  
Подпись и дата:

## **Постановка задачи**

Изучить функциональные возможности языка Python:

1. Реализовать генератор `field`. Генератор `field` последовательно выдает значения ключей словаря.
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.
3. Необходимо реализовать итератор `Unique(данные)`, который принимает на вход массив или генератор и итерируется по элементам, пропуская дубликаты.
4. Дан массив `1`, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив `2`, который содержит значения массива `1`, отсортированные по модулю в порядке убывания. Сортировку необходимо осуществлять с помощью функции `sorted`.
5. Необходимо реализовать декоратор `print_result`, который выводит на экран результат выполнения функции.
6. Необходимо написать контекстные менеджеры `cm_timer_1` и `cm_timer_2`, которые считают время работы блока кода и выводят его на экран.
7. Применим полученные навыки для решения следующей задачи:
  - 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 С# 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 С# с опытом Python, зарплата 137287 руб. Используйте `zip` для обработки пары специальность — зарплата.

## Текст программы

*field.py*

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

    if len(args) == 1:
        result = (item[args[0]] for item in items if args[0] in item and item[args[0]]
is not None)
        yield ', '.join(result)
    else:
        for item in items:
            filtered_item = {}
            include_item = False
            for field_name in args:
                if field_name in item and item[field_name] is not None:
                    filtered_item[field_name] = item[field_name]
                    include_item = True
            if include_item:
                yield filtered_item

goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]

gen = field(goods, 'title')
while True:
    try:
        result = next(gen)
        print(result)
    except StopIteration:
        break

print()

gen = field(goods, 'title', 'price')
while True:
    try:
        result = next(gen)
        print(result)
    except StopIteration:
        break

print()

gen = field(goods, 'title', 'color')
while True:
    try:
        result = next(gen)
        print(result)
    except StopIteration:
        break
```

```
print()

gen = field(goods, 'title', 'price', 'color')
while True:
    try:
        result = next(gen)
        print(result)
    except StopIteration:
        break
```

### *gen\_random.py*

```
import random

def gen_random(count, minimum, maximum):
    for _ in range(count):
        yield random.randint(minimum, maximum)

random_numbers = gen_random(6, 1, 50)
for number in random_numbers:
    print(number, end=' ')
```

### *unique.py*

```
import random

class Unique:
    def __init__(self, data, **kwargs):
        self.data = data
        self.ignore_case = kwargs.get('ignore_case', False)
        self.seen = set()
        self.iterator = iter(data)

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            try:
                item = next(self.iterator)
            except StopIteration:
                raise StopIteration
            key = item.lower() if self.ignore_case and isinstance(item, str) else item
            if key not in self.seen:
                self.seen.add(key)
                return item

def gen_random(count, minimum, maximum):
    for _ in range(count):
        yield random.randint(minimum, maximum)
```

```

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
unique_data = Unique(data)
for item in unique_data:
    print(item, end=' ')

print()

random_numbers = gen_random(10, 1, 3)
unique_numbers = Unique(random_numbers)
for number in unique_numbers:
    print(number, end=' ')

```

### *sort.py*

```

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

if __name__ == '__main__':
    print(data)

    # Без использования lambda-функции
    result = sorted(data, key=abs, reverse=True)
    print(result)

```

### *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()
    print()
    test_2()
    print()
    test_3()
    print()
    test_4()

```

### *cm\_timer.py*

```

import time
from contextlib import contextmanager

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

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"time: {elapsed_time}")

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    end_time = time.time()
    elapsed_time = end_time - start_time
    print(f"time: {elapsed_time}")

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

    with cm_timer_2():
        time.sleep(3.5)

```

## *process\_data.py*

```
import json
import functools
import time
import sys
from random import randint

# Путь к файлу с данными
path = "C:/Users/shaki/OneDrive/Programming/Python/Lab2/data_light.json"

# Загрузка данных из JSON файла
with open(path, encoding='utf-8') as f:
    data = json.load(f)

# Декоратор для печати результатов
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

# Контекстный менеджер для засечения времени
class cm_timer_1:
    def __enter__(self):
        self.start_time = time.time()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        end_time = time.time()
        elapsed_time = end_time - self.start_time
        print(f"time: {elapsed_time}")

# Функция f1 - сортировка и уникальность профессий
@print_result
def f1(data):
    return sorted(set(item['job-name'].lower() for item in data))

# Функция f2 - фильтрация профессий, начинающихся с "программист"
@print_result
def f2(data):
    return list(filter(lambda x: x.startswith('программист'), data))
```



```
# Функция f3 - добавление "с опытом Python" к профессиям
@print_result
def f3(data):
    return list(map(lambda x: x + ', с опытом Python', data))

# Функция f4 - генерация зарплат для профессий
@print_result
def f4(data):
    salaries = [f"{item[0]}, зарплата {item[1]} руб." for item in zip(data, (randint(100000,
200000) for _ in range(len(data))))]
    return salaries

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

## Анализ результатов тестирования

### 1) Задание 1

```
Ковер, Диван для отдыха

{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}

{'title': 'Ковер', 'color': 'green'}
{'title': 'Диван для отдыха', 'color': 'black'}

{'title': 'Ковер', 'price': 2000, 'color': 'green'}
{'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
PS C:\Users\shaki\OneDrive\Programming\Python>
```

### 2) Задание 2

```
PS C:\Users\shaki\OneDrive\Programming\Python>
50 9 40 5 48 48
PS C:\Users\shaki\OneDrive\Programming\Python>
```

### 3) Задание 3

```
1 2
2 3
a A b B
a b
PS C:\Users\shaki\OneDrive\Programming\Python>
```

### 4) Задание 4

```
[4, -30, 100, -100, 123, 1, 0, -1, -4]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
PS C:\Users\shaki\OneDrive\Programming\Python>
```

### 5) Задание 5

```
!!!!!!!
test_1
1

test_2
iu5

test_3
a = 1
b = 2

test_4
1
2
PS C:\Users\shaki\OneDrive\Programming\Python>
```

### 6) Задание 6

```
time: 5.501317262649536
time: 3.500598907470703
```

## 7) Задание 7

```
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 2 категории
юрисконсульт. контрактный управляющий
юрист
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 189377 руб.
программист / senior developer, с опытом Python, зарплата 102699 руб.
программист 1с, с опытом Python, зарплата 167056 руб.
программист с#, с опытом Python, зарплата 179943 руб.
программист с++, с опытом Python, зарплата 156044 руб.
программист с++/с#/java, с опытом Python, зарплата 120006 руб.
программист/ junior developer, с опытом Python, зарплата 150403 руб.
программист/ технический специалист, с опытом Python, зарплата 129399 руб.
программист-разработчик информационных систем, с опытом Python, зарплата 137991 руб.
time: 0.3128833770751953
```

## Вывод

Я изучил функциональные возможности языка Python и применил полученные навыки при решении задачи парсинга .json файла