

**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 №4
«Функциональные возможности языка Python»

Выполнил:
студент группы ИУ5-35Б
Ходырев Роман
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 `l`, содержащий положительные и отрицательные числа. Необходимо одной строкой кода вывести на экран массив `l2`, который содержит значения массива `l`, 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

```
goods = [  
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},  
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}  
]  
  
def field(items, *args):  
    assert len(args) > 0  
  
    for item in items:  
        if len(args) == 1:  
            key = args[0]  
            if key in item and item[key] is not None:  
                yield item[key]  
        else:  
            result = {}  
            has_value = False # Флаг для проверки, есть ли хотя бы одно непустое поле  
            for key in args:  
                if key in item and item[key] is not None:  
                    result[key] = item[key]  
                    has_value = True  
            if has_value:  
                yield result
```

```

if __name__ == "__main__":
    for item in field(goods, 'title'):
        print(item, end = " ")
    print()
    for item in field(goods, 'title', 'price'):
        print(item)

```

gen_random.py

```

from random import randint

def gen_random(num, min, max):
    if num == 0:
        return None
    else:
        times = 0
        while times!=num:
            yield randint(min, max)
            times += 1

if __name__ == "__main__":
    result = [x for x in gen_random(5, 1, 3)]
    print(result)

```

unique.py

```

class unique(object):
    def __init__(self, items, **kwargs):
        if "ignore_case" not in kwargs.keys():
            self._ignore_case = False
        else:
            self._ignore_case = kwargs["ignore_case"]
        self.uniq_items = []
        for i in items:
            if self._ignore_case:
                temp = i.lower()
            else:
                temp = i
            if temp not in self.uniq_items:
                self.uniq_items.append(temp)
        self.index = 0

    def __next__(self):
        if self.index != len(self.uniq_items):
            print(self.uniq_items[self.index])
            self.index+= 1

    def __iter__(self):
        return self

data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]

```

```

result = iter(unique(data))
next(result)
next(result)
next(result)
next(result)

data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
result = iter(unique(data, ignore_case = True))
next(result)
next(result)
next(result)
next(result)

```

sort.py

```

def sort_key(number):
    return abs(number)

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

result = sorted(data, key=sort_key, reverse=True)
print(result)
result = sorted(data, key=lambda x:abs(x), reverse=True)
print(result)

```

print_result.py

```

def print_result(func):
    def wrapper():
        print(func.__name__)
        i = func()
        if type(i) == type(dict()):
            for key, value in i.items():
                print(key, '=', value)
        elif type(i) == type(list()):
            for value in i:
                print(value)
        else:
            print(i)
    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

class cm_timer_1():
    def __init__(self):
        self.start = 0.0
        self.end = 0.0

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

    def __exit__(self, exc_type, exc_value, exc_traceback):
        self.end = time.time()
        print('Время выполнения: {} секунд.'.format(self.end - self.start))

with cm_timer_1():
    time.sleep(5.5)

```

process_data.py

```

import json
from random import randint
from cm_timer import cm_timer_1

# Путь к файлу с данными
PATH = "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

# 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

```
PS F:\Рабочий стол\3 сем\PCPL\Labs> & C:/Users/
Ковер Диван для отдыха
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха', 'price': 5300}
PS F:\Рабочий стол\3 сем\PCPL\Labs>
```

2) Задание 2

```
[3, 1, 2, 1, 3]
PS F:\Рабочий стол\3 сем\PCPL\Labs>
```

3) Задание 3

```
[3, 1, 2, 1, 3]
PS F:\Рабочий стол\3 сем\PCPL\Labs> &
1
2
a
b
PS F:\Рабочий стол\3 сем\PCPL\Labs> [
```

4) Задание 4

```
PS F:\Рабочий стол\3 сем\PCPL\Labs> & C:/Us
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
PS F:\Рабочий стол\3 сем\PCPL\Labs> [
```

5) Задание 5

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

6) Задание 6

```
PS F:\Рабочий стол\3 сем\PCPL\Labs> & C:/Us
Время выполнения: 5.500159740447998 секунд.
PS F:\Рабочий стол\3 сем\PCPL\Labs> [
```


7) Задание 7

```
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
программистр-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
программистр-разработчик информационных систем, с опытом Python
f4
программист, с опытом Python, зарплата 136942 руб.
программист / senior developer, с опытом Python, зарплата 144809 руб.
программист 1с, с опытом Python, зарплата 187425 руб.
программист с#, с опытом Python, зарплата 121735 руб.
программист с++, с опытом Python, зарплата 150732 руб.
```

Вывод

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