МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по лабораторной работе № 3
“Функциональные возможности языка Python.”

Выполнил:
Студент группы ИУ5-36Б
Левочкин В.В.

Преподаватель:
Нардид. А.Н.

Москва 2025

Задание

Задание:

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.

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.

Cm_timer.py

```
importtime
importcontextlib
fromdatetimeimportdatetime



classcm_timer_1:
    def__enter__(self):
        self.start_time=time.time()
        returnself

    def__exit__(self, exc_type, exc_val, exc_tb):
        elapsed_time=time.time() -self.start_time
        print(f"time: {elapsed_time:.1f}")



@contextlib.contextmanager
defcm_timer_2():
    start_time=time.time()
    yield
    elapsed_time=time.time() -start_time
    print(f"time: {elapsed_time:.1f}")



if__name__=="__main__":
    print("Test cm_timer_1:")
    withcm_timer_1():
        time.sleep(1.5)

    print("\nTest cm_timer_2:")
    withcm_timer_2():
        time.sleep(0.5)
```

Data_light.json

```
[
    {"job-name": "Программист C#", "salary": 120000, "city": "Москва"},
    {"job-name": "программист Python", "salary": 150000, "city": "Санкт-
Петербург"},
    {"job-name": "Аналитик данных", "salary": 110000, "city": "Москва"},
    {"job-name": "Программист Java", "salary": 130000, "city": "Екатеринбург"},
    {"job-name": "Дизайнер UI/UX", "salary": 90000, "city": "Москва"},
    {"job-name": "Программист C++", "salary": 140000, "city": "Новосибирск"},
    {"job-name": "Менеджер проектов", "salary": 100000, "city": "Москва"}
]
```

Field.py

```
deffield(items, *args):


    assertlen(args) >0
```

```
    foriteminitems:
        iflen(args) ==1:
            field_name=args[0]
            iffield_nameinitemanditem[field_name] isnotNone:
                yielditem[field_name]
        else:
            result={}
            has_values=False

            forfield_nameinargs:
                iffield_nameinitemanditem[field_name] isnotNone:
                    result[field_name] =item[field_name]
                    has_values=True

            ifhas_values:
                yieldresult



if__name__=="__main__":
    goods=[
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'},
        {'title': None, 'price': 1500},
        {'price': 3000, 'color': 'blue'}
    ]

    print("Test 1 - один аргумент:")
    fortitleinfield(goods, 'title'):
        print(title)

    print("\nTest 2 - несколько аргументов:")
    foriteminfield(goods, 'title', 'price'):
        print(item)
```

Gen_random.py

```
importrandom


defgen_random(num_count, begin, end):


    for_inrange(num_count):
        yieldrandom.randint(begin, end)



if__name__=="__main__":
    print("Test gen_random(5, 1, 3):")
    fornumingen_random(5, 1, 3):
        print(num, end=" ")
    print()
```

Print_result.py

```
defprint_result(func):


    defwrapper(*args, **kwargs):
```

```
        result=func(*args, **kwargs)

        print(func.__name__)

        ifisinstance(result, list):
            foriteminresult:
                print(item)
        elifisinstance(result, dict):
            forkey, valueinresult.items():
                print(f"{key}= {value}")
        else:
            print(result)

        returnresult

    returnwrapper



@print_result
deftest_1():
    return1



@print_result
deftest_2():
    return'iu5'



@print_result
deftest_3():
    return{'a': 1, 'b': 2}



@print_result
deftest_4():
    return[1, 2]



if__name__=='__main__':
    print('!!!!!!!!')
    test_1()
    test_2()
    test_3()
    test_4()
```

Procces_data.py

```
importjson
importsys
from.fieldimportfield
from.gen_randomimportgen_random
from.uniqueimportUnique
from.print_resultimportprint_result
from.cm_timerimportcm_timer_1
```

```
path=sys.argv[1] iflen(sys.argv) >1else"data_light.json"


withopen(path, encoding='utf-8') asf:
    data=json.load(f)


@print_result
deff1(arg):


    professions=list(Unique(field(arg, 'job-name'), ignore_case=True))


    returnsorted(professions, key=lambdax: x.lower())


@print_result
deff2(arg):


    returnlist(filter(lambdax: x.lower().startswith('программист'), arg))


@print_result
deff3(arg):


    returnlist(map(lambdax: f"{x}с опытом Python", arg))


@print_result
deff4(arg):


    salaries=list(gen_random(len(arg), 100000, 200000))
    result=[]
    forprofession, salaryinzip(arg, salaries):
        result.append(f"{profession}, зарплата {salary}руб.")
    returnresult


if__name__=='__main__':


    withcm_timer_1():
        f4(f3(f2(f1(data))))
```

Sort.py

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


if__name__=='__main__':

    result=sorted(data, key=lambdax: -abs(x))
    print("С lambda:", result)



    defsort_key(x):
        return-abs(x)

    result_without_lambda=sorted(data, key=sort_key)
    print("Без lambda:", result_without_lambda)
```

Unique.py

```
classUnique(object):


    def__init__(self, items, **kwargs):


        self.items=iter(items)
        self.ignore_case=kwargs.get('ignore_case', False)
        self.seen=set()

    def__iter__(self):
        returnself

    def__next__(self):
        whileTrue:
            item=next(self.items)



            ifself.ignore_caseandisinstance(item, str):
                check_item=item.lower()
            else:
                check_item=item



            ifcheck_itemnotinself.seen:
                self.seen.add(check_item)
                returnitem




if__name__=="__main__":
    print("Test 1 - числа:")
    data=[1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    foriteminUnique(data):
        print(item, end=" ")

    print("\n\nTest 2 - строки без ignore_case:")
    data=['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
```

```
    for item in Unique(data):
        print(item, end=" ")

    print("\n\nTest 3 - строки с ignore_case=True:")
    for item in Unique(data, ignore_case=True):
        print(item, end=" ")
    print()
```

Вывод программы

```
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/unique.py
Test 1 - числа:
1 2

Test 2 - строки без ignore_case:
a A b B

Test 3 - строки с ignore_case=True:
a b
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/sort.py
С lambda: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
Без lambda: [123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/process_data.py
f1
Аналитик данных
Дизайнер UI/UX
Менеджер проектов
Программист C#
Программист C++
Программист Java
программист Python
f2
f2
Программист C#
Программист C++
Программист Java
программист Python
f3
Программист C# с опытом Python
Программист C++ с опытом Python
Программист Java с опытом Python
программист Python с опытом Python
f4
Программист C# с опытом Python, зарплата 172966 руб.
Программист C++ с опытом Python, зарплата 179177 руб.
Программист Java с опытом Python, зарплата 195984 руб.
```

```
time: 0.0
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/print_result.py
!!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/gen_random.py
Test gen_random(5, 1, 3):
1 1 2 2 1
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/field.py
Test 1 - один аргумент:
Ковер
Диван для отдыха

Test 2 - несколько аргументов:
{'title': 'Ковер', 'price': 2000}
{'title': 'Диван для отдыха'}
{'price': 1500}
{'price': 3000}
@vasyteri ➜/workspaces/Python/Lab_3 (main) $ /usr/bin/python3 /workspaces/Python/Lab_3/cm_timer.py
Test cm_timer_1:
time: 1.5

Test cm_timer_2:
time: 0.5
```