

**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 ИУ5-32Б

Лялько Никита

Подпись и дата:

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

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_fr. Решение каждой задачи должно располагаться 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 количество 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 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 выводиться 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 повторов (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 С# 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_fr:

field.py

```
1 def field(items, *args):
2     assert len(args) > 0
3     if not (len(items)):
4         return
5     for arg in args:
6         if not (arg in items[0].keys()):
7             return
8     if len(args) == 1:
9         for item in items:
10             yield item[args[0]]
11     else:
12         for item in items:
13             record = {}
14             for arg in args:
15                 record[arg] = item[arg]
16             yield record
17
18
19 def main():
20     goods = [
21         {'title': 'Ковер', 'price': 2000, 'color': 'green'},
22         {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
23     ]
24
25     gen1 = field(goods, 'title') # должен выдавать 'Ковер', 'Диван для отдыха'
26     gen2 = field(goods, 'title',
27                 'price') # должен выдавать {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха', 'price': 5300}
28
29     for i in gen1:
30         print(i, end=' ')
31     print()
32
```

gen_random.py


```

1  import random
2  def gen_random(num_count, begin, end):
3      assert begin < end
4      assert num_count >= 0
5
6      for i in range(num_count):
7          yield random.randint(begin, end)
8
9  gen1 = gen_random(5, 1, 3)
10 gen2 = gen_random(3, 5, 25)
11
12 def main():
13
14     for i in gen1:
15         print(i, end=' ')
16     print()
17
18     for i in gen2:
19         print(i, end=' ')
20     print()
21
22 if __name__ == "__main__":
23     main()

```

unique.py

```

1  class Unique(object):
2      def __init__(self, items, ignore_case = False):
3          self.ignore_case = ignore_case
4          self.data = items
5          self.occured = set()
6          self.index = -1
7          pass
8
9      def __next__(self):
10         while True:
11             self.index += 1
12             if self.index >= len(self.data):
13                 raise StopIteration
14             if self.check():
15                 current = self.data[self.index]
16                 self.occured.add(current)
17                 return current
18             pass
19
20     def __iter__(self):
21         return self
22
23     def check(self):
24         el = self.data[self.index]
25         if self.ignore_case:
26             if type(el) == str:
27                 return not((el.lower() in self.occured) or (el.upper() in self.occured))
28             else:
29                 return not(el in self.occured)
30         else:
31             return not(el in self.occured)
32         pass
33
34
35 def main():
36     for i in Unique(['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']):
37         print(i)
38
39 if __name__ == "__main__":
40     main()

```

sort.py

```
1  def sort_wo_lambda(data):
2      return sorted(data, key=abs, reverse=True)
3
4  def sort_lambda(data):
5      return sorted(data, key=lambda x: abs(x), reverse=True)
6
7  def main():
8      data = [4, -30, 30, 100, -100, 123, 1, 0, -1, -4]
9      print(sort_lambda(data))
10     print(sort_wo_lambda(data))
11
12  if __name__ == "__main__":
13     main()
```

print_result.py

```
1 def print_result(fun):
2     def decorator(lst = [], *args, **kwargs):
3         print(fun.__name__)
4
5         if len(lst):
6             result = fun(lst, *args, **kwargs)
7         else:
8             result = fun(*args, **kwargs)
9
10        if type(result) == dict:
11            for key, el in result.items():
12                print(f'{key} = {el}')
13
14        elif type(result) == list:
15            for i in result:
16                print(i)
17        else:
18            print(result)
19
20        return result
21
22    return decorator
23
24
25    @print_result
26    def test_1():
27        return 1
28
29
30    @print_result
31    def test_2():
32        return 'iu5'
```

```

33
34
35     @print_result
36 def test_3():
37     return {'a': 1, 'b': 2}
38
39
40     @print_result
41 def test_4():
42     return [1, 2]
43
44
45 def main():
46     print('!!!!!!!')
47     test_1()
48     test_2()
49     test_3()
50     test_4()
51
52 if __name__ == "__main__":
53     main()

```

cm_timer.py

```

1  from time import *
2  from contextlib import *
3
4
5  class cm_timer_1():
6
7      def __init__(self):
8          self.start = 0
9          self.finish = 0
10
11     def __enter__(self):
12         self.start = time()
13
14     def __exit__(self, ex_type, ex_value, ex_traceback):
15         self.finish = time()
16         print("time: ", self.finish - self.start)
17
18     @contextmanager
19     def cm_timer_2():
20         start = time()
21         yield None
22         finish = time()
23         print("time: ", finish - start)
24
25     def main():
26         with cm_timer_1():
27             sleep(1.5)
28         with cm_timer_2():
29             sleep(2)
30
31     if __name__ == "__main__":
32         main()

```

Финальная задача
main.py

```

1 import json
2
3 from lab_python_fp.print_result import print_result
4 from lab_python_fp.gen_random import gen_random
5 from lab_python_fp.cm_timer import cm_timer_1
6 from lab_python_fp.unique import Unique
7 from lab_python_fp.field import field
8
9 path = 'data_light.json'
10
11 with open(path, encoding='utf-8') as f:
12     data = json.load(f)
13
14
15 @print_result
16 def f1(arg):
17     return sorted(list(j for j in Unique(list(i for i in field(arg, 'job-name')))), key=lambda x: x.lower())
18
19
20 @print_result
21 def f2(arg):
22     return list(filter(lambda s: (s.split())[0].lower() == 'программист', arg))
23
24
25 @print_result
26 def f3(arg):
27     return list(map(lambda lst: lst + ' с опытом Python', arg))
28
29
30 @print_result
31 def f4(arg):
32     return dict(zip(arg, ['зарплата' + str(i) + ' руб.' for i in gen_random(len(arg), 100000, 200000)]))
33
34
35 def main():
36     with cm_timer_1():
37         f4(f3(f2(f1(data))))
38
39
40 main()

```

Экранные формы

Задача 1

```

C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\field.py
Ковер Диван для отдыха
{'title': 'Ковер', 'price': 2000} {'title': 'Диван для отдыха', 'price': 5300}

Process finished with exit code 0

```

Задача 2

```
C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\gen_random.py
1 1 1 1 2
22 7 8

Process finished with exit code 0
```

Задача 3

```
C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\unique.py
a
A
b
B

Process finished with exit code 0
```

Задача 4

```
C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\sort.py
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 30, 4, -4, 1, -1, 0]

Process finished with exit code 0
```

Задача 5

```
C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\print_result.py
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0
```

Задача 6

```
C:\Users\lyalk\PycharmProjects\pythonProject4\venv\Scripts\python.exe C:\Users\lyalk\PycharmProjects\pythonProject4\lab_python_fp\cm_timer.py
time: 1.507291555404663
time: 2.0099456310272217

Process finished with exit code 0
```

Задача 7


```
f2
Программист
программист
Программист / Senior Developer
Программист 1C
программист 1C
Программист C#
Программист C++
Программист C++/C#/Java
f3
Программист с опытом Python
программист с опытом Python
Программист / Senior Developer с опытом Python
Программист 1C с опытом Python
программист 1C с опытом Python
Программист C# с опытом Python
Программист C++ с опытом Python
Программист C++/C#/Java с опытом Python
f4
Программист с опытом Python = зарплата 162503 руб.
программист с опытом Python = зарплата 173953 руб.
Программист / Senior Developer с опытом Python = зарплата 183252 руб.
Программист 1C с опытом Python = зарплата 189282 руб.
программист 1C с опытом Python = зарплата 158078 руб.
Программист C# с опытом Python = зарплата 112752 руб.
Программист C++ с опытом Python = зарплата 189351 руб.
Программист C++/C#/Java с опытом Python = зарплата 118834 руб.
time: 0.02108168601989746
```

Фрагмент вывода для f1:

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
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