

МГТУ им. Н. Э. Баумана

Кафедра «Системы обработки информации и управления»

Лабораторная работа №3

по курсу «Разработка интернет-приложений»

«Функциональные возможности языка Python»

Выполнил:

Алёшин А.Д., ИУ5-53Б _____

Преподаватель:

Гапанюк Ю.Е. _____

Москва, 2021 год

Задание:

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.

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:

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 выводиться 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 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.

Задача 1 (файл field.py)



```

1 def field(items, *args):
2     assert len(args) > 0
3     for i in items:
4         if len(args) == 1: # если передан только один аргумент, то
5             if i.get(args[0]): # метод get() возвращает значение для
6                 yield i[args[0]]
7         else:
8             res = {}
9             for a in args:
10                 res[a] = i.get(a)
11             if len(res.items()) != 0:
12                 yield res
13             # print(res)
14
15
16 goods = [
17     {'title': 'Ковёр', 'price': 2000, 'color': 'green'},
18     {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
19 ]
20 result = field(goods, 'title', 'color')
21 for r in result:
22     print(r)
23

```

Экранная форма:



```

C:\Users\79772\AppData\Local\Programs\Python\Python
{'title': 'Ковёр', 'color': 'green'}
{'title': 'Диван для отдыха', 'color': 'black'}

Process finished with exit code 0

```

Задача 2 (файл gen_random.py)

```
main.py x cm_timer.py x data_light.json x
1 from random import randint
2
3
4 def gen_random(n, mi, ma):
5     for i in range(n):
6         yield randint(mi, ma)
7
8
9 rand = gen_random(5, 3, 10)
10 for r in rand:
11     print(r)
```

Экранная форма:

```
gen_random x
C:\Users\79772\AppData\Local\Programs\Py
5
3
3
3
3
7
Process finished with exit code 0
```

Задача 3 (файл unique.py)

```
main.py x cm_timer.py x unique.py x data_light.json x field.py x
1 class Unique:
2     """Итератор, оставляющий только уникальные значения."""
3
4     def __init__(self, data, **kwargs):
5         self.used_elements = set()
6         self.data = data
7         self.index = 0
8         if 'ignore_case' not in kwargs:
9             self.ignore_case = False
10        else:
11            self.ignore_case = kwargs['ignore_case']
12
13    def __iter__(self):
14        return self
```

```

16     def __next__(self):
17         while True:
18             if self.index >= len(self.data):
19                 raise StopIteration
20             else:
21                 current = self.data[self.index]
22                 self.index = self.index + 1
23                 if self.ignore_case:
24                     if current.lower() not in self.used_elements:
25                         self.used_elements.add(current.lower())
26                         return current
27                 else:
28                     if current not in self.used_elements:
29                         # Добавление в множество производится
30                         # с помощью метода add
31                         self.used_elements.add(current)
32                         return current
33
34
35     lst2 = [1, 3, 2, 3, 2, 1, 4, 3, 3, 3]
36     data2 = ['a', 'A', 'c', 'C', 'C', 'B', 'b', 'b']
37     for i in Unique(lst2):
38         print(i)
39     print("-----")
40     for d2 in Unique(data2, ignore_case=False):
41         print(d2)
42     print("-----")
43     for d3 in Unique(data2, ignore_case=True):
44         print(d3)
45

```

Экранная форма:

```

1
3
2
4
7
-----
a
A
c
C
B
b
-----
a
c
B

Process finished with exit code 0

```

Задача 4 (файл sort.py)

```
main.py × cm_timer.py × unique.py × sort.py × data_light.json × field.py ×
1
2 data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
3
4 result = sorted(data, key=abs, reverse=True)
5 print(result)
6
7 result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
8 print(result_with_lambda)
9
```

Экранная форма:

```
Run: sort ×
C:\Users\79772\AppData\Local\Programs\Python
[123, 100, -100, -30, 4, -4, 1, -1, 0]
[123, 100, -100, -30, 4, -4, 1, -1, 0]
Process finished with exit code 0
```

Задача 5 (файл print_result.py)

```
main.py × cm_timer.py × unique.py × sort.py × print_result.py ×
1 # Здесь должна быть реализация декоратора
2 def print_result(func_to_decorate):
3     def decorate_func(*args):
4         print(func_to_decorate.__name__)
5         f = func_to_decorate(*args)
6         if isinstance(f, list):
7             for v in f:
8                 print(v)
9             return f
10        elif isinstance(f, dict):
11            for key, value in f.items():
12                print("{} = {}".format(key, value))
13            return f
14        else:
15            print(f)
16            return f
17
```

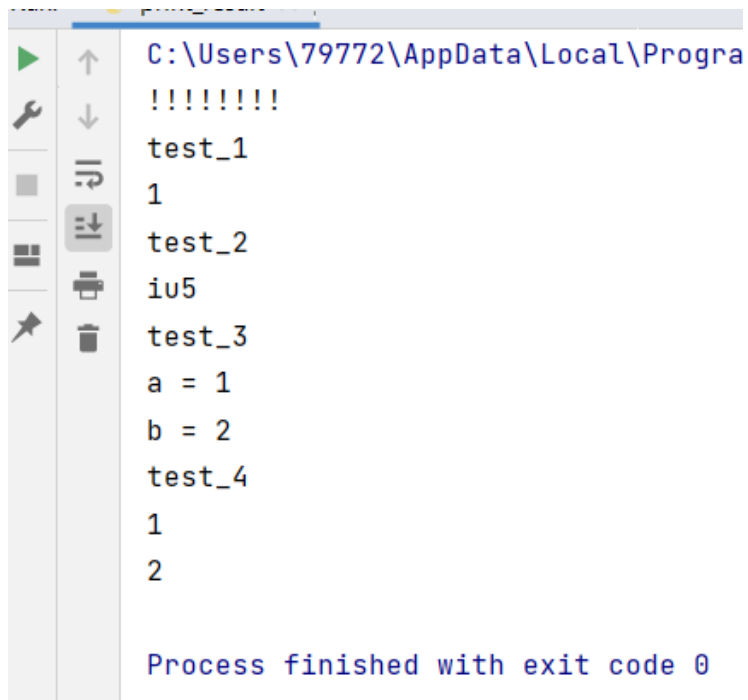


```

18     return decorate_func
19
20     @print_result
21     def test_1():
22         return 1
23
24     @print_result
25     def test_2():
26         return 'iu5'
27
28     @print_result
29     def test_3():
30         return {'a': 1, 'b': 2}
31
32     @print_result
33     def test_4():
34         return [1, 2]
35
36     print('!!!!!!!')
37     test_1()
38     test_2()
39     test_3()
40     test_4()

```

Экранная форма:



```

C:\Users\79772\AppData\Local\Progra
!!!!!!!
test_1
1
test_2
iu5
test_3
a = 1
b = 2
test_4
1
2

Process finished with exit code 0

```

Задача 6 (файл cm_timer.py)

```
main.py x cm_timer.py x unique.py x sort.py x print_result.py x
1  from contextlib import contextmanager
2  import time
3
4  class Cm_timer_1:
5      def __init__(self, before_ms, after_ms):
6          self.before_ms = before_ms
7          self.after_ms = after_ms
8
9      def __enter__(self):
10         print(self.before_ms)
11         self.time = time.time()
12         return self.time
13
14     def __exit__(self, exc_type, exc_val, exc_tb):
15         if exc_type is not None:
16             print(exc_type, exc_val, exc_tb)
17         else:
18             print("time1:", time.time() - self.time)
19             print(self.after_ms)
20
21     before_ms = "Сообщение при входе в контекстный менеджер на основе класса"
22     after_ms = "Сообщение при выходе из контекстного менеджера на основе класса"
23     with Cm_timer_1(before_ms, after_ms) as cm_object:
24         time.sleep(5.5)
25
26
27     @contextmanager
28     def cm_timer_2():
29         t = time.time() # начальное время
30         yield t
31         print("time2:", time.time() - t) # текущее время - начальное
32
33     with cm_timer_2():
34         time.sleep(5.5)
```

Экранная форма:

```
Run: cm_timer x
C:\Users\79772\AppData\Local\Programs\Python\Python39\python.exe "
Сообщение при входе в контекстный менеджер на основе класса
time1: 5.514511823654175
Сообщение при выходе из контекстного менеджера на основе класса
time2: 5.500858306884766
Process finished with exit code 0
```

Задача 7 (файл process_data.py)

```
main.py × cm_timer.py × unique.py × sort.py × print_result.py × data_light.json × field.py × gen_random.py × process_data
1 import json
2 from lab_python_fp.gen_random import gen_random
3 import sys
4 from lab_python_fp.cm_timer import Cm_timer_1
5 from lab_python_fp.print_result import print_result
6
7 path = "data_light.json"
8
9 with open(path) as f:
10     data = json.load(f) # метод считывает файл в формате JSON и возвращает объекты Python
11
12 @print_result
13 def f1(arg):
14     return sorted(set([p.lower() for p in arg]), key=str.lower)
15
16 @print_result
17 def f2(arg):
18     return list(filter(lambda x: str.startswith(x, 'программист'), arg))
19
20 @print_result
21 def f3(arg):
22     return list(map(lambda x: x + ' с опытом Python', arg))
23
24 @print_result
25 def f4(arg):
26     t = list(zip(arg, [" зарплата " + str(el) + " py6." for el in list(gen_random(len(arg), 100000, 200000))]))
27     return [e[0] + e[1] for e in t]
28
29 if __name__ == '__main__':
30     before_ms = "Сообщение при входе в контекстный менеджер на основе класса"
31     after_ms = "Сообщение при выходе из контекстного менеджера на основе класса"
32     with Cm_timer_1(before_ms, after_ms) as cm_object:
33         f4(f3(f2(f1([el['job-name'] for el in data])))))
34
```

Экранная форма:

```
Сообщение при входе в контекстный менеджер на основе класса
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
автожестящик
автоинструктор
автомаляр
```

автомойщик
автор студенческих работ по различным дисциплинам
автослесарь
автослесарь - моторист
автоэлектрик
агент
агент банка
агент нпф
агент по гос. закупкам недвижимости
агент по недвижимости
агент по недвижимости (стажер)
агент по недвижимости / риэлтор
агент по привлечению юридических лиц
агент по продажам (интернет, тв, телефония) в пао ростелеком в населенных пунктах амурской области: г. благовещенск, г. белогорск
агент торговый
агрегатчик-топливник komatsu
агроном
агроном по защите растений

И т.д.

юрист (специалист по сопровождению международных договоров, английский - разговорный)
юрист волонтер
юристконсульт
f2
программист
программист / senior developer
программист 1с
программист c#
программист c++
программист c++/c#/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 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
программист-разработчик информационных систем с опытом Python

f4
программист с опытом Python зарплата 161479 руб.
программист / senior developer с опытом Python зарплата 180521 руб.
программист 1с с опытом Python зарплата 132505 руб.
программист c# с опытом Python зарплата 121041 руб.
программист c++ с опытом Python зарплата 182315 руб.
программист c++/c#/java с опытом Python зарплата 171669 руб.
программист/ junior developer с опытом Python зарплата 149993 руб.
программист/ технический специалист с опытом Python зарплата 146235 руб.
программист-разработчик информационных систем с опытом Python зарплата 149055 руб.
time1: 0.03988933563232422
Сообщение при выходе из контекстного менеджера на основе класса

Process finished with `exit` code 0