Федеральное государственное бюджетное образовательное учреждение   
высшего образования

| **Описание: Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана»  (национальный исследовательский университет)***  ***(МГТУ им. Н.Э. Баумана)*** |
| --- | --- |

ФАКУЛЬТЕТ Инженерный бизнес и менеджмент

КАФЕДРА Бизнес-информатика

**Отчет по лабораторной работе №3**

**“Парадигмы и конструкции языков программирования”**

Студент  **В.Д. Зазовский**

**Группа ИБМ3-34Б**

Преподаватель

Москва, 2024

# Отчет по лабораторной работе

Цель лабораторной работы: изучение возможностей функционального программирования в языке Python.

## Задача 1 (field.py)

Описание задания:

Необходимо реализовать генератор field, который последовательно выдает значения ключей словаря.   
Пример:  
goods = [{'title': 'Ковер', 'price': 2000, 'color': 'green'}, {'title': 'Диван для отдыха', 'color': 'black'}]  
field(goods, 'title') -> 'Ковер', 'Диван для отдыха'  
field(goods, 'title', 'price') -> {'title': 'Ковер', 'price': 2000}, {'title': 'Диван для отдыха'}

Код решения:

def field(items, \*args):  
 assert len(args) > 0  
 if len(args) == 1:  
 for item in items:  
 value = item.get(args[0])  
 if value is not None:  
 yield value  
 else:  
 for item in items:  
 result = {key: item[key] for key in args if key in item and item[key] is not None}  
 if result:  
 yield result

## Задача 2 (gen\_random.py)

Описание задания:

Необходимо реализовать генератор gen\_random(количество, минимум, максимум),   
который последовательно выдает заданное количество случайных чисел в диапазоне.

Код решения:

import random  
  
def gen\_random(num\_count, begin, end):  
 for \_ in range(num\_count):  
 yield random.randint(begin, end)

## Задача 3 (unique.py)

Описание задания:

Необходимо реализовать итератор Unique, который итерируется по элементам, пропуская дубликаты.   
Параметр ignore\_case определяет, учитывать ли регистр строк.

Код решения:

class Unique:  
 def \_\_init\_\_(self, items, \*\*kwargs):  
 self.ignore\_case = kwargs.get('ignore\_case', False)  
 self.seen = set()  
 self.items = iter(items)  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 while True:  
 item = next(self.items)  
 comp\_item = item.lower() if self.ignore\_case and isinstance(item, str) else item  
 if comp\_item not in self.seen:  
 self.seen.add(comp\_item)  
 return item

## Задача 4 (sort.py)

Описание задания:

Отсортировать массив по модулю в порядке убывания двумя способами: с использованием lambda и без.

Код решения:

data = [4, -30, 100, -100, 123, 1, 0, -1, -4]  
  
# Без использования lambda  
result = sorted(data, key=abs, reverse=True)  
print(result)  
  
# С использованием lambda  
result\_with\_lambda = sorted(data, key=lambda x: abs(x), reverse=True)  
print(result\_with\_lambda)

## Задача 5 (print\_result.py)

Описание задания:

Реализовать декоратор print\_result, который выводит результат выполнения функции,   
печатая имя функции и результат. Если результат - список, выводить элементы в столбик.

Код решения:

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 k, v in result.items():  
 print(f"{k} = {v}")  
 else:  
 print(result)  
 return result  
 return wrapper

## Задача 6 (cm\_timer.py)

Описание задания:

Написать контекстные менеджеры cm\_timer\_1 и cm\_timer\_2, которые считают время выполнения блока кода.

Код решения:

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\_value, traceback):  
 print(f"time: {time.time() - self.start\_time:.2f}")  
  
@contextmanager  
def cm\_timer\_2():  
 start\_time = time.time()  
 try:  
 yield  
 finally:  
 print(f"time: {time.time() - start\_time:.2f}")

## Задача 7 (process\_data.py)

Описание задания:

Реализовать функции f1, f2, f3, f4 для обработки данных из файла data\_light.json.  
Использовать декоратор print\_result и контекстный менеджер cm\_timer\_1.

Код решения:

import json  
import random  
from print\_result import print\_result  
from cm\_timer import cm\_timer\_1  
  
@print\_result  
def f1(data):  
 return sorted(set(job.lower() for job in data), key=str.lower)  
  
@print\_result  
def f2(data):  
 return list(filter(lambda x: x.lower().startswith("программист"), data))  
  
@print\_result  
def f3(data):  
 return list(map(lambda x: f"{x} с опытом Python", data))  
  
@print\_result  
def f4(data):  
 return [f"{x}, зарплата {random.randint(100000, 200000)} руб." for x in data]