Код программы:

Field.py

```
def field(items, *args):
    assert len(args) > 0
    for item in items:
        result = {}
        for arg in args:
            if item.get(arg) is not None:
                result[arg] = item[arg]
        if len(result) > 0:
            yield result
        elif len(args) == 1:
            yield item.get(args[0])
```

Gen_random.py

```
import random

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randint(begin, end)
```

Unique.py

```
class Unique(object):
    def __init__(self, items, **kwargs):
        self.ignore_case = kwargs.get('ignore_case', False)
        if self.ignore_case:
            self.seen = set()
            self.items = [item.lower() for item in items if item.lower()
not in self.seen and not self.seen.add(item.lower())]
        else:
            self.seen = set()
            self.items = [item for item in items if item not in self.seen
and not self.seen.add(item)]
        self.index = 0

    def __next__(self):
        if self.index < len(self.items):
            result = self.items[self.index]
            self.index += 1
            return result
        raise StopIteration

    def __iter__(self):
        return self
```

Sort.py

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

if __name__ == '__main__':
    result = sorted(data, key=abs, reverse=True)
    print(result)

    result_with_lambda = sorted(data, key=lambda x: abs(x), reverse=True)
    print(result_with_lambda)
```

Print_result.py

```
def print_result(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(f"{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

@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 __enter__(self):
        self.start_time = time.time()

    def __exit__(self, exc_type, exc_val, exc_tb):
        print(f"time: {time.time() - self.start_time}")

from contextlib import contextmanager

@contextmanager
def cm_timer_2():
    start_time = time.time()
    yield
    print(f"time: {time.time() - start_time}")
Procces_data.py
import json
import sys
from lab_python_fp.field import field
from lab_python_fp.unique import Unique
from lab_python_fp.gen_random import gen_random
from lab_python_fp.cm_timer import cm_timer_1, cm_timer_2
from lab_python_fp.print_result import print_result

path = sys.argv[1] if len(sys.argv) > 1 else "data_light.json"

import codecs

with open(path, 'rb') as f:
    data = json.loads(codecs.getdecoder('utf-8')(f.read())[0])

@print_result
def f1(data):
    return sorted(
        list(Unique([item["job-name"] for item in list(field(data, "job-
name"))], ignore_case=True)),
        key=str.lower
    )

@print_result
def f2(data):
    return list(filter(lambda x: x.startswith("программист"), data))

@print_result
def f3(data):
    return list(map(lambda x: x + " с опытом Python", data))

@print_result
def f4(data):
    return list(map(lambda x: x + ", зарплата " + str(next(gen_random(1,
100000, 200000))) + " руб.", data))

if __name__ == '__main__':
    with cm_timer_1():
        f4(f3(f2(f1(data))))
Результат в отдельном файле.
```