

Государственный Университет Молдовы  
Факультет Математики и Информатики  
Департамент Информатики

**Славов Константин, группа I2302**

**Отчет**  
по дисциплине “Программирование в Python”

Руководитель: \_\_\_\_\_ Плешка Наталья, лектор  
(подпись)

Автор: \_\_\_\_\_  
(подпись)

Кишинев, 2024

Альтернативное задание ко второй аттестации по курсу “Программирование на Python”, отчет по проделанной работе (3 вариант).

**Условие программы:** Написать код, который должен позволить пользователю выбрать из меню опцию:

- Записать данные о зарплатах сотрудников в файл, только если они удовлетворяют всем требованиям;
- Вывести среднюю зарплату сотрудников (сумма всех зарплат делится на количество записей);
- Вывести данные сотрудника с самой высокой зарплатой;
- Вывести данные сотрудника с наименьшей зарплатой;
- Вывести среднюю зарплату по каждому департаменту (сумма всех зарплат департамента делится на количество записанных зарплат по данному департаменту в файл);
- Выход.

Каждая строка, записанная в файл будет содержать информацию о сотруднике и его зарплате: фамилия, имя, департамент, зарплата.

Проверяется чтобы введенная фамилия и имя состояли из букв. Возможен ввод сложных имен и фамилий, разделенных через тире.

Проверяется чтобы название департамента состояло из букв, а если название состоит из нескольких слов – чтобы они были разделены одним пробелом.

Зарплата это вещественное число из интервала 1000.00 – 77000.00.

### ***Ход работы:***

При работе над данным заданием, я ориентировался по заданию и по-порядку выполнял каждый из пунктов. При этом, я неоднократно тестировал его на наличие ошибок, что в итоге привело к ее корректной работе на всех этапах.

Сама программа выглядит так:

```
1 import re
2 usage
3 def check_name(name):
4     """Проверяет, состоит ли имя из букв, дефисов и апострофов,
5     а также не начинается и не заканчивается дефисом."""
6     pattern = r"^[a-zA-Z]+(?:[-\s][a-zA-Z]+)*$"
7     return re.match(pattern, name) is not None
8
9 usage
10 def main():
11     # Списки для хранения данных о сотрудниках
12     names = []
13     departments = []
14     salaries = []
15
16     # Словарь для хранения информации о департаментах
17     department_info = {}
18
19 while True:
20     # Отображение меню
21     print("\nГлавное меню:")
22     print("1. Ввести данные о сотруднике")
23     print("2. Вывести среднюю зарплату по департаментам")
24     print("3. Найти сотрудника с самой высокой зарплатой")
25     print("4. Найти сотрудника с наименьшей зарплатой")
26     print("0. Выход")
27
28     # Получение выбора пользователя
29     choice = input("Введите номер пункта меню: ")
30
31     # Обработка выбора пользователя
32     if choice == '1':
33         # Ввод данных о сотруднике
34         name = input("Введите фамилию и имя: ")
35         while not check_name(name):
36             print("Неверный формат имени или фамилии. Повторите ввод.")
37             name = input("Введите фамилию и имя: ")
38
39         department = input("Введите название департамента: ").lower()
40         salary = input("Введите зарплату: ")
```

```

40     try:
41         salary = float(salary)
42         if not 1000 <= salary <= 77000:
43             print("Зарплата должна быть в диапазоне от 1000.00 до 77000.00.")
44             continue
45     except ValueError:
46         print("Неверный формат зарплаты.")
47         continue
48
49     names.append(name)
50     departments.append(department)
51     salaries.append(salary)
52
53     if department in department_info:
54         department_info[department]['count'] += 1
55         department_info[department]['total_salary'] += salary
56     else:
57         department_info[department] = {'count': 1, 'total_salary': salary}
58
59
60
61
62     elif choice == '2':
63         # Вывод данных о сотрудниках
64         if not names:
65             print("Список сотрудников пуст.")
66         else:
67             total_salary = sum(salaries)
68             average_salary = total_salary / len(salaries)
69             print(f"\nСредняя зарплата сотрудников: {average_salary:.2f}")
70
71
72     elif choice == '3':
73         # Вывод информации о средней зарплате по департаментам
74         if not department_info:
75             print("Информация о департаментах отсутствует.")
76         else:
77             print("\nСредняя зарплата по департаментам:")
78             with open("average_salaries.txt", 'w') as file:
79                 for department, info in department_info.items():
80                     average_salary = info['total_salary'] / info['count']
81                     print(f"{department}: {average_salary:.2f}")
82                     file.write(f"{department}: {average_salary:.2f}\n")
83

```

```

84 elif choice == '4':
85     # Поиск сотрудника с самой высокой зарплатой
86     if not names:
87         print("Список сотрудников пуст.")
88     else:
89         highest_salary = max(salaries)
90         highest_paid_employee_index = salaries.index(highest_salary)
91         print(f"\nСотрудник с самой высокой зарплатой: {names[highest_paid_employee_index]}: "
92               f"{departments[highest_paid_employee_index]}, "
93               f"{highest_salary:.2f}")
94
95
96 elif choice == '5':
97     # Поиск сотрудника с наименьшей зарплатой
98     if not names:
99         print("Список сотрудников пуст.")
100    else:
101        lowest_salary = min(salaries)
102        lowest_paid_employee_index = salaries.index(lowest_salary)
103        print(
104            f"\nСотрудник с наименьшей зарплатой: {names[lowest_paid_employee_index]}: "
105            f"{departments[lowest_paid_employee_index]}, "
106            f"{lowest_salary:.2f}")
107
108 elif choice == '0':
109     print("Вы вышли из программы.")
110     break
111
112 else:
113     # Неверный выбор
114     print("Неверный номер пункта меню. Повторите попытку.")
115
116 if __name__ == "__main__":
117     main()

```

Весь код программы был написан в одном файле и выполняет все функции, которые были написаны в задании. Коротко пройдуся по каждому из пунктов:

- **import re** - это инструкция в Python, которая загружает стандартную библиотеку re, которая предоставляет функции для работы с регулярными выражениями (Regular Expressions).

Регулярные выражения - это мощный инструмент для работы с текстом: они позволяют искать определенные шаблоны символов в строках, заменять их на другие строки, разбивать строки на подстроки и многое другое.

Далее я объявляю функцию `check_name`, которая получает в качестве аргумента переменную `name`. После чего в функции регулярный шаблон проверяет соответствует ли строка формату, который задан.

- **`return re.match (pattern, name) is not None`** - проверяет то, соответствует ли переданное имя (переменную `name`) заданному шаблону (переменная `pattern`)

- **`re.match (pattern, name)`** - функция из модуля `re`, которая пытается найти совпадение шаблона `pattern` в начале строки `name`.

- **`is not None`** - проверяет, что результат поиска совпадения не равен `None`. Если совпадение найдено, условие возвращает `True`, что означает, что имя соответствует заданному шаблону, в противном случае возвращается `False`.

Далее объявляю основную функцию `main`, в которой будут происходить все действия программы. В самом начале, после объявления функции, я создал списки для хранения данных сотрудников, для каждого пункта отдельно, т.е. отдельно имена, департаменты, зарплаты. Также я создал словарь для хранения информации о департаментах, там будут храниться данные о работниках, отделах и зарплатах.

Далее начинается цикл ``while True``. В самом конце программы, цикл будет прерван оператором `break`, иначе цикл будет бесконечным. После начала цикла при помощи функции `print()` вывожу на экран меню, так, как оно будет выглядеть в консоли пользователя, чтобы он мог пользоваться программой. Меню содержит в себе 5 пунктов: Ввести данные о сотруднике, Вывести среднюю зарплату по департаментам, Найти сотрудника с самой

большой зарплатой, Найти сотрудника с самой наименьшей зарплатой.

После я объявил переменную `choice`, которая будет использоваться для выбора пункта меню. Пользователь с клавиатуры должен ввести тот пункт, который ему нужен.

Чтобы обработать выбранный пользователем пункт меню, я создал большое условие по каждому из номеров.

Коротко пройду по каждому из условий:

- В условии `if choice == '1'` происходит следующее:

1. Пользователю предлагается ввести данные о сотруднике:

- ``name = input("Введите фамилию и имя: ")``: Пользователю предлагается ввести фамилию и имя сотрудника.

- ``department = input("Введите название департамента: ").lower()``: Пользователю предлагается ввести название департамента. ``.lower()`` используется для приведения введенного названия к нижнему регистру, чтобы обеспечить однородность данных.

- ``salary = input("Введите зарплату: ")``: Пользователю предлагается ввести зарплату сотрудника.

2. Данные проверяются на корректность:

- ``while not check_name(name):``: Имя и фамилия сотрудника проверяются с помощью функции ``check_name``, чтобы удостовериться, что они соответствуют заданному формату. Если формат неправильный, пользователю выводится сообщение о неверном формате, и ему предлагается ввести данные заново.

- ``try`...`except ValueError``: Зарплата преобразуется в число с помощью ``float(salary)``, и проверяется на соответствие диапазону от 1000 до 77000. Если ввод некорректный (например, не является числом или выходит за пределы допустимого диапазона), пользователю выводятся соответствующие сообщения об ошибке, и выполнение цикла продолжается.

3. Если данные корректны, они добавляются в соответствующие списки:

- `names.append(name)`: Имя и фамилия сотрудника добавляются в список `names`. При этом, метод `append()` всегда переносит новую добавленную информацию в конец списка. Это относится и ко всем последующим спискам, которые содержат данный метод.
- `departments.append(department)`: Название департамента добавляется в список `departments`.
- `salaries.append(salary)`: Зарплата сотрудника добавляется в список `salaries`.

4. Обновляется информация о департаментах:

- Если департамент уже существует в словаре `department_info`, то количество сотрудников в этом отделе увеличивается на 1, и общая сумма зарплат для этого отдела увеличивается на введенную зарплату.
- Если такого департамента нет в словаре `department_info`, создается новая запись с количеством сотрудников равным 1 и суммарной зарплатой равной введенной зарплате.

- В условии `elif choice == '2'` происходит следующее:

1. Проверка наличия данных о сотрудниках:

- Условие `if not names:` проверяет, пуст ли список `names`, который хранит имена сотрудников.
- Если список `names` пустой (то есть нет ни одного сотрудника), программа выводит сообщение "Список сотрудников пуст."

2. Вычисление средней зарплаты:



- Если список ``names`` не пустой, программа переходит к блоку кода под ``else``.
- С помощью функции ``sum(salaries)`` вычисляется общая сумма всех зарплат, хранящихся в списке ``salaries``.
  - Затем средняя зарплата вычисляется как отношение общей суммы зарплат к количеству сотрудников в списке ``salaries``:  
``average_salary = total_salary / len(salaries)``.

### 3. Вывод средней зарплаты:

- Полученное значение средней зарплаты выводится на экран с помощью ``print()``.
- ``{average_salary:.2f}`` - это форматированное представление средней зарплаты, где ``.2f`` обозначает, что число должно быть отформатировано до двух десятичных знаков.

Таким образом, если в списке есть хотя бы один сотрудник, программа вычисляет и выводит среднюю зарплату всех сотрудников. Если список пуст, выводится сообщение о том, что список сотрудников пуст.

**- В условии `elif choice == '3'` происходит следующее:**

#### 1. Проверка наличия информации о департаментах:

- Проверка выполняется с помощью условного оператора ``if not department_info:``.
  - Если словарь ``department_info`` пустой (то есть не содержит информации о департаментах), программа переходит к блоку кода, который выводит сообщение о том, что информация о департаментах отсутствует.

#### 2. Вывод информации о средней зарплате по департаментам:

- Если в словаре ``department_info`` содержится информация о департаментах, программа переходит к блоку кода под ``else``.

- Создается файл "average\_salaries.txt" для записи средних зарплат по департаментам с помощью ``with open("average_salaries.txt", 'w') as file:``.
- Для каждого департамента и связанной с ним информации в словаре ``department_info``:
- Средняя зарплата по текущему департаменту вычисляется как отношение общей суммы зарплат к количеству сотрудников в этом отделе: ``average_salary = info['total_salary'] / info['count']``.
- Средняя зарплата выводится на экран с использованием ``print()``, который формирует строку вида "Название департамента: средняя зарплата", где ``department`` - название департамента, а ``average_salary`` - средняя зарплата. Вывод происходит так: ``print(f'{department}: {average_salary:.2f}')`.
- Та же информация записывается в файл "average\_salaries.txt" с использованием ``write()``, формируя строки такого же формата: ``file.write(f'{department}: {average_salary:.2f}\n')``.

**- В условии `elif choice == '4'` происходит следующее:**

1. Проверка наличия данных о сотрудниках:

- Условие ``if not names:`` проверяет, пуст ли список ``names``, который хранит имена сотрудников.
- Если список ``names`` пустой (то есть нет ни одного сотрудника), программа выводит сообщение "Список сотрудников пуст."

2. Поиск сотрудника с самой высокой зарплатой:

- Если список ``names`` не пустой, программа переходит к блоку кода под ``else``.
- С помощью функции ``max(salaries)`` находится максимальная зарплата из списка ``salaries``, который содержит зарплаты всех сотрудников.
- С помощью метода ``index(highest_salary)`` находится индекс этой максимальной зарплаты в списке ``salaries``. Этот индекс

будет использоваться для нахождения соответствующих имени сотрудника и его департамента.

- Имя сотрудника, его департамент и самая высокая зарплата выводятся на экран с помощью функции ``print()'`.
- ``{names[highest_paid_employee_index]}'` - это имя сотрудника с самой высокой зарплатой. Оно получается из списка ``names'` по индексу ``highest_paid_employee_index'`.
- ``{departments[highest_paid_employee_index]}'` - это департамент сотрудника с самой высокой зарплатой. Он получается из списка ``departments'` по тому же индексу ``highest_paid_employee_index'`.
- ``{highest_salary:.2f}'` - это сама высокая зарплата, округленная до двух десятичных знаков. Она уже найдена и сохранена в переменной ``highest_salary'`.

**- В условии `elif choice == '5'` происходит следующее:**

1. Проверка наличия данных о сотрудниках:

- Условие ``if not names:'` проверяет, пуст ли список ``names'`, который хранит имена сотрудников.
- Если список ``names'` пустой (то есть нет ни одного сотрудника), программа выводит сообщение "Список сотрудников пуст."

2. Поиск сотрудника с наименьшей зарплатой:

- Если список ``names'` не пустой, программа переходит к блоку кода под ``else'`.
- С помощью функции ``min(salaries)'` находится минимальная зарплата из списка ``salaries'`, который содержит зарплаты всех сотрудников.
- С помощью метода ``index(lowest_salary)'` находится индекс этой минимальной зарплаты в списке ``salaries'`. Этот индекс будет использоваться для нахождения соответствующих имени сотрудника и его департамента.

- Имя сотрудника, его департамент и наименьшая зарплата выводятся на экран с помощью функции `print()`.
- `{names[lowest_paid_employee_index]}` - это имя сотрудника с наименьшей зарплатой. Оно получается из списка `names` по индексу `lowest_paid_employee_index`.
- `{departments[lowest_paid_employee_index]}` - это департамент сотрудника с наименьшей зарплатой. Он получается из списка `departments` по тому же индексу `lowest_paid_employee_index`.
- `{lowest_salary:.2f}` - это сама наименьшая зарплата, округленная до двух десятичных знаков. Она уже найдена и сохранена в переменной `lowest_salary`.

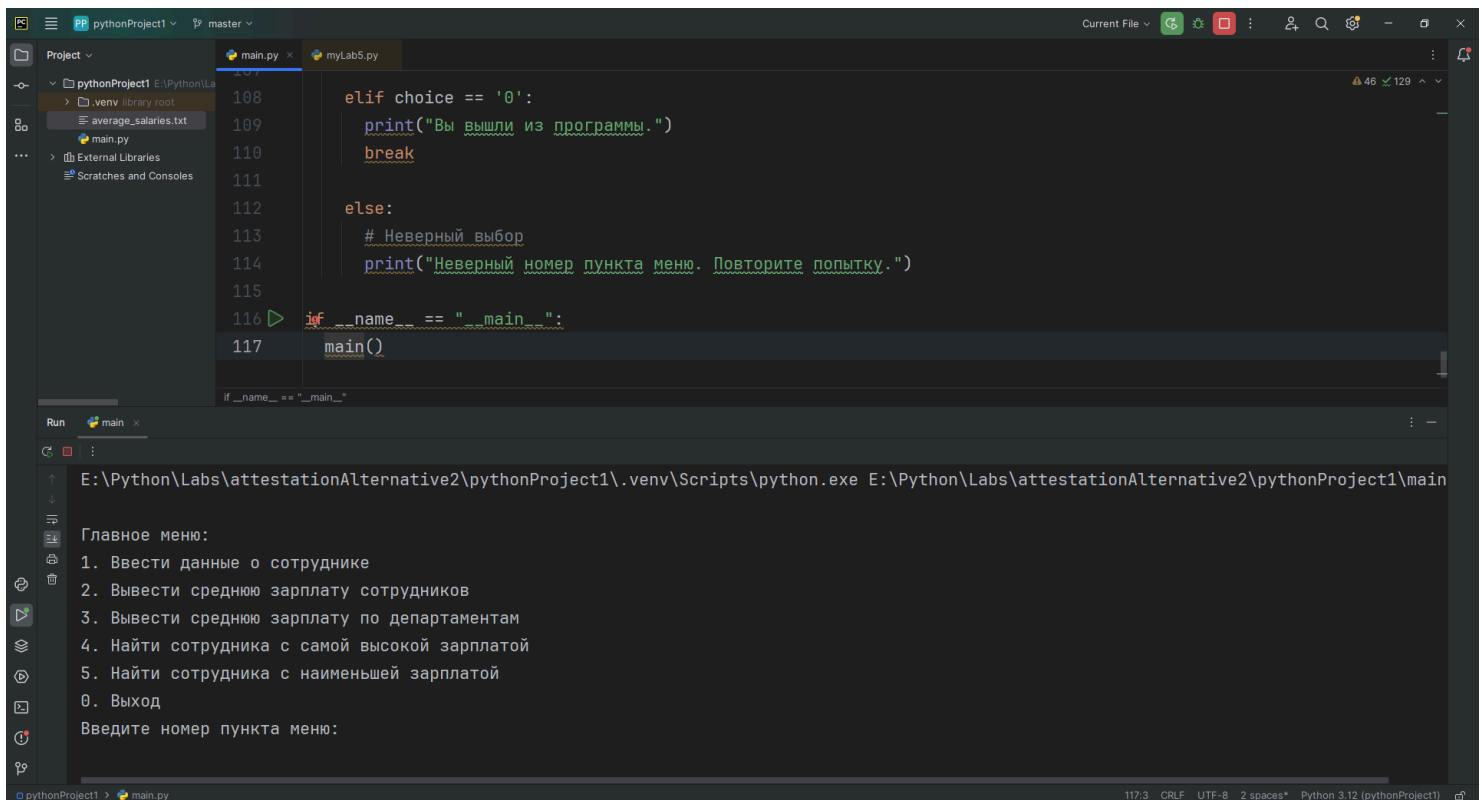
- В условии `elif choice == '0'` происходит следующее:

- При помощи функции `print()` на экран выводится сообщение “Вы вышли из программы” и при помощи оператора `break` бесконечный цикл прерывая работу самой программы.
- `else: print(“Неверный номер пункта меню. Повторите попытку.”)`. Это выводится в том случае, если был введен неверный номер пункта меню.

`if __name__ == "__main__": main()`: Это проверка, которая вызывает функцию `main()` только в том случае, если скрипт был запущен напрямую (а не импортирован как модуль в другой программе). Это позволяет избежать выполнения кода, если скрипт используется в качестве модуля.

Теперь я запущу программу, чтобы показать, что она работает корректно.

При запуске программы, в консоли выводится меню, при помощи которого пользователь сможет взаимодействовать с ней.



The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a console at the bottom. The code editor displays the following Python code:

```
108 elif choice == '0':
109     print("Вы вышли из программы.")
110     break
111
112 else:
113     # Неверный выбор
114     print("Неверный номер пункта меню. Повторите попытку.")
115
116 if __name__ == "__main__":
117     main()
```

The console output shows the program's execution:

```
Run main
E:\Python\Labs\attestationAlternative2\pythonProject1\.venv\Scripts\python.exe E:\Python\Labs\attestationAlternative2\pythonProject1\main

Главное меню:
1. Ввести данные о сотруднике
2. Вывести среднюю зарплату сотрудников
3. Вывести среднюю зарплату по департаментам
4. Найти сотрудника с самой высокой зарплатой
5. Найти сотрудника с наименьшей зарплатой
0. Выход
Введите номер пункта меню:
```

Выберу первый пункт меню и запишу туда несколько человек в разные департаменты.

```
Введите номер пункта меню: 1
Введите фамилию и имя: Ana-Maria Gasly
Введите название департамента: IT
Введите зарплату: 25000
Введите номер пункта меню: 1
Введите фамилию и имя: Mitkov Ivan
Введите название департамента: Marketing
Введите зарплату: 36500
```

```
Введите номер пункта меню: 1
Введите фамилию и имя: Yuki Tsunoda
Введите название департамента: Management
Введите зарплату: 14500
Введите номер пункта меню: 1
Введите фамилию и имя: Fernando Alonso
Введите название департамента: Financy
Введите зарплату: 45600
```

Теперь я специально буду вводить несуществующие имена, чтобы показать, что программа устойчива к такому роду недочетов.

```
Введите номер пункта меню: 1
Введите фамилию и имя: фф----
Неверный формат имени или фамилии. Повторите ввод.
Введите фамилию и имя: --aaaa
Неверный формат имени или фамилии. Повторите ввод.
Введите фамилию и имя: ---fff
Неверный формат имени или фамилии. Повторите ввод.
Введите фамилию и имя: а-а----
Неверный формат имени или фамилии. Повторите ввод.
Введите фамилию и имя: aa----
Неверный формат имени или фамилии. Повторите ввод.
```

Как можно заметить, введенный pattern в начале программы грамотно сработал при вводе несуществующих имен.

Далее, вывожу на экран второй пункт меню, а именно, среднюю зарплату сотрудников.

```
Введите номер пункта меню: 2

Средняя зарплата сотрудников: 30400.00
```

Как можно заметить, программа корректно выводит среднюю зарплату всех сотрудников.

Далее, я выведу среднюю зарплату по департаментам. Чтобы вычисления были более логичными, в каждый из департаментов я добавлю по 2-3 человека.

Введите номер пункта меню: 1	Введите номер пункта меню: 1
Введите фамилию и имя: <i>Arseni Alexander</i>	Введите фамилию и имя: <i>Pulev Vladislav</i>
Введите название департамента: <i>Financy</i>	Введите название департамента: <i>Financy</i>
Введите зарплату: 11500	Введите зарплату: 14250
Введите номер пункта меню: 1	Введите номер пункта меню: 1
Введите фамилию и имя: <i>Cuciuc Fiodor</i>	Введите фамилию и имя: <i>Chaz Gorot</i>
Введите название департамента: <i>IT</i>	Введите название департамента: <i>IT</i>
Введите зарплату: 41200	Введите зарплату: 20000

Введите фамилию и имя: *Shevchenko Andrei*  
Введите название департамента: *Management*  
Введите зарплату: 28700

Что получилось в итоге:

```
Средняя зарплата по департаментам:  
it: 28733.33  
management: 21600.00  
marketing: 36500.00  
financy: 23783.33
```

Департаменты отображаются с маленькой буквы, потому что я использовал метод `lower()`. Это помогает не совершить ошибку при вводе департамента и приведения введенного названия к нижнему регистру, чтобы обеспечить однородность данных. Данные отображаются корректно.

Далее, покажу на экран 4 пункт программы - поиск сотрудника с самой высокой зарплатой.

```
Введите номер пункта меню: 4  
  
Сотрудник с самой высокой зарплатой: Fernando Alonso: financy, 45600.00
```

Данный пункт также отображается корректно.

5 пункт программы - поиск сотрудника с наименьшей зарплатой.

```
Введите номер пункта меню: 5
```

```
Сотрудник с наименьшей зарплатой: Arseni Alexander: financy, 11500.00
```

Проверим последний пункт программы - выход из нее.

```
Введите номер пункта меню: 0
```

```
Вы вышли из программы.
```

```
Process finished with exit code 0
```

Как можно заметить, при выходе из программы, она прекращает свою работу.

Остается только проверить то, что будет, если я введу другую цифру, которой нет в меню. Для этого тоже есть условие, оно было обозначено в else.

```
Введите номер пункта меню: 8
```

```
Неверный номер пункта меню. Повторите попытку.
```

```
Главное меню:
```

1. Ввести данные о сотруднике
2. Вывести среднюю зарплату сотрудников
3. Вывести среднюю зарплату по департаментам
4. Найти сотрудника с самой высокой зарплатой
5. Найти сотрудника с наименьшей зарплатой
0. Выход

```
Введите номер пункта меню: |
```

Я ввел цифру 8, но программа вывела на экран то, что был введен неверный номер пункта меню. Все работает так, как нужно.



Еще стоит обратить внимание на одну деталь: при вводе каждого номера пункта меню и его выполнения, главное меню снова появлялось на экране, чтобы пользователю не пришлось перезапускать программу для начала работы с другим пунктом меню. т.е., когда пользователь закончит работу с необходимыми пунктами меню, он просто выберет пункт выхода из программы.

### ***Итог:***

Данная программа полностью выполняет весь функционал, который был обозначен в задании. Были использованы регулярные выражения, функции, условия, циклы, словари и списки. Для меня данная работа на языке программирования Python оказалась совсем несложной, потому что сам со себе язык легкий и логичный, а также совсем не сложный в применении, потому что все делается понятно.