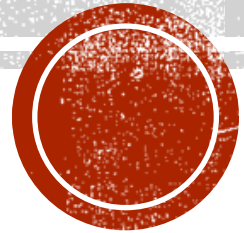
ЯЗЫКИ ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ

Лекция 1

КРАТКО О КУРСЕ



ЧТО ЖДЕТ ВПЕРЕДИ?

- Лекции каждую неделю
- 9 лабораторных работ
- 8 лабораторных пар в компьютерной аудитории
- 8 семинаров в семинарской аудитории, которые используются также для приема ЛР
- В конце экзамен

ЧТО ЖДЕТ ВПЕРЕДИ?

- Модуль 1: Python как язык анализа данных, git
- Модуль 2: C#, как прикладной ООП язык
- Модуль 3: Основы прикладной backend-разработки (БД, web, тестирование, логирование)
- Дополнительные лекции: если успеем, полезное из опыта

О ЛЕКТОРЕ

- Сычев Святослав Антонович
- Выпускник МГТУ
- Более 7 лет опыта прикладной разработки: от конструктора аналитических отчетов до системы отправляющих за месяц миллиарды **email**, **sms** и пушей
- Сейчас - **Engineering Manager** в **Mindbox**, отвечаю за результат команды из 30+ человек

tg @sych474

О БАЛЛАХ И ОЦЕНКАХ

- 7 баллов за посещение
- 63 балла за лабы:
  - 7 – если в срок
  - 5 – если не в срок
- 18-30 баллов экзамен
- Есть доп-задания за доп-баллы – спрашивайте у лаборанта

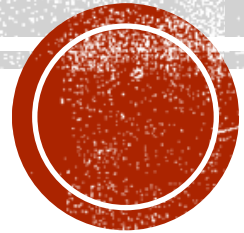
ПРАВИЛА ЛЕКЦИЙ

- Опоздали – не мешайте, заходите тихо
- В начале лекции отмечаем посещения
- С едой – в буфет или в коридор, воду можно
- Вопросы задаем по ходу лекции поднимая руку
- Главное – понять суть, а не переписать текст слайда/слова лектора

ОБЩИЕ ТРЕБОВАНИЯ К ЛР

- Следовать принципам KISS, DRY, YAGNI и т.п.
- Код приложения, если это не jupyter notebook должен быть разделен на “слои”/компоненты, как минимум:
  - компонент логики - отвечает за логику приложения, содержит доменные модели, классы, позволяющие с ними взаимодействовать и реализующие основную задачу, поставленную перед приложением. Должен быть независим от конкретной реализации других компонентов! Не должен содержать логику ввода-вывода.
  - компоненты ввода/вывода - отвечают за ввод данных пользователем, чтение и запись данных в файл и т. п. не должен содержать логики обработки данных, только чтение, парсинг, форматированный вывод.
- Приложение должно корректно обрабатывать (выводить понятную пользователю ошибку , а не падать) ошибочный ввод пользователя, ситуации отсутствия необходимого файла, некорректных данных и т.п.
- Код должен соответствовать code-style соответствующего языка

PYTHON



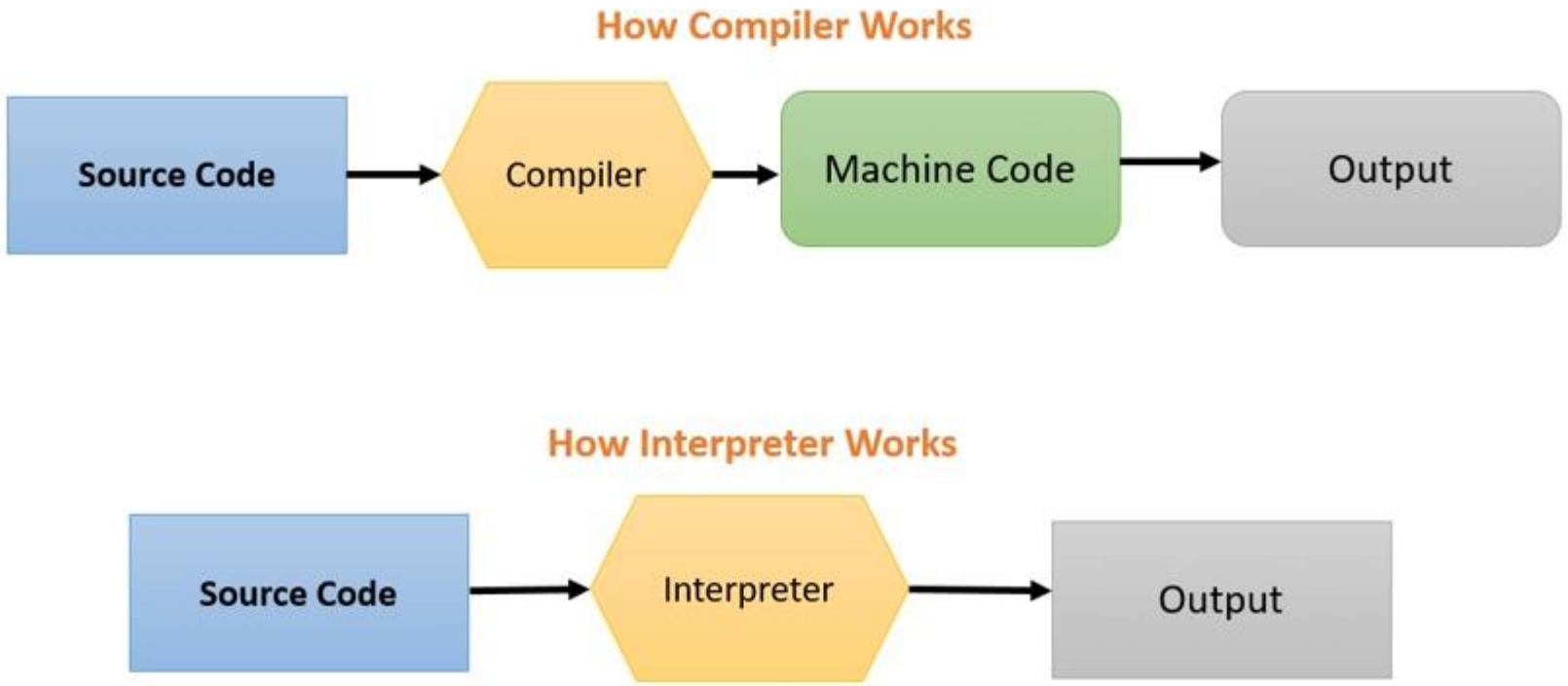
ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА

- **Python:**
  - Интерпретируемый
  - Мультипарадигменный
  - С динамической, строгой типизацией
  - С автоматическим управлением памятью
  - Кроссплатформенный

ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА

- Области применения:
  - Скрипты, небольшие программы и прототипы
  - **Web (django, flask)**
  - **Computer since** (аналитика данных, машинное обучение, нейросети)
  - Лингвистика
  - **etc.**

ИНТЕРПРЕТИРУЕМЫЙ ЯЗЫК



ДИНАМИЧЕСКАЯ ТИПИЗАЦИЯ

- Статическая типизация – типы данных проверяются на этапе компиляции
- Динамическая типизация –не требует указывать тип, и не выводит его. Типы переменных неизвестны до того момента, когда у них есть конкретные значения при запуске.
- Можно считать что *Типом обладают значения, а не параметры**

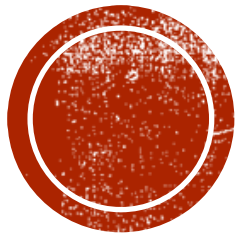
```
def f(x, y):
    return x + y
```

АВТОМАТИЧЕСКОЕ УПРАВЛЕНИЕ ПАМЯТЬЮ

- Память очищается автоматически, для этого используются:
  - Счетчики ссылок в объектах (все есть объект, очистка запускается когда ссылок 0)
  - Сборщик мусора (для очистки кольцевых ссылок)
- Память представляется в виде иерархической структуры

  - https://habr.com/ru/company/vk/blog/336156/
  - https://proglib.io/p/pomnit-vse-kak-rabotaet-pamyat-v-python-2021-03-14

- *Концепция: «Ресурсы сейчас дешевле, чем время специалиста»*



СИНТАКСИС И КОНСТРУКЦИИ ЯЗЫКА

СИНТАКСИС

- Конец строки является концом инструкции (точка с запятой не требуется).
- Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков.
- Инструкции могут быть разделены ; на одной строке.
- Инструкция может быть размещена на нескольких строках, если заключена в () {} или []

```
Основная инструкция:
    Вложенный блок инструкций
```

КЛЮЧЕВЫЕ СЛОВА

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

УСЛОВНЫЙ ОПЕРАТОР

- Инструкция ветвления: **if -> elif -> else**

- x < y - строго x меньше y,
- x <= y - x меньше или равно y,
- x > y - строго x больше y,
- x >= y - x больше или равно y,
- x == y - x равно y,
- x != y - x не равно y.

```
if test1:
    state1
elif test2:
    state2
else:
    state3
```

ТЕРНАРНЫЙ ОПЕРАТОР

```
if X:
    A = Y
else:
    A = Z
```

```
A = Y if X else Z
```

```python
a = "str"
match a:
    case "шаблон_1":
        print("действие_1")
    case "шаблон_2":
        print("действие_2")

    #...............

    case "шаблон_N":
        print("действие_N")
    case _:
        print("действие_по_умолчанию")

def print_hello(language):
    match language:
        case "russian":
            print("Привет")
        case "american english" | "british english" | "english":
            print("Hello")
        case _:
            print("Undefined")
```

PATTERN MATCHING

Аналог **switch-case**, появился в версии 3.10, подробный разбор оператора можно почитать по ссылке.

ЦИКЛ WHILE

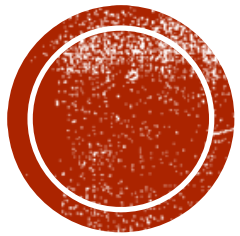
```
number = 1

while number < 5:
    print(f"number = {number}")
    number += 1
print("Работа программы завершена")
```

Цикл for

- Аналог **foreach** из **C#**
- ***for*** *переменная* ***in*** *набор_значений:*
  - *Итерация*
- *Break* – выход из цикла
- *Continue* – переход к следующей итерации
- *Else* – блок который будет выполнен после выхода из цикла

```
message = "Hello"
for c in message:
    print(c)
else:
    print(f"Последний символ: {c}. Цикл завершен");
```



«ПРОСТЫЕ» ТИПЫ ДАННЫХ

ЛОГИЧЕСКИЕ ЗНАЧЕНИЯ

- Любое число != 0 или непустой объект = **true**
- 0, пустые объекты, **None** = **false**
- Логическое и – **A and B**
- Логическое или – **A or B**
- Логическое отрицание – **not A** (подробности)

ЧИСЛА

- **Int**
  - Целые числа
  - Поддерживают длинную арифметику (потребует больше памяти)

- **Float**
  - Неточные (в силу специфики хранения)
  - Не поддерживают длинную арифметику

- **Complex**
  - Комплексные числа
  - Для работы с ними использовать **cmath**

ОПЕРАЦИИ НАД ЧИСЛАМИ

x + y	Сложение
x - y	Вычитание
x * y	Умножение
x / y	Деление
x // y	Получение целой части от деления
x % y	Остаток от деления
-x	Смена знака числа
abs(x)	Модуль числа
divmod(x, y)	Пара (x // y, x % y)
x ** y	Возведение в степень
pow(x, y[, z])	x^y по модулю (если модуль задан)

БИТОВЫЕ ОПЕРАЦИИ

x | y	Побитовое *или*
x ^ y	Побитовое *исключающее или*
x & y	Побитовое *и*
x << n	Битовый сдвиг влево
x >> y	Битовый сдвиг вправо
~x	Инверсия битов

СТРОКИ

- Строки в Python - упорядоченные **неизменяемые** последовательности символов, используемые для хранения и представления текстовой информации
- Строковый литерал может записываться как в двойных, так и в одинарных кавычках
- Если перед открывающей кавычкой стоит символ ‘r’ – экранирование будет отключено (*r'C:\new_dir'*)
- Для записи многострочного литерала используются тройные кавычки

```
>>> c = '''это очень большая
... строка, многострочный
... блок текста'''
>>> c
'это очень большая\nстрока, многострочный\nблок текста'
```

СТРОКИ

Экранированная последовательность	Назначение
\n	Перевод строки
\a	Звонок
\b	Забой
\f	Перевод страницы
\r	Возврат каретки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\N{id}	Идентификатор ID базы данных Юникода
\uhhhh	16-битовый символ Юникода в 16-ричном представлении
\Uhhhh…	32-битовый символ Юникода в 32-ричном представлении
\xhh	16-ричное значение символа
\ooo	8-ричное значение символа
\0	Символ Null (не является признаком конца строки)

СТРОКИ

- Конкатенация: `s1 + s2`
- Дублирование: `s * 3`
- Длина строки (и не только): `len(s)`
- Форматный вывод при помощи `format` (примеры)

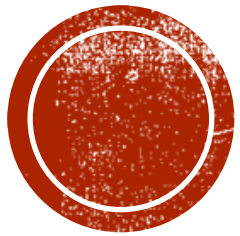
СТРОКИ

```
# аргументы по умолчанию
print("Hello {}, your balance is {}.".format("Adam", 230.2346))

# позиционные аргументы
print("Hello {0}, your balance is {1}.".format("Adam", 230.2346))

# аргументы ключевые слова
print("Hello {name}, your balance is {blc}.".format(name="Adam", blc=230.2346))

# смешанные аргументы
print("Hello {0}, your balance is {blc}.".format("Adam", blc=230.2346))
```



«СЛОЖНЫЕ» ТИПЫ ДАННЫХ

СПИСКИ

- Список: упорядоченная изменяемая коллекция объектов произвольных типов переменной длины
- Способы создать список:
  - `a = list(1, 2, 3)` – при помощи встроенной функции
  - `a = ["1", 42, []]` – при помощи литерала
  - `a = [i for i in range(5)]` – при помощи генератора `// [0, 1, 2, 3, 4]`

- *Массивы – в модуле* `array`

СПИСКИ (ОСНОВНЫЕ МЕТОДЫ)

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-ый элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

ИНДЕКСЫ

- Индексация с 0, при помощи []
- При попытке доступа к несуществующему индексу возникает исключение IndexError
- Поддерживаются отрицательные индексы: `a[0] = a[len(a)]`

СРЕЗЫ

- item[START:STOP:STEP]
  - START — индекс первого элемента в выборке, по умолчанию 0
  - STOP — индекс элемента списка, перед которым срез должен закончиться (т.е. сам элемент с индексом STOP не будет входить в выборку), по умолчанию = длине объекта
  - STEP — шаг прироста выбираемых индексов по умолчанию 1.
- Параметры могут быть опущены.
- Срез – тоже объект.

СРЕЗЫ – ПРИМЕРЫ

- item[START:STOP:STEP]
  - START — индекс первого элемента в выборке, по умолчанию 0
  - STOP — индекс элемента списка, перед которым срез должен закончиться (т.е. сам элемент с индексом STOP не будет входить в выборку), по умолчанию = длине объекта
  - STEP — шаг прироста выбираемых индексов по умолчанию 1.
- Параметры могут быть опущены.
- Срез – тоже объект.

```
a = [1, 3, 8, 7]
a[:] >>> [1, 3, 8, 7]
a[1:] >>> [3, 8, 7]
a[:3] >>> [1, 3, 8]
a[::2] >>> [1, 8]
a[::-1] >>> [7, 8, 3, 1]
a[:-2] >>> [1, 3]
a[-2::-1] >>> [8, 3, 1]
a[1:4:-1] >>> []
```

СЛОВАРИ

- Словарь – неупорядоченная коллекция произвольных объектов с доступом по ключу.
- Создание:
  - d = {}
  - d = dict(short='dict', long='dictionary’) => {'short': 'dict', 'long': 'dictionary'}
  - d = dict.fromkeys(['a', 'b']) => {'a': None, 'b': None}
  - d = {a: a ** 2 for a in range(7)} => {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
- Доступ к значению:
  - d[‘short’]
  - если ключа нет – исключение **KeyError**
  - Может быть изменено при помощи присваивания

СЛОВАРИ

Метод	Что делает
dict.clear()	очищает словарь
dict.copy()	возвращает копию словаря
dict.fromkeys(seq[, value])	создает словарь с ключами из seq и значением value (по умолчанию None)
dict.get(key[, default])	возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None)
dict.items()	возвращает пары (ключ, значение)
dict.keys()	возвращает ключи в словаре
dict.pop(key[, default])	удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение)
dict.popitem()	удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены
dict.setdefault(key[, default])	возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None)
dict.update([other])	обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются.
dict.values()	возвращает значения в словаре

МНОЖЕСТВА

- **set** – коллекция, содержащая неповторяющиеся элементы в случайном порядке.
- Создание:
  - `a = set()`
  - `a = set('hello') >>> {'h', 'o', 'l', 'e'}`
  - `a = {'a', 'b', 'c', 'd'} >>> {'b', 'c', 'a', 'd'}`
  - `a = {i ** 2 for i in range(10)} >>> {0, 1, 4, 81, 64, 9, 16, 49, 25, 36}`
- **frozenset** – неизменяемый вариант **set**.

МНОЖЕСТВА

Метод	Что делает
x in s	Принадлежит ли **x** множеству **s**
set == other	Совпадение множеств
set <= other или set.issubset(other) set >= other или set.issuperset(other)	Подмножество
set.union set | other |…	Объединение
set & other & … set.intersection(other, …)	Пересечение
set - other - … set.difference(other, …)	Разность множеств
set ^ other set.symmetric_difference(other)	Симметрическая разность множеств
set.copy()	Создает копию множества

МНОЖЕСТВА

Метод	Что делает
set.add(elem)	Добавить элемент в множество
set.remove(elem)	Удалить элемент из множества. KeyError, если такого элемента не существует.
set.discard(elem)	Удаляет элемент, если он находится в множестве.
set.pop()	Удаляет элемент из множества (случайный)
set.clear()	Очистка множества

КОРТЕЖ

- Кортеж – неизменяемый список
  - Занимает меньше памяти.
  - Может быть использован как ключ словаря
- Создание:
  - `a = tuple()`
  - `a = ()`
  - `a = ('s', )`
- Доступны все операции списков, не изменяющие значений.
- `a, b = b, a`

ОПРЕДЕЛЕНИЕ ФУНКЦИЙ

- Функция начинается с **def**,
- Функция может и не заканчиваться инструкцией return, при этом функция вернет значение None.
- Переменное число параметров:
  - ***args** – кортеж входных параметров
  - ****kwargs** – словарь входных параметров

```
def add(x, y):
    return x + y
```

```
def func(*args):
    return args
```

```
def func(**kwargs):
    return kwargs
```

```
def func():
    pass
```

ОБРАБОТКА ИСКЛЮЧЕНИЙ И ЧТЕНИЕ ФАЙЛОВ

```
f = open('1.txt')
ints = []
try:
    for line in f:
        ints.append(int(line))
except ValueError:
    print('Это не число. Выходим.')
except Exception:
    print('Это что ещё такое?')
else:
    print('Всё хорошо.')
finally:
    f.close()
    print('Я закрыл файл.')
```

ЛАБОРАТОРНАЯ РАБОТА #1

- Прочитать **CSV** файл с числовыми данными (**russian demographic.csv** из 1 года)
- Вычислить статистические метрики по набору данных:
  - Максимум, минимум, среднее
  - Медиана
  - Таблица перцентилей от 0 до 100 с шагом 5
- Интерфейс – консольный
- Программа должна корректно обрабатывать все виды ошибок и уведомлять об этом пользователя (некорректный входной файл, ошибки данных и т.п.).
- *N-й перцентиль - это такое число X, что N% элементов массива меньше или равны этому числу X.*

ЧТО ИЗУЧАТЬ?

- самоучители по **python**
  - https://metanit.com/python/tutorial/
  - https://pythonworld.ru/samouchitel-python

- https://www.python.org/doc/ - официальная документация по языку
- Марк Лутц «Изучаем Python»
- Эрик Мэтиз «Изучаем Python. Программирование игр, визуализация данных, веб-приложения»
- Эл Свейгарт «Автоматизация рутинных задач с помощью Python. Практическое руководство для начинающих»
- https://habr.com/ru/company/sberbank/blog/679852/ - статья с разбором книг про **python**


Мой код не работает :-(
Начни отсюда
Ты получаешь ошибку, когда запускаешь код?
нет
В коде используются циклы или переменные if?
нет
циклы
if
Какой тип ошибки?
да
AtributeError
Ты используешь метод на объекте неверного типа
SyntaxError
Ты забыл взять строку в кавычки
Ты забыл добавить двоеточие в конце строки def/if/for
У тебя не сходится количество открывающих и закрывающих скобок в выражении
TypeError
Ты используешь оператор на объектах неверного типа
Объект, который должен обладать значением, является None
Ты используешь нецелые числа в слайсе списка
Ты вызвал метод/функцию с неверным количеством или типом аргументов
Indentation Error
Ты одновременно используешь и отступы, и пробелы
Ты не выровнял строки в блокер
NameError
Ты допустил ошибку в названии переменной, метода или функции
Ты забыл импортировать модуль
Ты не определил переменную
Твой код вызывает функцию до того, как она была определена
Ты пытаешься напечатать отдельное слово и забыл про кавычки
IOError
Ты пытаешься открыть файл, который не существует
KeyError
Ты пытаешься осуществить поиск по ключу, которого нет в словаре
Переменная, которая должна содержать значение, не содержит его
Ты хранишь возвращаемое значение функции, которое изменяет саму переменную
Дробное число становится нулём в Python 2
Ты делишь целые числа вместо использования чисел с плавающей точкой.
Преобразуй свои числа в числа с плавающей точкой или используй "from __future__ import division".
Я пытаюсь напечатать значение, но получаю только странную строку
Ты печатаешь сам объект (например, FileObject), а нужен результат метода объекта
Регулярное выражение не подходит, когда я этого ожидаю
Ты забыл использовать неформатированные строки или экранировать обратный слэш
Я пытаюсь прочитать файл, но не могу ничего считать
Ты уже читал содержимое файла в коде ранее, и курсор сейчас находится в конце файла
Числа, которые должны быть равны, не равны
Ты сравниваешь число со строковым представлением числа (например, if 3 == '3')
Составное условие не выдаёт ожидаемый результат
Порядок приоритетов в условии двусмысленный — добавь скобки
Список, который должен иметь значение на каждый проход цикла, имеет только одно значение
Ты определил список внутри цикла, вынеси его наружу
Цикл, использующий range-функцию, упускает последнее значение
Range-функция не включает в себя конец указанного интервала, увеличь его на 1
Я пытаюсь перебрать элементы коллекции строк, но вместо этого перебираю отдельные символы
Ты ошибочно перебираешь внутри строки вместо коллекции строк
Я пытаюсь записать в файл несколько строк, но на выходе получаю одну строку
Ты открываешь файл внутри цикла, вынеси это действие наружу
также проверь...