

*School of
Computer
Science*

КОРТЕЖИ. ПОТОКИ. ФАЙЛЫ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

- Выберите из списка ниже случаи, подходящие для использования цикла **while** и цикла **for** в Python:
1. Количество итераций зависит от ввода пользователя
 2. Конечное, заранее известное количество итераций
 3. Бесконечный цикл с выходом по условию в теле цикла с помощью **break**
 4. Перебор значений из списка

?



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

Рассмотрите этот код:

```
for <элемент> in <список>:
    <блок1>                # может ли <блок1> НЕ исполниться?
    if <условие>:          # и в каком случае?
        continue
    else:
        break
    <блок2>                # а <блок2> когда-нибудь выполнится?
else:
    <блок3>                # когда выполнится <блок3>?
```

Ваши ответы?



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

– Какой ряд чисел образуется после выполнения алгоритма

```
for i in range(1,10+1):  
    print(i):
```

1) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

2) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

3) 1, 11, 23, 36, 40, 45, 61, 78, 96, 115

?



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

- Рассмотрите код, который печатает каждый с шагом 3 символы строки `str_3`:

```
>>> str_3 = 'abcABCabcABC'  
>>> for i in range(0, len(str_3), 3):  
    print(_____)
```

- Что должен напечатать этот код?
- Выберите выражения, которые можно подставить в функцию `print()`:

1. `i`
2. `str_3[i : i+1]`
3. `str_3[i + i]`
4. `str_3[i]`



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

– К чему приведет обращение к непустому списку по индексу «-1»:

1. Вернётся последний элемент
2. Ошибка `IndexError`
3. Вернется первый элемент
4. Ошибка `KeyError`

?



ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

- Сколько элементов будет содержать список `students` после следующих операций:

```
1 students = ['John', 'Peter', 'Mary']  
2 students += ['Kate']  
3 students += 'Kate'  
4
```

- Объясните почему так происходит?



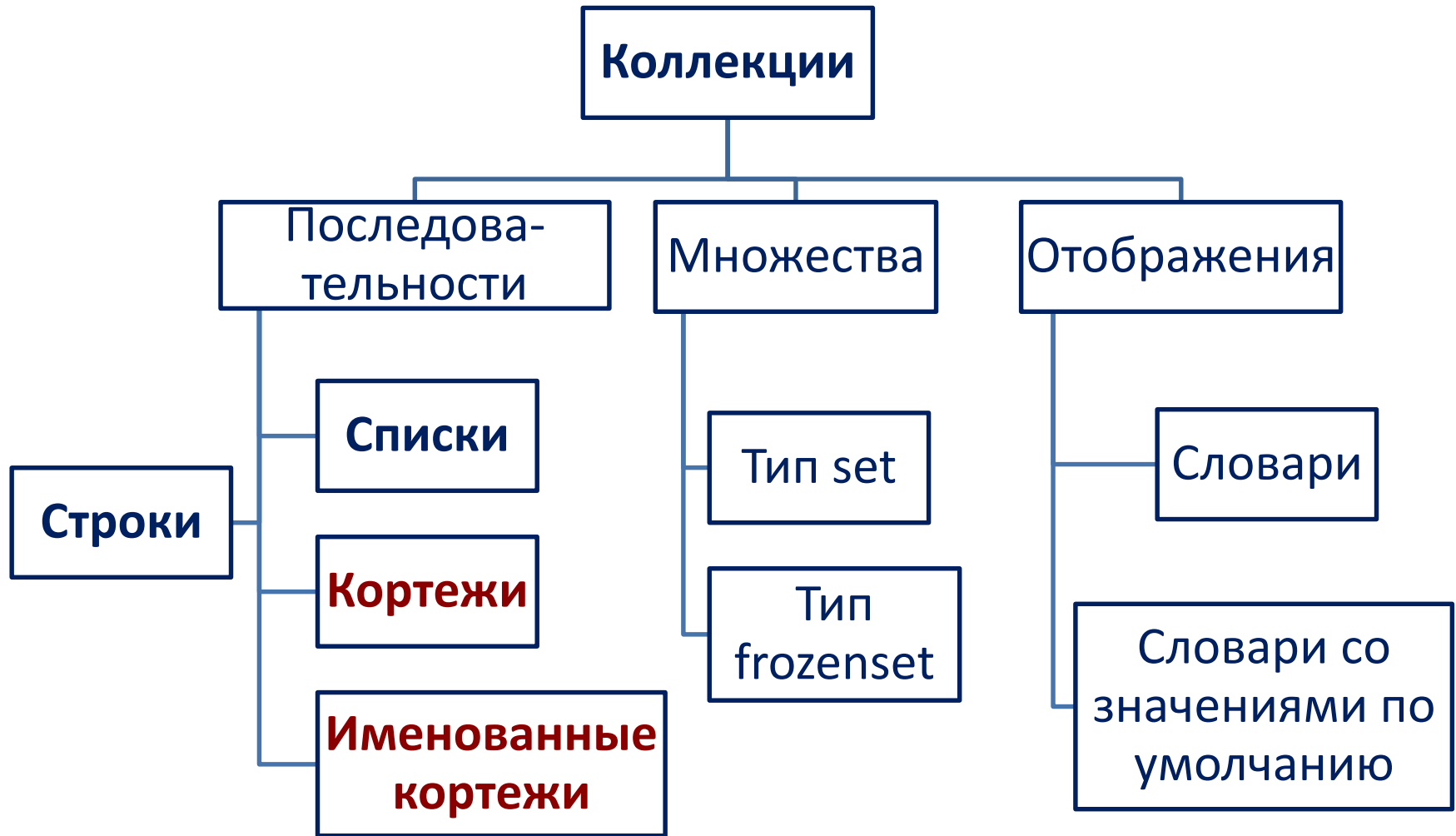
ВОПРОС ПО ПРОШЛОМУ ЗАНЯТИЮ

- Имеется программа, код которой указан ниже.
- Укажите, какие значения будут содержать списки в помеченных участках:

```
1  list_1 = [1, 2, 3]
2  list_2 = list_1
3  # значения списка list_2?
4
5  list_1[1] = 10
6  # значения списка list_2?
7
8  list_2[0] = 20
9  # значения списка list_1?
10
11 list_1 = [5, 6]
12 # значения списка list_2?
13
```




ТИПЫ КОЛЛЕКЦИЙ В PYTHON





КОРТЕЖИ (TUPLES)

- Кортеж – это последовательность, поддерживающая следующие операции:
 - Конкатенация и тиражирование: `+` и `*`
 - Проверки на вхождение `in` и `not in`
 - Функция определения размера `len(object)`
 - Индексация: `object[index]`
 - Извлечение срезов: `object[start:stop:step]`
 - Итерации, гарантирующие строгую последовательность элементов
- Соответствует строке данных в таблице
- Кортеж, в отличие от списков, это **неизменяемый** тип данных



ИНИЦИАЦИЯ КОРТЕЖЕЙ

- Объявление:

- `tuple()`
- `()`
- *(элементы через запятую)*

- Примеры:

- `tup1 = tuple()` # пустой кортеж
- `tup2 = ()` # тоже пустой кортеж
- `tup3 = (1,)` # кортеж с одним
#элементом
- `tup3 = (-17.5, 'text', 83)`
- `tup4 = (['Monday', -18], # кортеж
['Tuesday', +5], # с вложен-
'Unknown') # ными списками`



МЕТОДЫ КОРТЕЖЕЙ

Вызов	Описание
T.count(x)	Возвращает число вхождений элемента x в кортеж T
T.index(x [, start, end])	Возвращает индекс самого первого (слева) вхождения элемента x в кортеж T (или в срез start:end кортежа T) или возбуждает исключение ValueError



ПРИМЕРЫ РАБОТЫ С КОРТЕЖАМИ

```
>>> tup1 = (1, 2, 3, 4)      # кортеж из 4-ех элементов
>>> len(tup1)                # длина
4
>>> tup1 + (5, 6)            # конкатенация
(1, 2, 3, 4, 5, 6)
>>> tup1[1]                  # извлечение элемента
2
>>> tup1[2:4]                # извлечение среза
(3, 4)
>>> tup1.index(4)            # значение 4 находится в позиции 3
3
>>> tup1.count(4)            # значение 4 присутствует 1 раз
1
>>> tup1[0] = 0              # кортежи являются неизменяемыми
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    tup1[0] = 0                # кортежи являются неизменяемыми
TypeError: 'tuple' object does not support item assignment
```



ИМЕНОВАННЫЕ КОРТЕЖИ

- Иницииируются с помощью `collections.namedtuple()`
- Нужен предварительный `import collections`
- Позволяет адресоваться к элементам кортежа не только числовыми, но и текстовыми индексами, например:
 - `sale.quantity * sale.price`
 - `aircraft.seating.maximum`
- См. примеры в `tuple_indexing.py` и `tuple_named.py`



УСОВЕРШЕНСТВОВАННЫЕ ИМЕНОВАННЫЕ КОРТЕЖИ

- Тип `typing.NamedTuple` был добавлен с версии Python 3.6
- Иницируются синтаксисом объявления классов и с аннотациями типов.
См. [Введение в аннотации типов Python](#)
- Сигнатуры типов не проверяются без отдельного инструмента проверки типов, такого как `mypy`: <http://mypy-lang.org>
- См. примеры в `typing_NamedTuple.py`



ПОТОКОВЫЙ ВВОД-ВЫВОД

- Стандартные потоки ввода-вывода ОС в Python представлены в модуле `sys`:
 - `sys.stdin` – интерактивный ввод, включая вызов функции `input()`
 - `sys.stdout` – используется для печати (вывода) информации, включая вызов функции `print()`
 - `sys.stderr` – сообщения самого интерпретатора Python, включая сообщения об ошибках
- По умолчанию `sys.stdin` связан с клавиатурой, а `sys.stdout` и `sys.stderr` – с монитором



ПОТОКИ КАК ФАЙЛЫ

- С потоками можно работать как с файлами
- Метод `read()` объекта `sys.stdin` аналогичен использованию функции `input()`
- Особенность при чтении – нужно смоделировать конец ввода (EOF) с помощью комбинации клавиш Ctrl+D
- Метод `write()` объекта `sys.stdout` аналогичен функции `print()`
- См. примеры в `std_in_out_shell.txt`

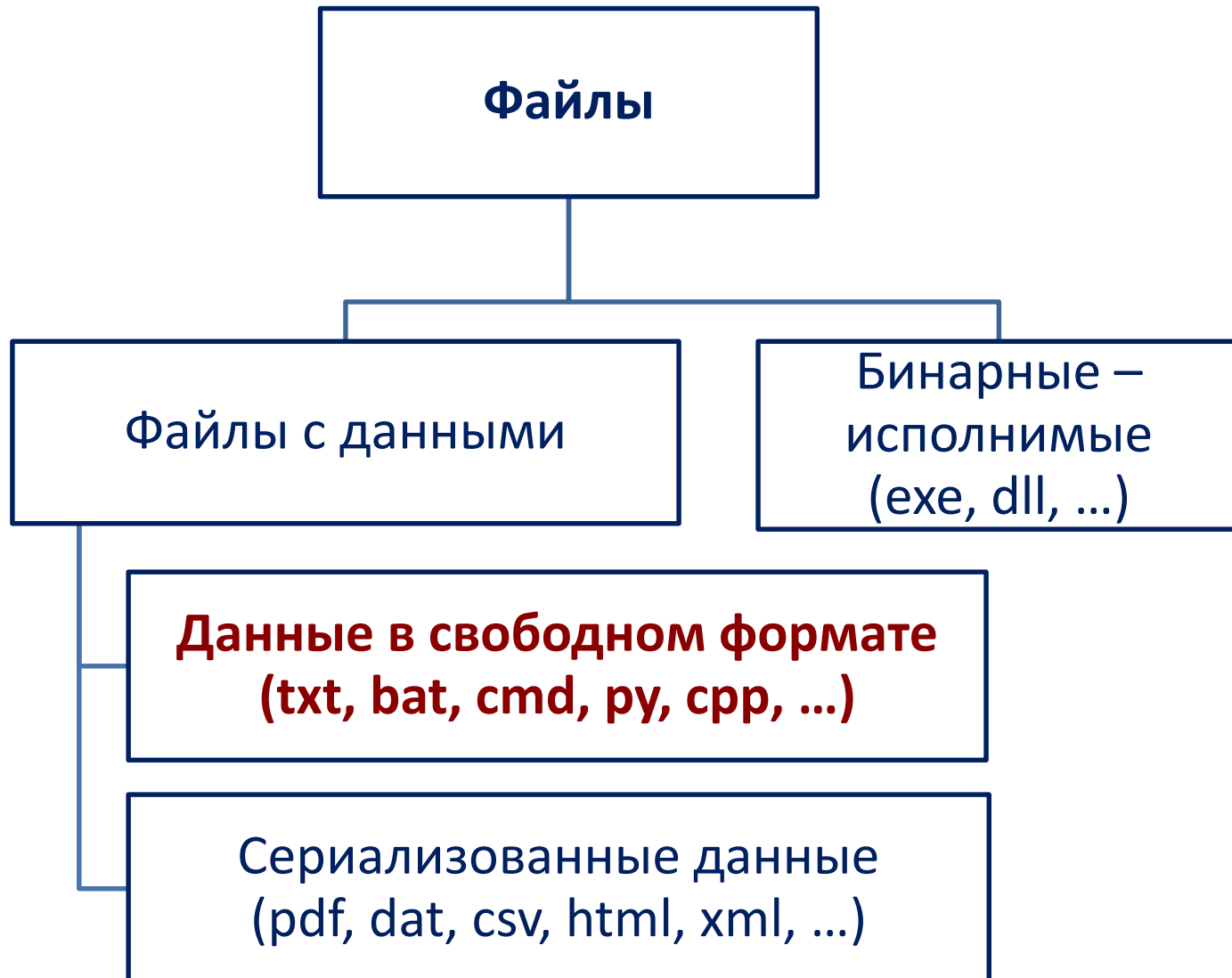


ПЕРЕНАПРАВЛЕНИЕ ПОТОКОВ ВВОДА-ВЫВОДА

- Направление потока ввода `sys.stdin` на чтение из файла:
 - `script.py < file_name`
- Направление потока вывода `sys.stdout` на запись в файл:
 - `script.py > file_name`
- Примеры:
 - `average_stream.py < Data\numbers.txt`
 - `average_stream.py < Data\numbers.txt > results.txt`



ФАЙЛОВЫЕ ФОРМАТЫ





РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ

- Открытие:
 - `file_object = open('имя файла', [режим])`
- Чтение или запись:
 - Читаем с помощью методов `file_object`:
`read()`, `readline()`, `readlines()` или в цикле `for`
 - Пишем с помощью `file_object.write()`
- Заккрытие:
 - `file_object.close()`



ОСНОВНЫЕ РЕЖИМЫ ТЕКСТОВОГО ФАЙЛА

Режим	Значение
r	Открыть для чтения существующий файл. Используется по умолчанию
w	Открыть для записи. Создает новый файл или перезаписывает существующий
a	Открыть для записи. Добавлять в конец файла, если файл существует
x	Открыть для эксклюзивной записи. Выдается исключение <code>FileExistsError</code> если такой файл уже существует



ДОПОЛНИТЕЛЬНЫЕ РЕЖИМЫ ТЕКСТОВОГО ФАЙЛА

Режим	Значение
r+	Открыть для чтения и для записи. Указатель на начало файла
w+	Открыть для записи и для чтения. Создает новый файл или перезаписывает существующий
a+	Открыть для добавления и чтения. Если файл существует, указатель устанавливается на конец файла и файл открывается в режиме добавления. Если файла не существует, то создаётся новый для чтения и для записи



НЕКОТОРЫЕ ВОЗМОЖНОСТИ ОБЪЕКТА FILE

Вызов	Описание
f.name	Имя открытого файла f
f.mode	Режим открытия файла f
f.closed	Возвращает False если файл f был закрыт, и True если нет
f.seek(index)	Установка указателя на заданную позицию в файле
f.readable()	Есть ли разрешение на чтение по файлу f (True или False)
f.writable()	Есть ли разрешение на запись по файлу f (True или False)



МЕТОДЫ ЧТЕНИЯ/ЗАПИСИ СТРОК ТЕКСТОВОГО ФАЙЛА

Вызов	Описание
f.readline()	Чтение строки, включая символ перевода строки
f.read()	Чтение всего файла
f.readlines()	Чтение строк, включающих символы перевода строки, в список
f.write()	Запись строки в файл. Символ перевода строки автоматически НЕ добавляется



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ПОСТРОЧНОЕ ЧТЕНИЕ ФАЙЛА

- Открываем файл `Data\poem_file.txt` для чтения в Python Shell
- Читаем последовательно каждую строку из файла с помощью метода `readline()`
- Читаем файл в цикле и распечатываем его содержимое
- Какой побочный эффект мы увидим?
- Как его можно избежать?



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ЧТЕНИЕ ВСЕГО ФАЙЛА ЗА ОДИН РАЗ

- Открываем файл `Data\poem_file.txt` для чтения в Python Shell
- Читаем ВСЕ содержимое файла в строку с помощью метода `read()`
- Распечатываем содержимое файла с помощью однократного `print()`
- В каких случаях применимо полное чтение файла?



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ИТЕРАЦИИ ПО ФАЙЛУ

- Открываем файл `Data\poem_file.txt` для чтения в Python Shell
- Текстовый файл – это аналог последовательности, в которой каждая строка является ее элементом
- В цикле `for` делаем итерации по файлу и распечатываем его построчно
- Строки читаются по мере надобности без риска переполнения памяти



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ЧТЕНИЕ ФАЙЛА В СПИСОК

- Открываем файл `Data\pоем_file.txt` для чтения в Python Shell
- Читаем ВСЕ содержимое файла в список с помощью метода `readlines()`
- Распечатываем содержимое файла в цикле `for` по списку
- Когда может быть востребовано чтение файла в список?



ПРАКТИЧЕСКОЕ ЗАНЯТИЕ.

ЧТЕНИЕ ИЗ ФАЙЛА, ЗАПИСЬ В ДРУГОЙ ФАЙЛ

```
from tkinter import filedialog as fd  
from_file_name = fd.askopenfilename()  
to_file_name = fd.asksaveasfilename()
```

- Открыть файл с именем `from_file_name` для чтения и прочитать все его содержимое
- Открыть файл с именем `to_file_name` для записи и записать туда “Копия\n” и содержимое `from_file_name`



- Имеется CSV файл с таким содержимым:
bank,release date,value,income
SBRF,31.12.2019,325000,5506
VTB24,30.06.2020,162500,3620
- Нужно сделать вычисления по строкам файла, исключая его 1-ую строку
- Какой подход лучше использовать:
 1. `readline()`
 2. итерация по файлу в цикле `for`
 3. `read()`
 4. `readlines()`

?



НОВОЕ ДОМАШНЕЕ ЗАДАНИЕ.

«УЖАСНАЯ ПОЭЗИЯ» V.3.0

- Формировать списки существительных, глаголов и **прилагательных** из файлов
- Запуск скрипта должен выглядеть так:
 - `awful_poetry.py <количество_строк>`
`<файл_с_существительными>`
`<файл_с_глаголами>`
`<файл_с_прилагательными>`
- Случайные прилагательные из файла вставлять перед существительным в 2/3 случаев, когда не используются наречия
- Смотрите все файлы с частями речи в **Data**, прилагательные в файле **adjectives.txt**
- **Оптимизируйте полученный код с применением собственных функций (*)**



ДОМАШНЕЕ ЗАДАНИЕ №2.

«ЛЕВ ТОЛСТОЙ И КОПИРАЙТЕР»

- Л.Н. Толстой заказывает копирайтеру на фрилансе переписать его роман «Война и мир»
- Средняя скорость письма копирайтера составляет 100 знаков в минуту, включая пробелы
- Плата за его работу – 150 рублей за символ (пробелы не оплачиваются)
- Помогите Льву Николаевичу и напишите программу, которая посчитает по файлу `Data/War & Peace.txt`:
 - *Общее количество знаков, включая пробелы, но не считая переводы строк*
 - *Сколько в романе символов, за которые нужно заплатить (не считаем пробелы внутри строк)*
 - *Сколько времени в 8-ми часовых рабочих днях потребуется копирайтеру, чтобы переписать роман*
 - *Сколько «Отец русской литературы» заплатит этому фрилансеру*

СПАСИБО ЗА ВНИМАНИЕ !
ВОПРОСЫ ?



*School of
Computer
Science*