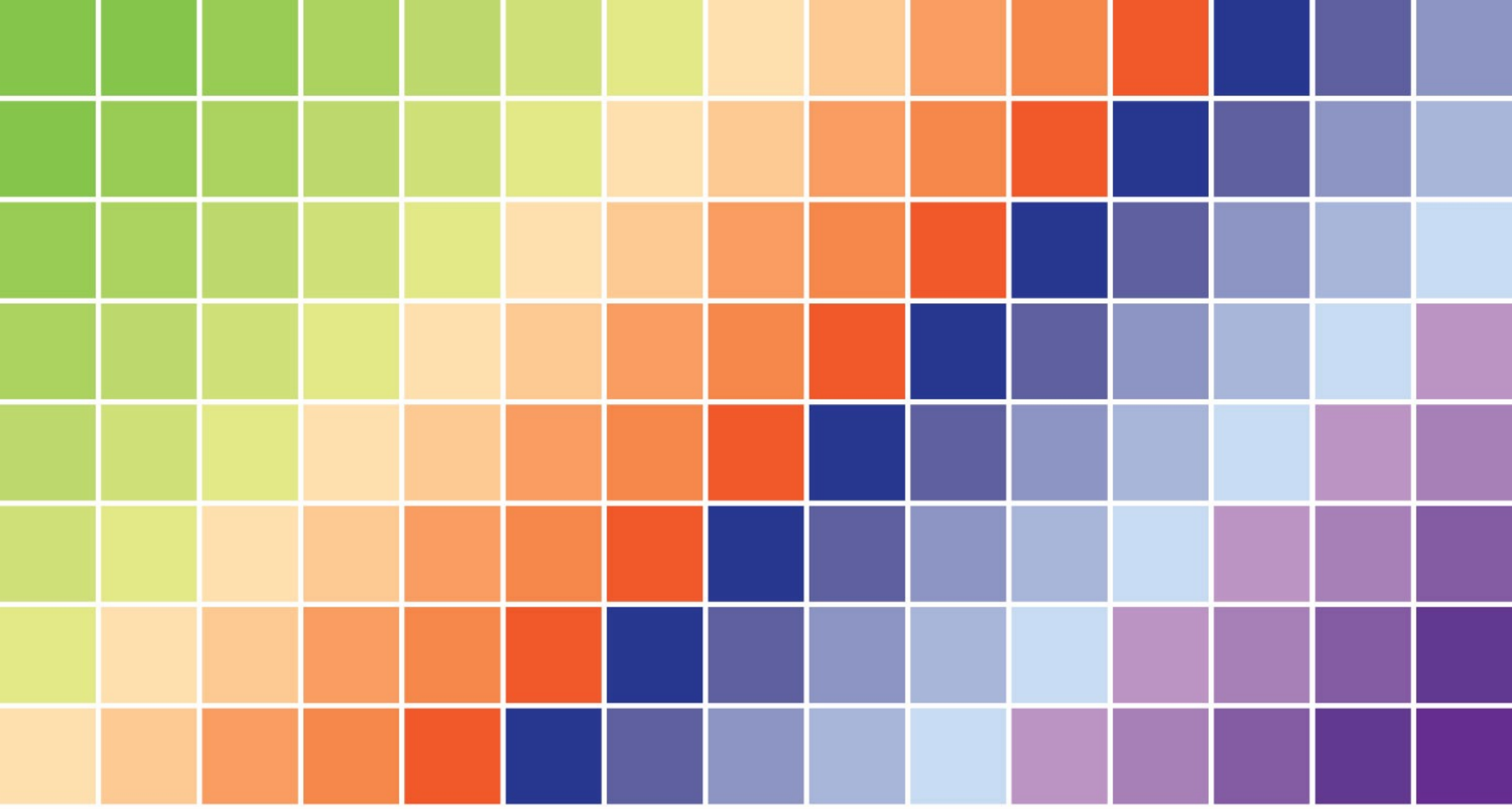


School of Computer Science

РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ. И СНОВА ПРО ИСКЛЮЧЕНИЯ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ.
АНОНИМНЫЕ ФУНКЦИИ

Найдите ошибки в определении анонимной функции:

```
>>> def lambda x, y:
        x *= 2
        return x - y
```

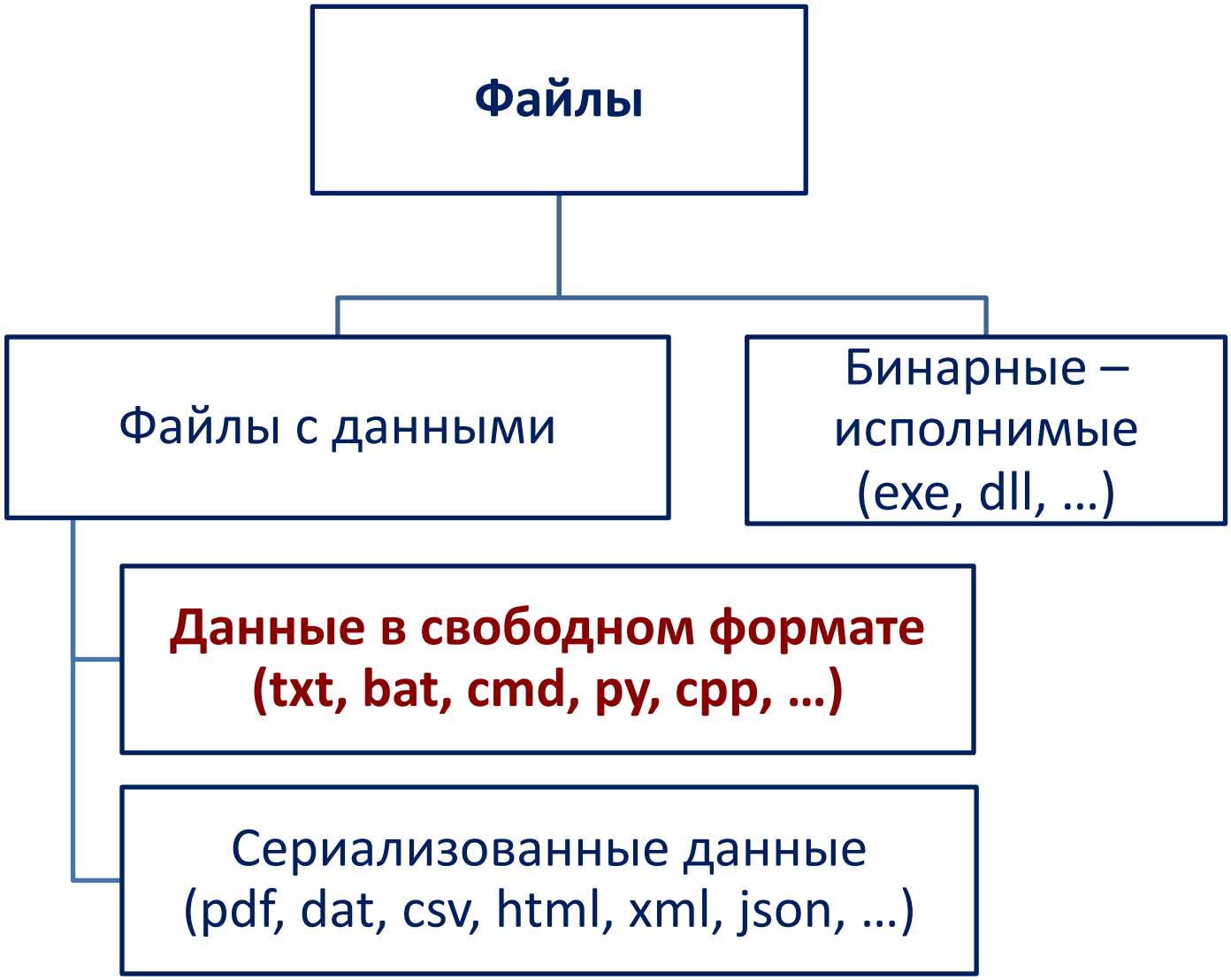
ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ.
ОБЛАСТИ ВИДИМОСТИ

```
>>> value = 10

>>> def assign_value():
        value = -10

>>> assign_value()
```

В какой области видимости находится переменная *value* в функции *assign_value()*?

Чему равно её значение в этой функции?

ФАЙЛОВЫЕ ФОРМАТЫ



РАБОТА С ТЕКСТОВЫМИ ФАЙЛАМИ

- Открытие:
  - file_object = open(‘имя файла’, [режим, кодировка])
- Чтение или запись:
  - Читаем с помощью методов file_object: read(), readline(), readlines() или в цикле for
  - Пишем с помощью file_object.write()
- Закрытие:
  - 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 FileExistsError 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 True если файл f был закрыт, и False 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\poem_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
src_file_name = fd.askopenfilename()
dst_file_name = fd.asksaveasfilename()
```

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

ИСКЛЮЧЕНИЯ. ПРОДОЛЖЕНИЕ

```
try:
    испытуемый код
except (exc1, exc2) [as variable1]:
    реакция на исключения группы 1
except excN [as variableN]:
    реакция на исключения группы N
[else:
    блок, “когда исключений не было”]
[finally:
    блок финальной обработки]
```

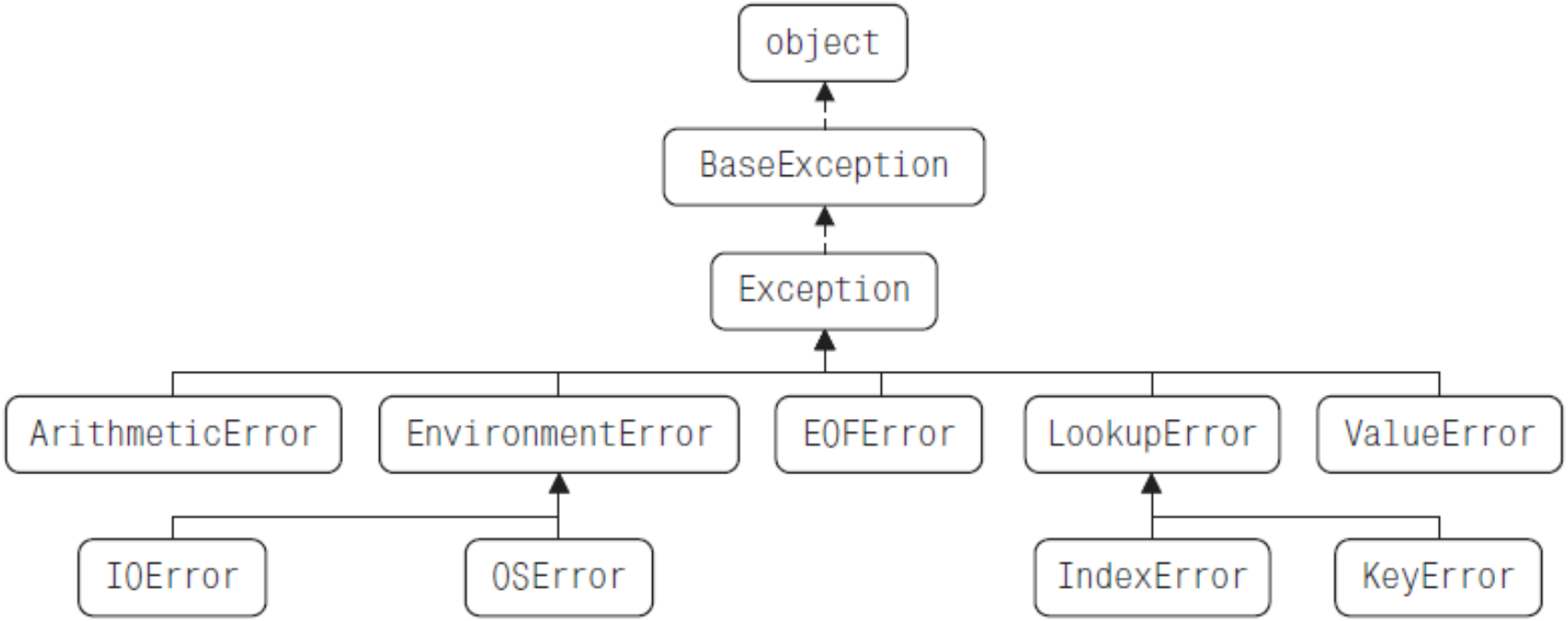
ПОРЯДОК ОБРАБОТКИ ИСКЛЮЧЕНИЙ

Потоки выполнения конструкции *try* … *except* … *finally*

ИСКЛЮЧЕНИЯ. ПРИМЕР ФИНАЛЬНОЙ ОБРАБОТКИ

- См. скрипт в no_blanks.py
- finally используется для надежной обработки ошибок
- finally вызывается **всегда**, вне зависимости от того было какое-то исключение или нет, предусмотрены для него обработчики или нет

ИЕРАРХИЯ ИСКЛЮЧЕНИЙ



Фрагмент иерархии классов исключений Python

ИЕРАРХИЯ ИСКЛЮЧЕНИЙ. ГДЕ ПОСМОТРЕТЬ?

- Если хотите почитать про системные исключения Python – смотрите доку:
  - https://docs.python.org/3/library/exceptions.html

- А чтобы увидеть исключения прямо в Python, используйте скрипт print_exceptions.py

РОДСТВЕННЫЕ ИСКЛЮЧЕНИЯ. ПРЕДОСТЕРЕЖЕНИЕ

- Несколько блоков except нужно располагать сверху вниз в порядке от более специализированных к более общим

```
>>> test_list = [1, 2, 3]
>>>
try:
    item = test_list[5]
except LookupError: # НЕВЕРНЫЙ ПОРЯДОК ИСКЛЮЧЕНИЙ
    print("Lookup error occurred")
except IndexError:  # ЭТА ЛОВУШКА ДОЛЖНА БЫТЬ ВЫШЕ
    print("Invalid index used")


Lookup error occurred
```

ИСКЛЮЧЕНИЯ.
СОБСТВЕННЫЕ ИСКЛЮЧЕНИЯ

- Пользовательское исключение – это наш собственный тип данных
- Создание своего типа исключения:
  class UserExceptionName(Exception): pass
- Генерация собственного исключения:
  raise UserExceptionName
- Это используется для:
  - Описания пользовательских типов ошибок
  - Управления потоком выполнения программы

ФУНКЦИЯ ENUMERATE()

```
>>> the_list = [10, 20, 30, 40]
>>> for tup in enumerate(the_list):
        print(tup)


(0, 10)
(1, 20)
(2, 30)
(3, 40)
>>> for tup in enumerate(the_list, 1):
        print(tup)


(1, 10)
(2, 20)
(3, 30)
(4, 40)
```

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ. ВИКТОРИНА

- План программы:
  - Представить тему и поздороваться
  - Пока не достигли конца файла:
    - Считывать блоки с вопросами
    - Задавать вопросы и запрашивать ответ
    - Проверять ответ на правильность
    - Подсчитывать количество верных ответов
  - Информировать о количестве очков
- См. вопросы викторины в py_struct.txt
- Разработанная по этому плану программа в каталоге Scripts/Quiz

ДОМАШНЕЕ ЗАДАНИЕ. ПРОГРАММА «ВИКТОРИНА» V 2.0

- Добавьте цену (вес) вопроса в структуре TXT-файла для указания уровня сложности вопроса
- В конце викторины сумма набранных очков должна учитывать вес каждого вопроса
- Добавьте хранение списка рекордов в отдельном файле **в виде словаря (*)**:
  "<*имя пользователя*>:<*рекорд*>\n"…
- Пользователь должен указывать свое имя при старте программы и видеть свое предыдущее достижение
- Создайте доп. эпизоды для проверки знаний по работе с исключениями и файлами

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science