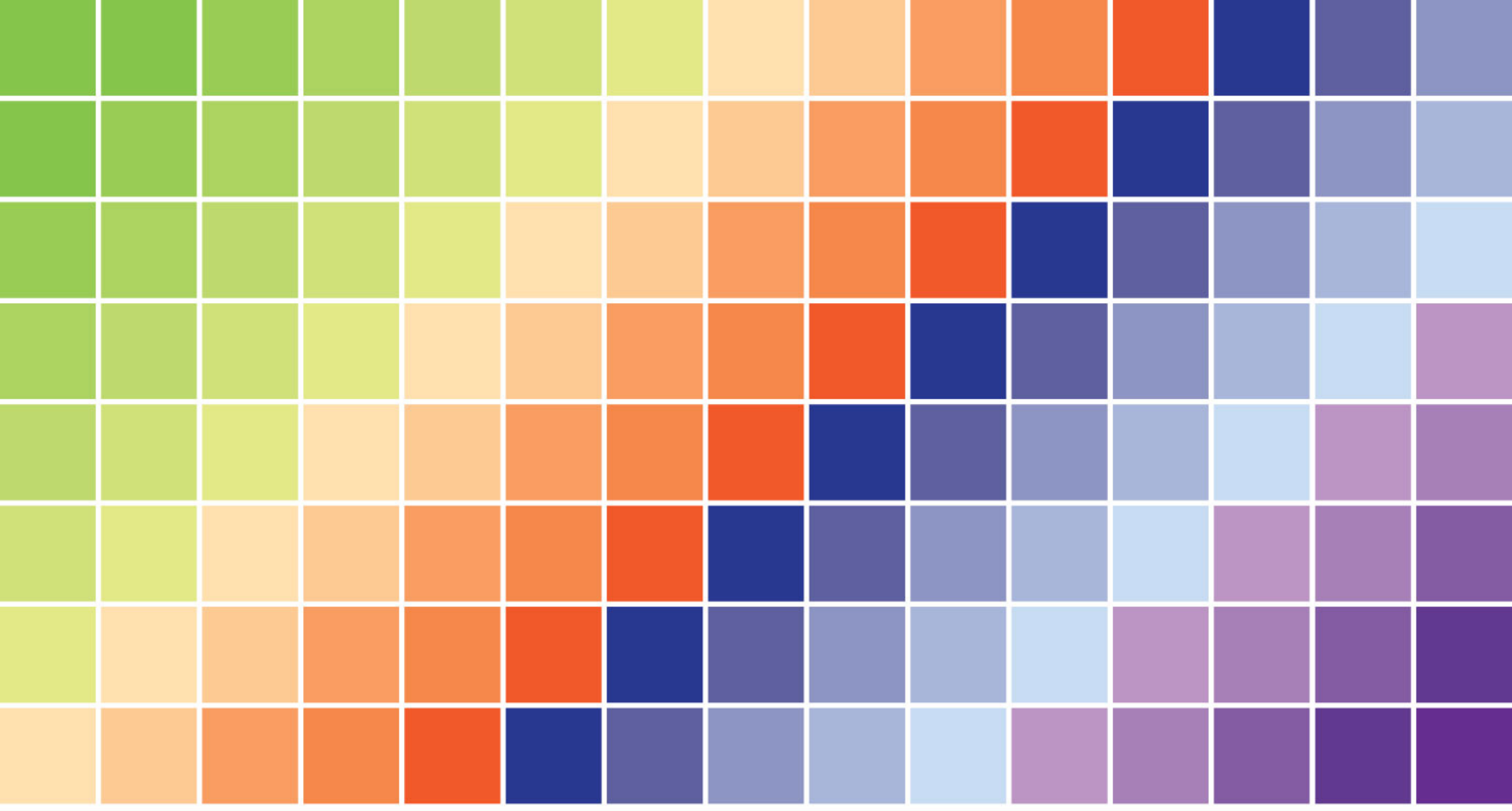


School of Computer Science

СОБСТВЕННЫЕ ФУНКЦИИ. ПРОСТРАНСТВА ИМЕН

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ

1. Какими способами можно проверить равна ли переменная None?

2. Какой способ сравнения с None лучше?

3. Что будет при выполнении арифметической операции с переменной, содержащей None?

4. Если функция возвращает None, то как получить результаты ее работы?

5. Для чего нужно явное присваивание None?

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ

6. Как проверить ссылаются ли две переменные на одну и ту же область памяти?

7. Как передать параметры через командную строку?

8. Как получить доступ к параметрам командной строки в программе на Python?

9. В каких случаях нужен range() в цикле for, а когда можно обойтись без него?

10. Где еще можно использовать range()?

ВОПРОСЫ ПО ПРОШЛОМУ ЗАНЯТИЮ

11. Чем кортеж (tuple) отличается от списка?

12. Что нужно сделать перед тем, как воспользоваться типом данных namedtuple?

13. Какие методы чтения текстового файлы вы знаете?

14. Как выполняется запись в текстовый файл?

15. Включаются ли переводы строк в данные при чтении/записи файла?

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ. КОРТЕЖ

– Какой способ создать кортеж является недопустимым?

1. >>> tuple_var = ()
2. >>> tuple_var = (1)
3. >>> tuple_var = (1, 2, 3)
4. >>> tuple_var = (1, "two", 3.0)
5. >>> tuple_var = (1,)
6. >>> tuple_var = tuple()

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ. КОРТЕЖ

- Есть 2 кортежа: >>> tup_1, tup_2 = (1,2), (4,3)

- Найдите недопустимую операцию:
  1. >>> tup_new = tup_1 + tup_2
  2. >>> tup_triple = tup_1 * 3
  3. >>> flag = 2 in tup_1
  4. >>> sorted(tup_2)
  5. >>> tup_1.sort()
  6. >>> list(reversed(tup_2))
  7. >>> tup_2 = tup_2[::-1]
  8. >>> val = tup_1[-1]
  9. >>> tup_1.index(val)

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ.
ОТКРЫТИЕ ФАЙЛА

– В Python файловая переменная создается с помощью функции:

1. file()
2. open()
3. fopen()
4. assign()
5. CreateFile()

ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ.
ЗАКРЫТИЕ ФАЙЛА

– Файл закрывается с помощью функции-метода:

1. file_obj.close()
2. file_obj.fclose()
3. file_obj.reset()
4. file_obj.flush()
5. file_obj.CloseFile()

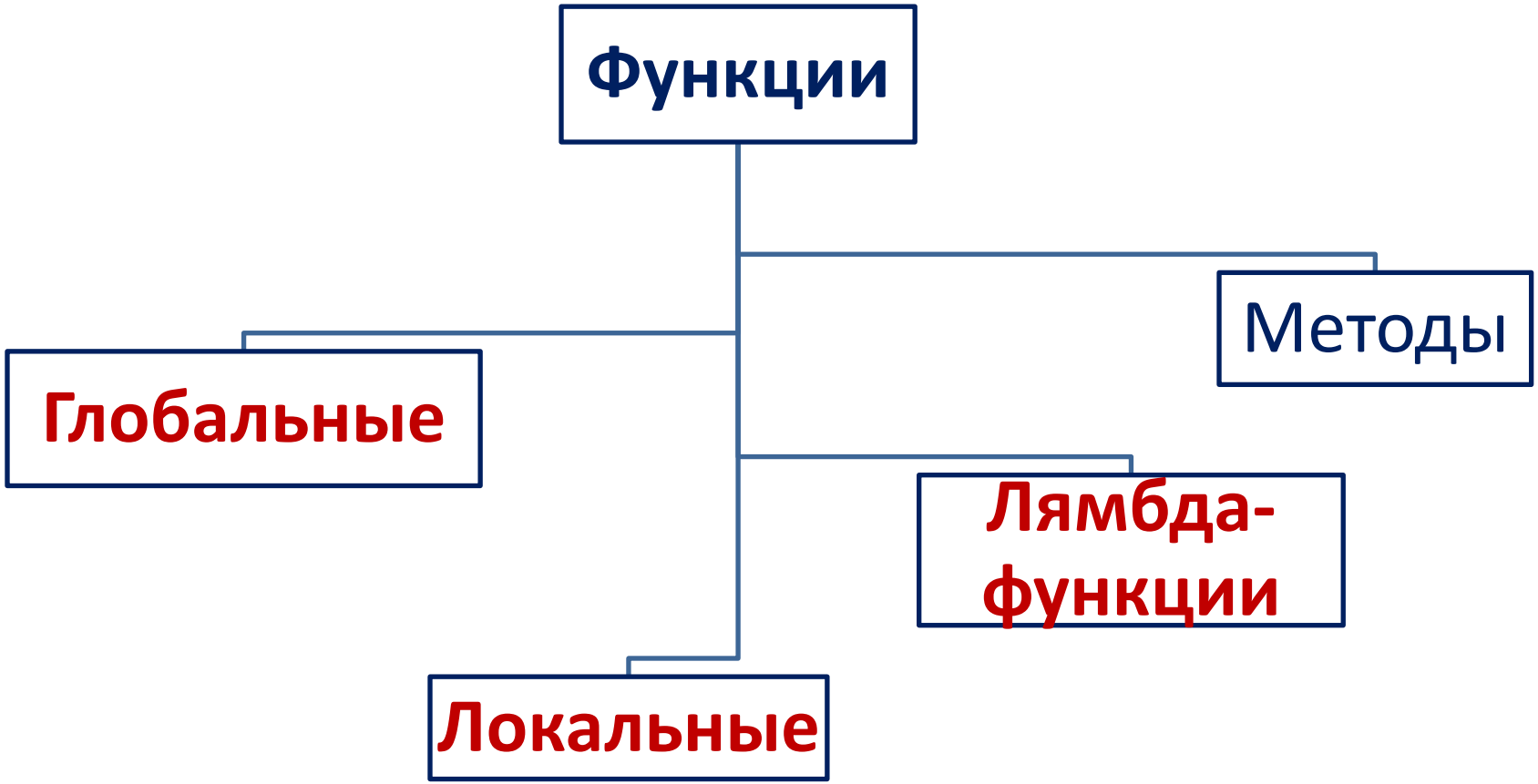
ВОПРОСЫ ПО ПРОШЛЫМ ЗАНЯТИЯМ.
А НУЖНО ЛИ ЗАКРЫВАТЬ ФАЙЛ?

- Если файл был открыт функцией open(), то закрытие этого файла методом close() является:
  - Обязательным
  - Желательным
  - Нежелательным

СВОИ ФУНКЦИИ. НЕОБХОДИМОСТЬ

- Структурирование кода:
  - Для часто вызываемого кода
  - В виде логически обособленных блоков кода
- Повторное использование кода
- Коллективная разработка
- Инкапсуляция и параметризация кода
- Новые области видимости переменных

ВИДЫ ФУНКЦИЙ В PYTHON



ФУНКЦИИ. СИНТАКСИС

Описание:

Отступы обяза-тельны!

```
def <имя>( [формальные параметры] ):
    <код функции>
    [return x]
```

Вызов: `<имя>( [аргументы] )`

Функция без **return** возвращает **None**

ПУСТЫЕ ФУНКЦИИ

```
def empty_func():
    pass
```

- Этот подход используется для того, чтобы отложить разработку кода на будущее
- Оператор pass можно использовать и в других блоках кода

DOC STRING

```
def empty_func():
    """ Describpe how it works
    """

    pass
```

ТИПИЧНЫЙ СОСТАВ DOC STRING

1. Типы параметров и возвращаемых значений
2. Описание того, что делает функция
3. Условия ее использования
4. Возбуждаемые исключения (если есть)
5. Примеры вызовов в стиле Shell

- Смотрите примеры в triangle.py

- Для пояснений хитрого алгоритма комментарии внутри исходного кода более предпочтительны, чем строки документации

ПЕРЕДАЧА ПАРАМЕТРОВ И ВОЗВРАТ ЗНАЧЕНИЙ

- Смотрите примеры в receive_and_return.py

- Именованные и позиционные параметры, значения по умолчанию:
  - Пример в birthday_wishes.py
  - Параметры со значениями по умолчанию должны объявляться последними
  - При вызове функции позиционные параметры должны передаваться в первую очередь

ПРАКТИЧЕСКОЕ ЗАДАНИЕ

- Рассмотрите библиотеку triangle.py
- Напишите функцию hypotenuse() в модуль triangle.py по скрипту NonFunc\hypot_len_v2.py
- Требования :
  - Функция должна принимать 2 параметра – положительные длины катетов
  - В функции должен присутствовать Doc String, выдающий справку в среде разработки

ХИТРОСТЬ СО ЗНАЧЕНИЯМИ ПО УМОЛЧАНИЮ

Значения по умолчанию вычисляются в точке определения функции в момент ее первого сканирования интерпретатором

```
>>> default_var = 5
>>> def double_print(param1, param2 = default_var):
        print(param1, param2)


>>> default_var = 6
>>> double_print(1)
1 5
>>> double_print(1, 2)
1 2
```

ПЕРЕДАЧА ПАРАМЕТРОВ ПО ССЫЛКЕ

Будьте осторожны со списками
в параметрах функции!

```
>>> def magic_list_append(p_list):
        p_list.append("Blue")


>>> my_list = ["Red", "Green"]
>>> magic_list_append(my_list)
>>>
>>> my_list
['Red', 'Green', 'Blue']
```

ПРИСВАИВАНИЕ ПАРАМЕТРА В ФУНКЦИИ

В момент присваивания в функции создается новая локальная переменная:

```
>>> my_list = ["Red", "Green"]
>>>
>>> def magic_list_assign(p_list):
        p_list = ["Orange", "Yellow"]


>>> magic_list_assign(my_list)
>>> my_list
['Red', 'Green']
```

ВОЗВРАТ МУЛЬТИ-ЗНАЧЕНИЙ ИЗ ФУНКЦИИ:

После return можно указывать кортеж:

```
>>> def multi_return():
        return 1, 2, 3

>>> res_tuple = multi_return()
>>> res_tuple
(1, 2, 3)
>>>
>>> res1, res2, re3 = multi_return()
>>> res1
1
>>> res2
2
>>>
```

АНОНИМНЫЕ ФУНКЦИИ

Создаются с помощью инструкции **lambda**:

lambda параметр1, параметр2, . . .:
выражение

```
>>> func = lambda x, y: x + y
>>> func(1, 2)
3
```

Анонимные функции могут содержать лишь **одно** выражение, но и выполняются они быстрее.
Их хорошо применять со встроенными функциями - *map, filter, reduce, sort*

АНОНИМНЫЕ ФУНКЦИИ

Фильтрация последовательности *a_list* с помощью *filter()*

```
>>> a_list = [2, 18, 9, 17, 8, 12, 27]
```

С использованием **def**:

```
>>> def filter_func(x):
        if x % 3 == 0:
            return True
        return False

>>> print(list(filter(filter_func, a_list)))

# [18, 9, 12, 27]
```

С использованием **lambda**:

```
>>> print(list(filter(lambda x: x % 3 == 0,
                      a_list)))

# [18, 9, 12, 27]
```

ПРОСТРАНСТВА ИМЁН

Пространство имён – отображение между идентификаторами и объектами

У блока кода, выполняемого в Python, есть три пространства имен:

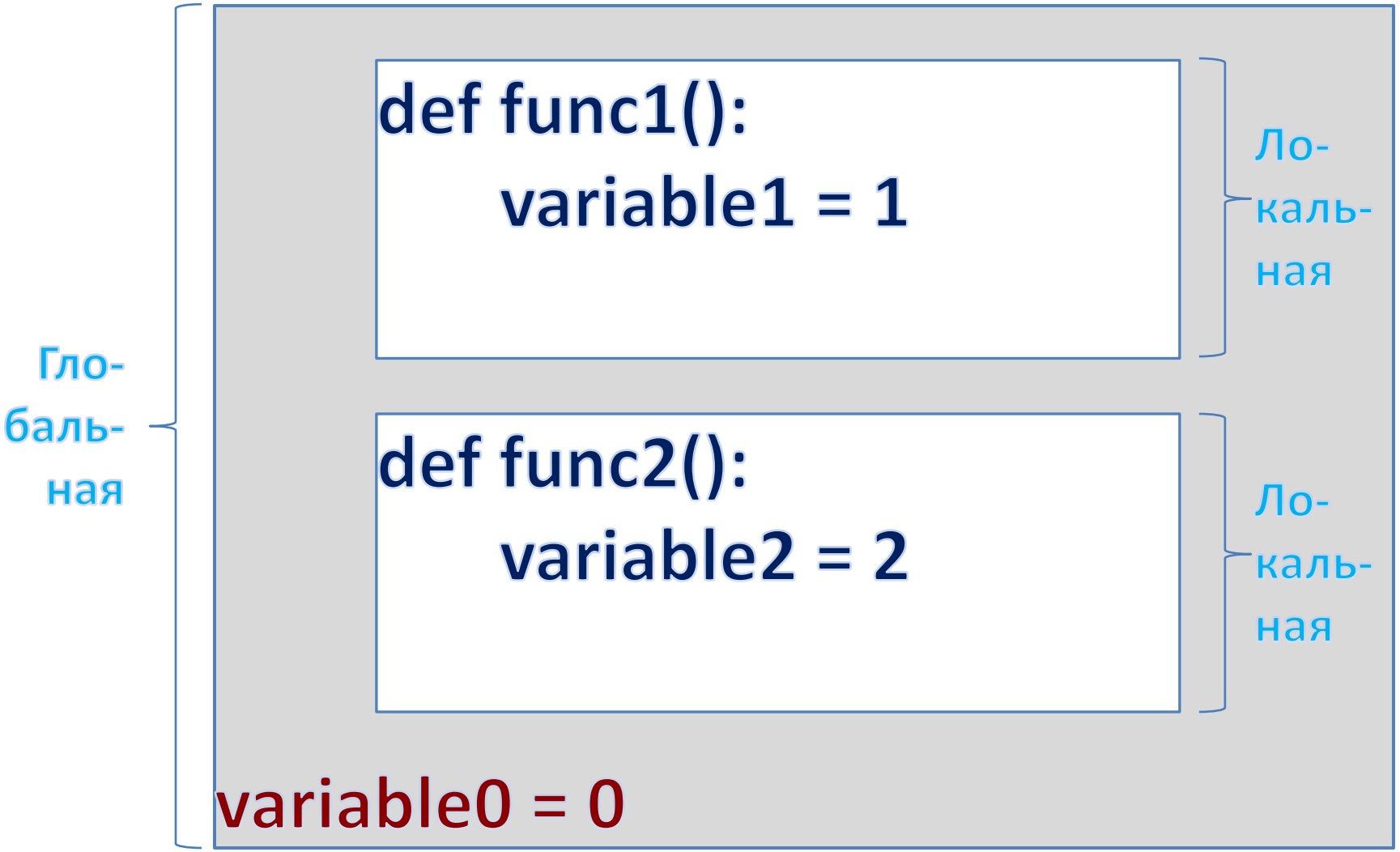
- Локальное
- Глобальное
- Встроенное

ПРОСТРАНСТВА ИМЁН

Порядок поиска идентификатора:

- Локальное
- Глобальное
- Встроенное
- Исключение NameError

ОБЛАСТИ ВИДИМОСТИ



ПЕРЕКРЫТИЕ ПЕРЕМЕННЫХ. СТЕК ИСПОЛНЕНИЯ

- Смотрите примеры в scopes.py

- При исполнении функция занимает память для локальных переменных
- После исполнения память освобождается
- Память выделяется в ***стеке***
- Проследим пошаговое исполнение скрипта convert_min_sec.py

- Используем для этого сайт http://www.pythontutor.com

ПРАКТИЧЕСКОЕ ЗАДАНИЕ. СИСТЕМЫ СЧИСЛЕНИЯ

- Смотрите шаблон в convert_bin_dec_template.py

- Вспомните правила преобразования из десятичной системы счисления в двоичную и обратно
- Этот сайт поможет вспомнить: cdn.cs50.net/2016/x/psets/0/pset0/bulbs.html

- Напишите функции to_binary() и to_decimal() по заданным требованиям
- Найдите стандартные функции Python, которые делают такие же преобразования

ДОМАШНЕЕ ЗАДАНИЕ. БИБЛИОТЕКА CIRCLE

- Создайте свою библиотеку circle.py
- В ней должны быть функции:
  - area(radius) – площадь круга
  - sector_area_len(radius, length) – площадь сектора круга через длину дуги
  - sector_area_ang(radius, angle) – площадь сектора круга через угол
  - circumference(radius) – длина окружности
  - volume(radius, height) – объём цилиндра
- Заполнение Doc String – обязательно для модуля и для каждой функции
- Значение числа Pi смотрите в math.pi

ДОМАШНЕЕ ЗАДАНИЕ. ГИСТОГРАММА В ТЕКСТОВОМ ФАЙЛЕ

- Создайте выходной файл с гистограммой значений, построенной на основании данных из файла grades.txt

- Разбиение на шаги «сверху-вниз»:
  - Прочитать входной файл
  - Посчитать количество оценок по диапазонам
  - Записать гистограмму в выходной файл

- См. структуру программы в grade_histogram_template.py и grade_template.py в каталоге HomeTask

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science