Алгоритмы и структуры данных на Python

Урок 6

Работа с динамической памятью

Представление в памяти коллекций. Управление памятью.

Вопросы

1. Как работает механизм выделения памяти?
2. Сколько памяти занимают переменные программы?
3. Как управлять памятью?

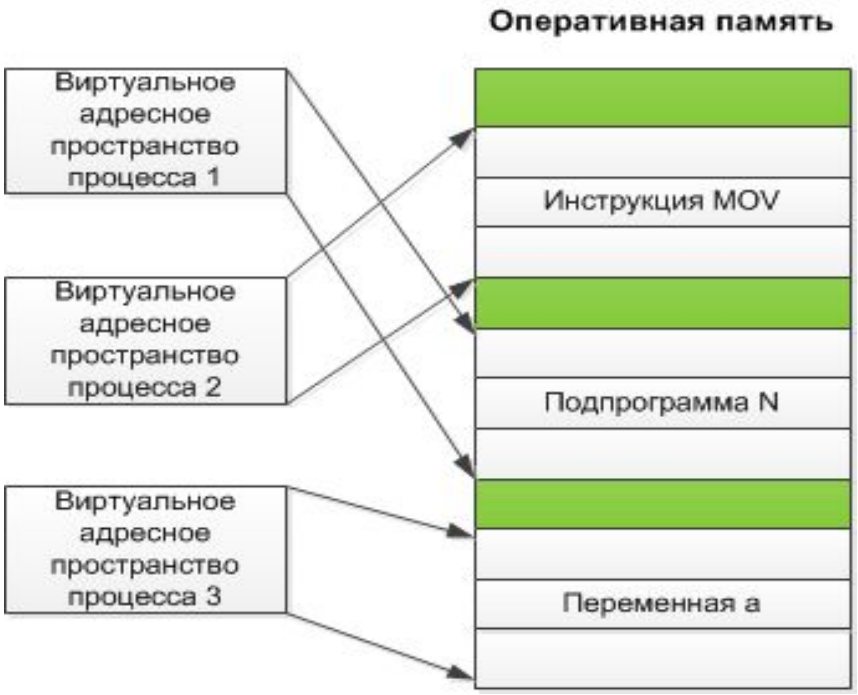
Цели урока

- Изучить представление данных в памяти
- Изучить способы работы с памятью

План урока

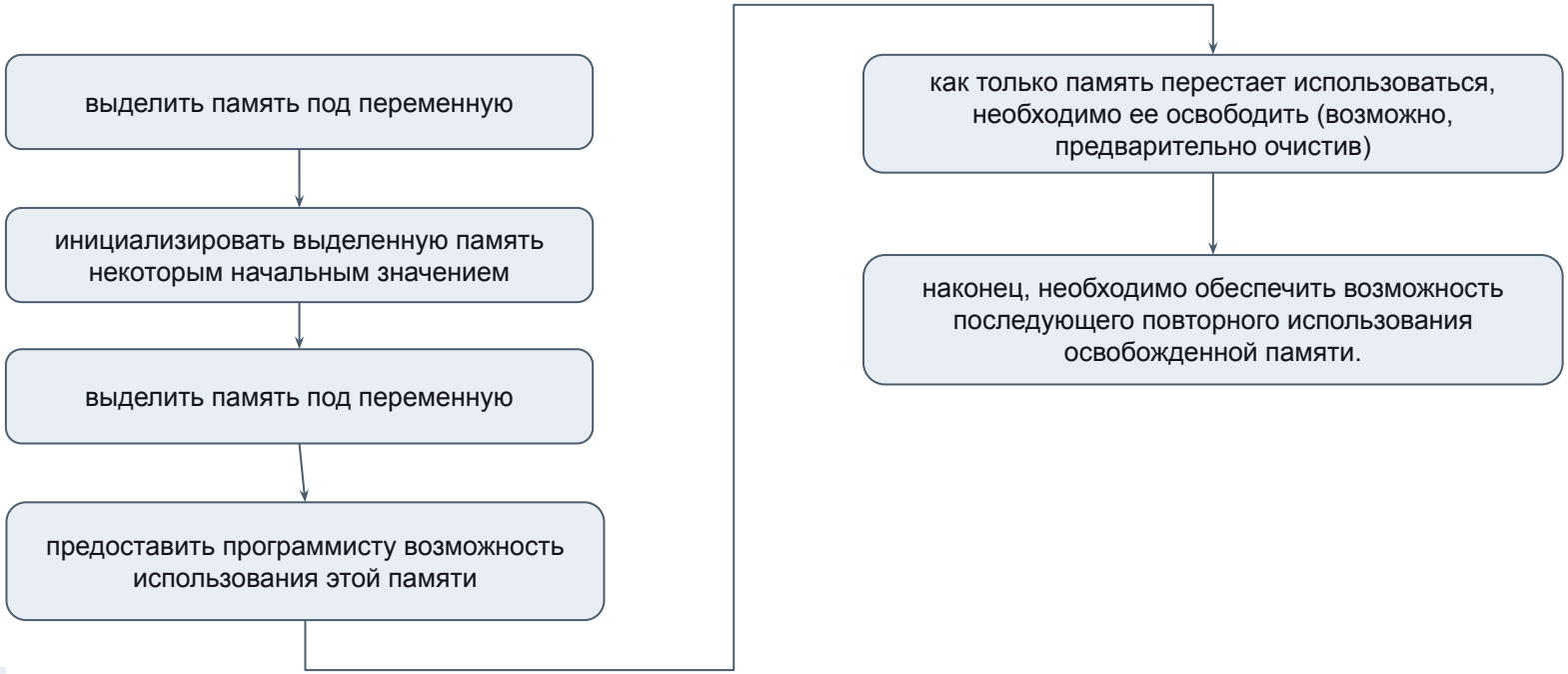
- Управление памятью с точки зрения разработчика компилятора
- Управление памятью – важнейшая особенность языка Python

Управление памятью с точки зрения разработчика компилятора



Управление памятью с точки зрения разработчика компилятора

Основные фазы работы с памятью



Проблемы управления памятью

Память не бесконечна

Явный механизм управления памятью

Способы выделения памяти

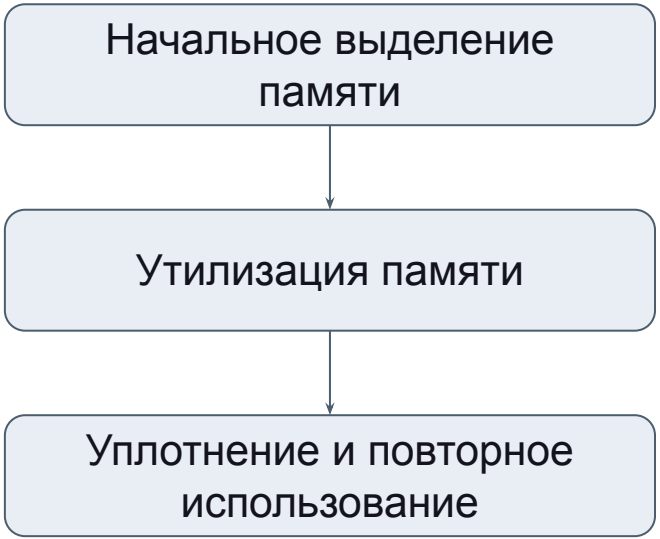
СТАТИЧЕСКАЯ

Статическая информация, т.е. информация, известная во время компиляции

ДИНАМИЧЕСКАЯ

Динамическая информация, т. е. сведения, неизвестные во время компиляции, но становящиеся известными во время выполнения программы

Фазы управления памятью

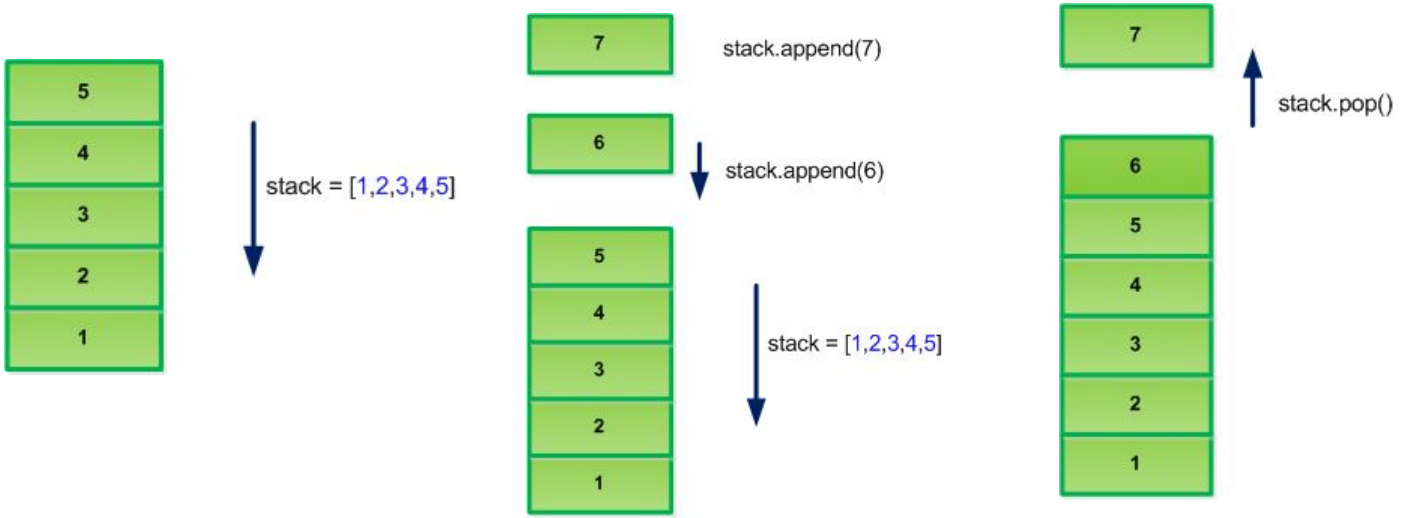


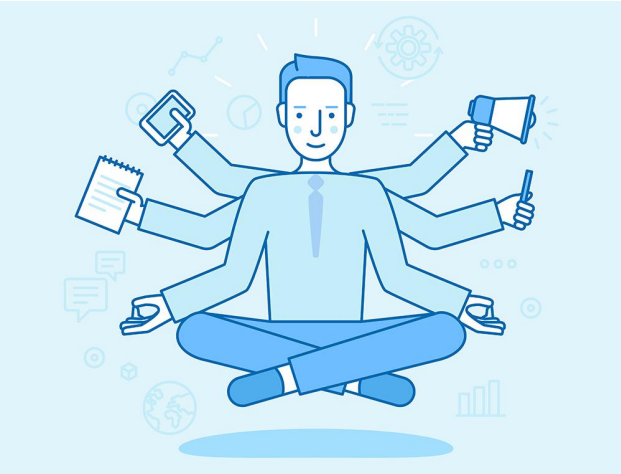
Основные методы управления памятью

- Статическое распределение памяти
- Стековое распределение памяти
- Представление памяти в виде кучи (heap)

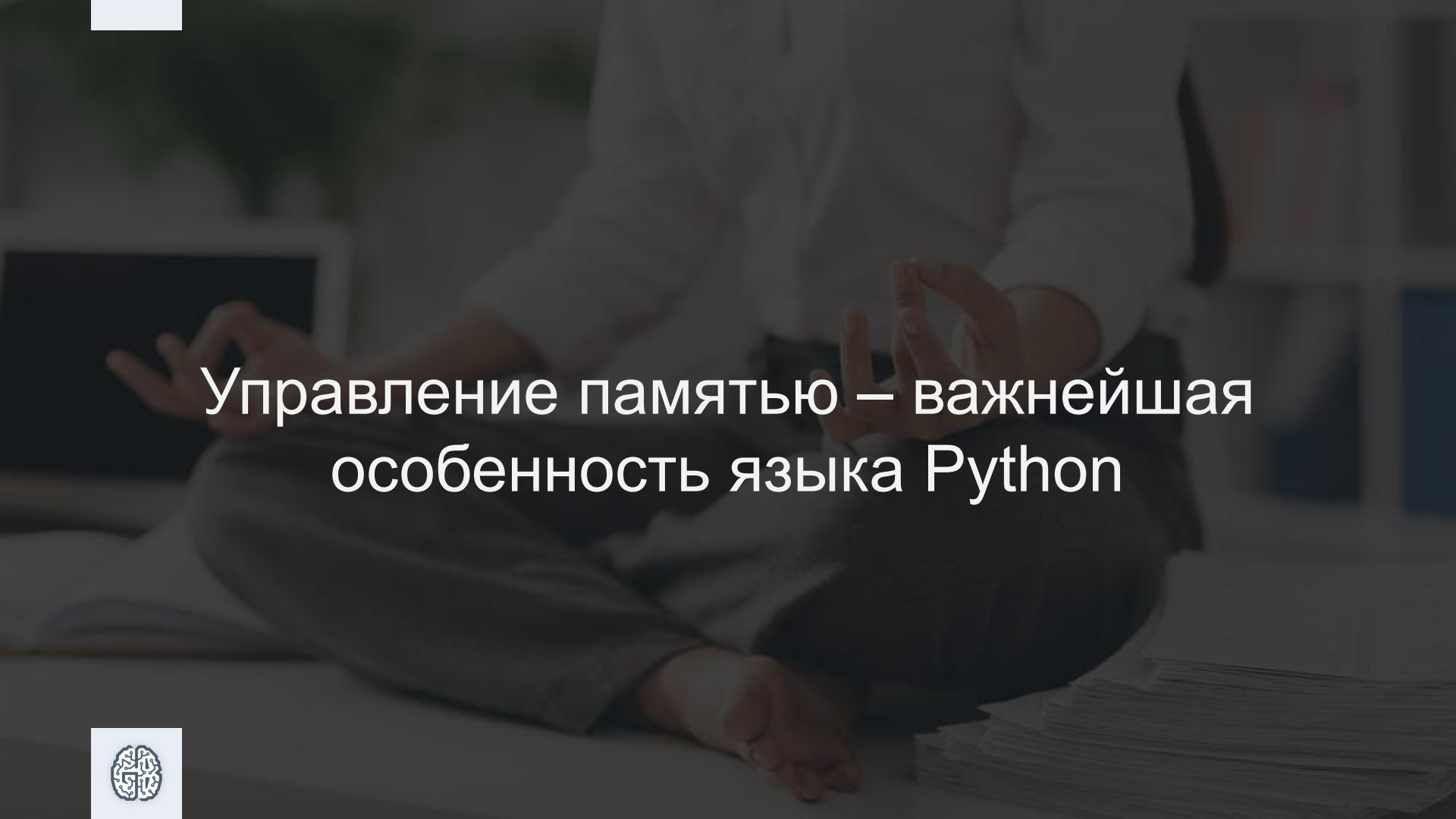
Статическое распределение памяти на примере двумерного массива

Стековое управление памятью





Управление кучей



Управление памятью – важнейшая особенность языка Python

Сколько памяти занимает 1 миллион целых чисел?

Символ	Значение C	Значение Python	Длина на 32битной машине
c	char	Строка из одного символа	1
i	int	int	4
l	long	int	4
L	unsigned long	int	4
d	double	float	8

В результате получим следующие данные:

Тип	Имя в Python	Формат	Формат, для вложенных объектов	Длина на 32 bit	Длина на 64 bit	Память для GC*
Int	PyIntObject	LLl		12	24	
float	PyFloatObject	LLd		16	24	
str	PyStringObject	LLLli+c*(длина+1)		21+длина	37+длина	

В результате получим следующие данные:

Тип	Имя в Python	Формат	Формат, для вложенных объектов	Длина на 32 bit	Длина на 64 bit	Память для GC*
unicode	PyUnicodeObject	LLLLIL	L*(длина+1)	28+4*длина	52+4*длина	
tuple	PyTupleObject	LLL+L*длина		12+4*длина	24+8*длина	Есть
list	PyListObject	L*5	L*длину	20+4*длина	40+8*длина	Есть

В результате получим следующие данные:

Тип	Имя в Python	Формат	Формат, для вложенных объектов	Длина на 32 bit	Длина на 64 bit	Память для GC*
Set/ frozen set	PySetO bject	L*7+(lL) *8+lL	LL* длина	(<=5 элементов) 100 (>5 элементов) 100+8*длина	(<=5 элементов) 200 (>5 элементов) 200+16*длина	Есть
dict	PyDictO bject	L*7+(lLL)*8	lLL*длина	(<=5 элементов) 124 (>5 элементов) 124+12*длина	(<=5 элементов) 248 (>5 элементов) 248+24*длина	Есть

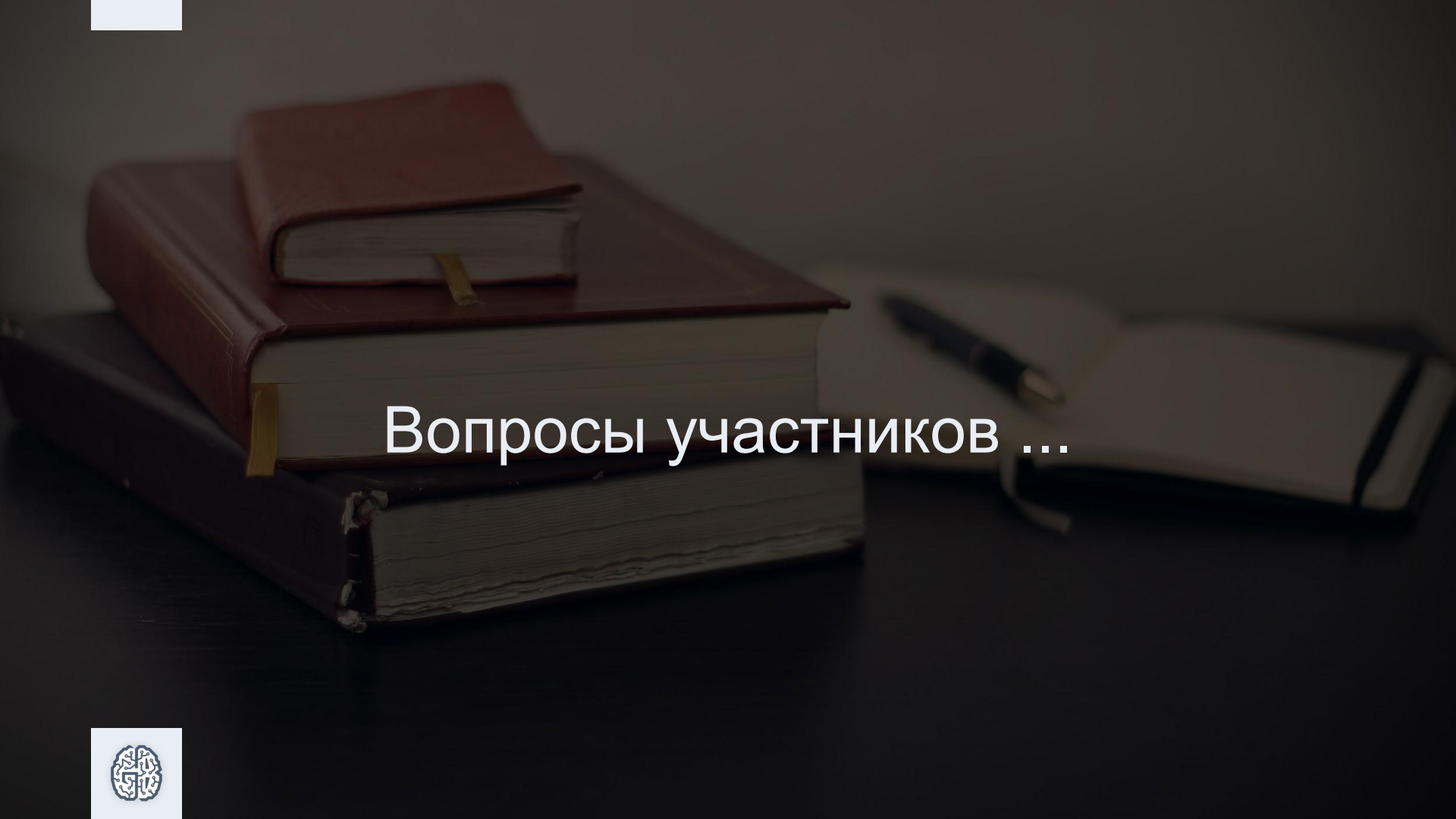
Сколько памяти занимает 1 миллион целых чисел?

Чтобы сохранить 1 миллион целых чисел, потребуется 11.4 мегабайт (12*10^6 байт) на сами числа и дополнительно 3.8 мегабайт (12 + 4 + 4*10^6 байт) на кортеж, который будет хранить ссылки на них.

Практическое задание

1. Подсчитать, сколько было выделено памяти под переменные в программах, разработанных на первых трех уроках. Проанализировать результат и определить программы с наиболее эффективным использованием памяти.

Примечание: Для анализа возьмите любые 1-3 ваших программы или несколько вариантов кода для одной и той же задачи. Результаты анализа вставьте в виде комментариев к коду. Также укажите в комментариях версию Python и разрядность вашей ОС.


Вопросы участников ...