АКАДЕМИЯ АЙТИ

День 19

academyit.ru

* Ветвления.
* Оператор if.
* Базовая форма цикла while.
* Операторы break и continue.
* Перебор (for).
* Генераторы словарей, списков, множеств.

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

Поверка домашнего задания

Задание 1.

Александр решил каким-то образом отобразить в тексте BACKSPACE (т.е. удаление последнего символа).

Он подумал, что символ «@» отлично для этого подходит.

Напишите функцию cleaned_str(st), которая будет выполнять BACKSPACE в его строках:
гр@оо@лк@оц@ва -> голова
сварка@@@@@лоб@ну@ -> слон

```
print(cleaned_str('гр@оо@лк@оц@ва'))
print(cleaned_str('сварка@@@@@лоб@ну@'))
```

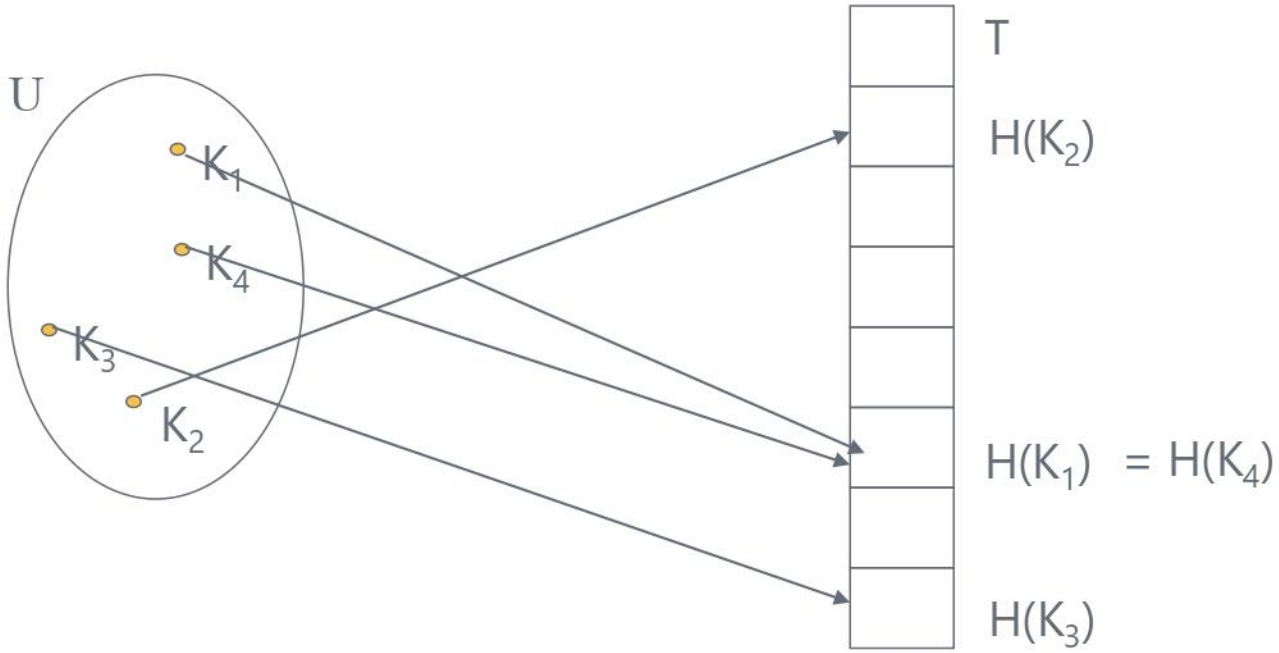
Структуры данных. Стеки. Очереди. Очереди с приоритетами.

Структуры данных

- Структура данных – контейнер (коллекция), в котором хранятся данных в определенном формате
- Делятся на **линейные**, которые образуют последовательность при обходе данных (пример – списки) и **нелинейные** (множества, словари)
- Простейшая линейная структура – список. Поддерживают обращение по индексу, вставку, удаление элементов. В Питоне вставка и удаление производятся быстро только в конец списка.

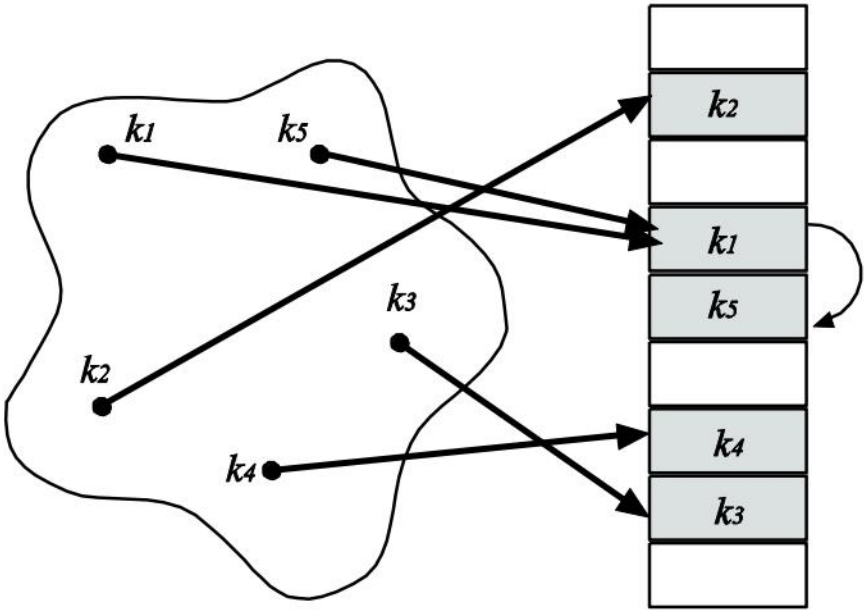
Множества и словари

- Множества и словари реализованы на основе хэш-таблиц
- Хэш-функция – на основе ключа вычисляет некоторое числовое значение, которое потом используется для вычисления номера элемента в хэш-таблице



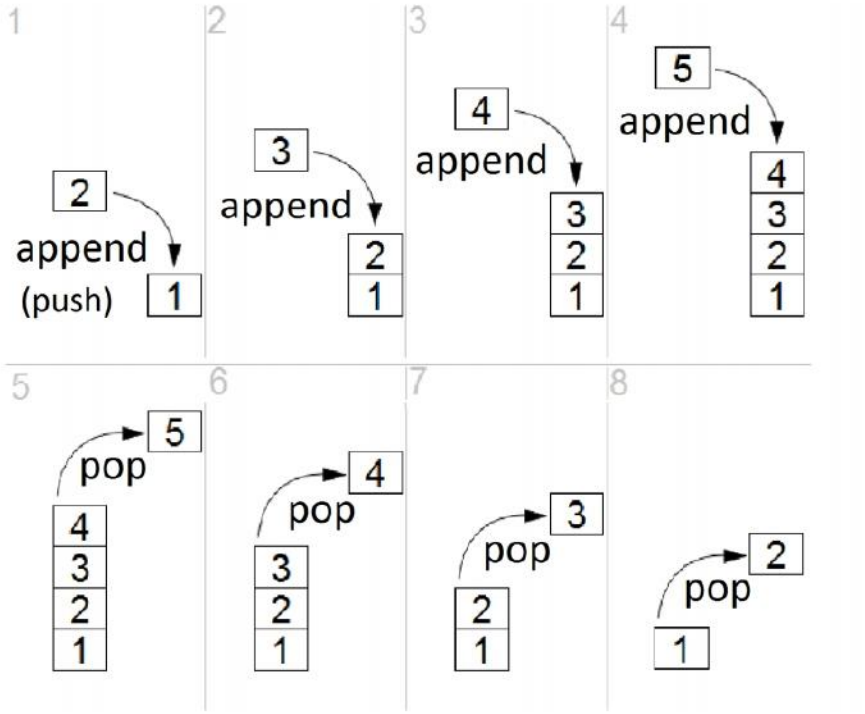
Коллизии в хэш-таблицах

- При хэшировании возможны коллизии, когда для разных ключей совпадают значения хэш-функции
- В случае коллизий для вычисления номера кроме ключа добавляется параметр – номер попытки
- Быстро реализуются добавление, удаление и поиск элемента по ключу
- Сложность операций определяется коэффициентом заполнения – отношением количества заполненных ячеек к количеству пустых



Стек

- Стек - структура данных, в которой получить доступ и удалить можно только тот элемент, который был добавлен последним (LIFO – Last In First Out).
- Для реализации стека в Python можно использовать список, добавлять в конец методом append, удалять из конца методом pop



Реализация стека массивом фиксированного размера

- Массив a = [0] * n
- Для стека требуется один указатель на голову h = 0
- При добавлении элемента k в голову стека записываем значение и увеличиваем указатель h.

  a[h] = k

  h = (h+1) % n
- Для удаления из головы стека требуется обратная операция:

  h = (h – 1) % n

  k = a[h]

Задача №51. Правильная скобочная последовательность

Рассмотрим последовательность, состоящую из круглых, квадратных и фигурных скобок. Программа дожна определить, является ли данная скобочная последовательность правильной.

Пустая последовательность является правильной. Если `A` – правильная, то последовательности `(A)`, `[A]`, `{A}` – правильные. Если `A` и `B` – правильные последовательности, то последовательность `AB` – правильная.

Входные данные

В единственной строке записана скобочная последовательность, содержащая не более 100000 скобок.

Выходные данные

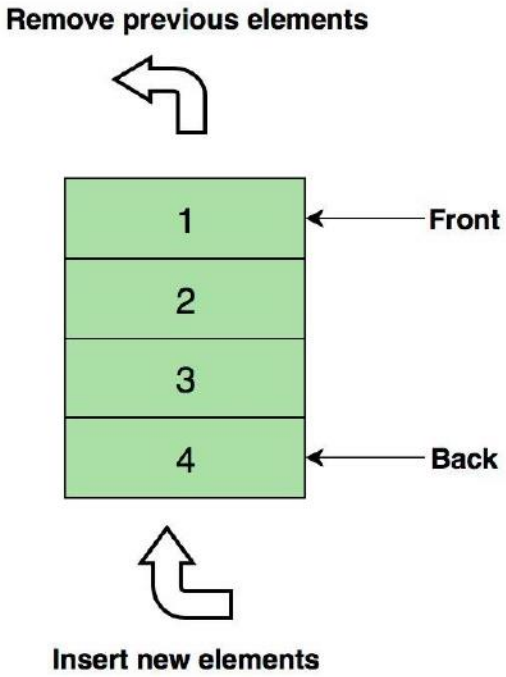
Если данная последовательность правильная, то программа должна вывести строку `yes`, иначе строку `no`.

Примеры

входные данные
`()[]`
выходные данные
yes

Очередь

- **Очередь** - структура данных, в которой удалить можно только тот элемент, который был добавлен первым (FIFO).
- Для реализации очереди можно использовать список.
- Добавление происходит в конец, а удаление - из начала списка pop(0). Скорость удаления будет низкой!



Реализация очереди массивом фиксированного размера

- Массив a = [0] * n
- Для очереди требуются два указателя – на голову h и хвост списка t
- Добавление в хвост очереди:

  a[t] = k
  t = (t+1) % n
- Удаление из головы очереди:

  k = a [h]
  h = (h+1) % n
- Очередь пуста, если h == t

Двусторонняя очередь Python

- Класс deque() из модуля collections - это двусторонняя очередь.
- deque() поддерживает добавление и извлечение элементов последовательности с любой стороны с производительностью O(1)
- Добавление в конец – append, в начало – appendleft
- Удаление с конца – pop, с начала – popleft
- Аналогично спискам есть методы count, index, insert, remove, reverse

Структурное программирование. Модули. Проектирование структуры программы.

Проблемы при разработке больших программ

- сложность решаемых задач
- сложность формального определения требований к программным системам
- изменчивость требований
- необходимость повторного использования кода
- коллективная разработка
- человеческий фактор

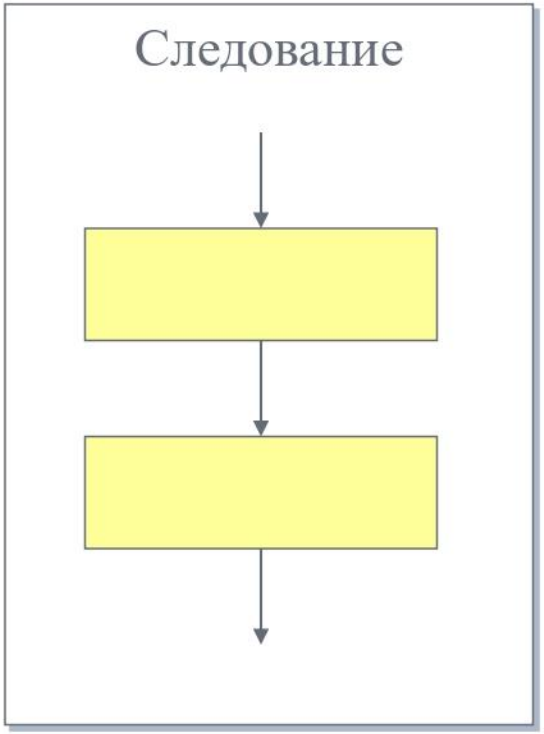
Структурное программирование

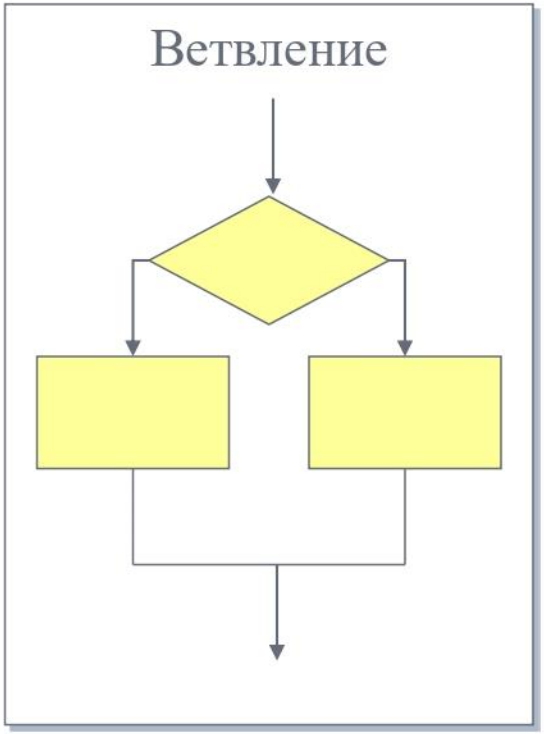
- Общие принципы и правила проектирования, разработки и оформления программ с целью облегчения процессов их создания и тестирования, повышения производительности труда программистов и улучшения понимания программ
- Структура программы и алгоритм решения задачи должны быть легкими для понимания, простыми для доказательства правильности и удобными для модификации
- Отказ от беспорядочного стиля в программировании и следование некоторым принципам, что особенно важно при разработке больших программных систем
- Основоположники структурного программирования – Эдсгер Дейкстра и Никлаус Вирт (конец 60-х годов)

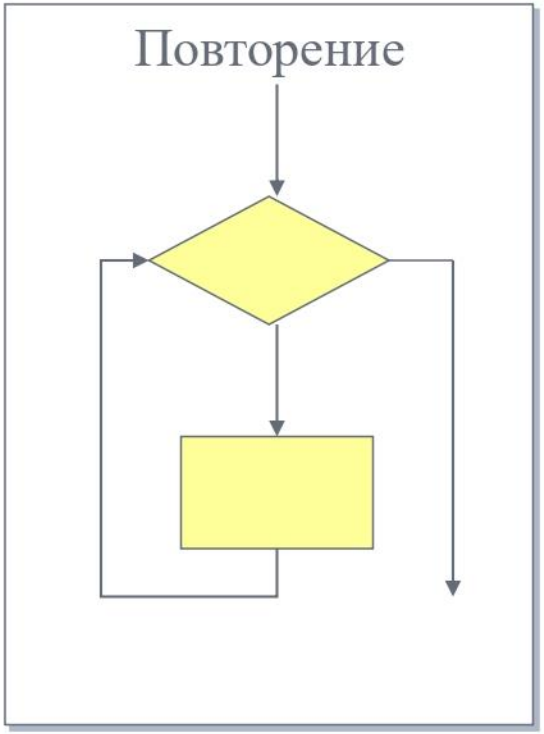
Принципы структурного программирования

- Программа строится из основных алгоритмических конструкций
- Модульная декомпозиция
- Пошаговая детализация программы
- Нисходящее или восходящее проектирование

Основные алгоритмические конструкции







Логическая структура программы может быть выражена комбинацией трех базовых структур

Модульная декомпозиция

- Исходный код программы разбивается на отдельные небольшие модули
- Модули могут быть скомпонованы вместе для создания более крупного приложения, решающего задачу
- В Питоне единицы модульности – это функции, классы, собственно модули и пакеты

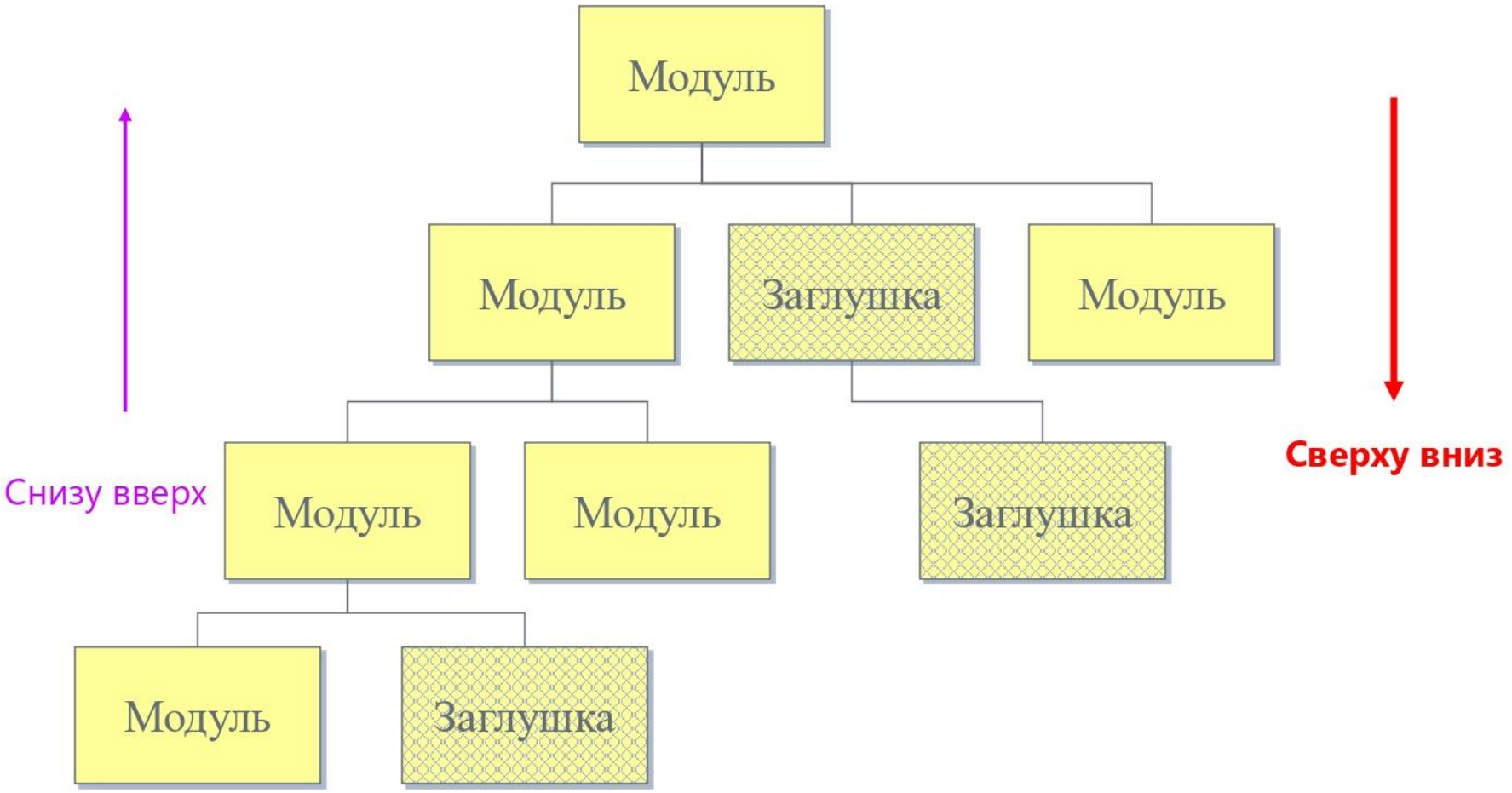
Преимущества модулей

- Простота: Код модуля обычно решает одну задачу, меньше код – меньше ошибок
- Модифицируемость: Если модули слабо зависят друг от друга, то появляется возможность изменять модуль, не изменяя другие части программы.
- Коллективная разработка: Каждый модуль может разрабатываться программистом независимо от других
- Повторное использование кода: Функции и классы из модуля могут быть использованы повторно другими приложениями
- Область действия: В модуле определено отдельное пространство имен, что помогает избежать совпадений между именами в разных областях программы

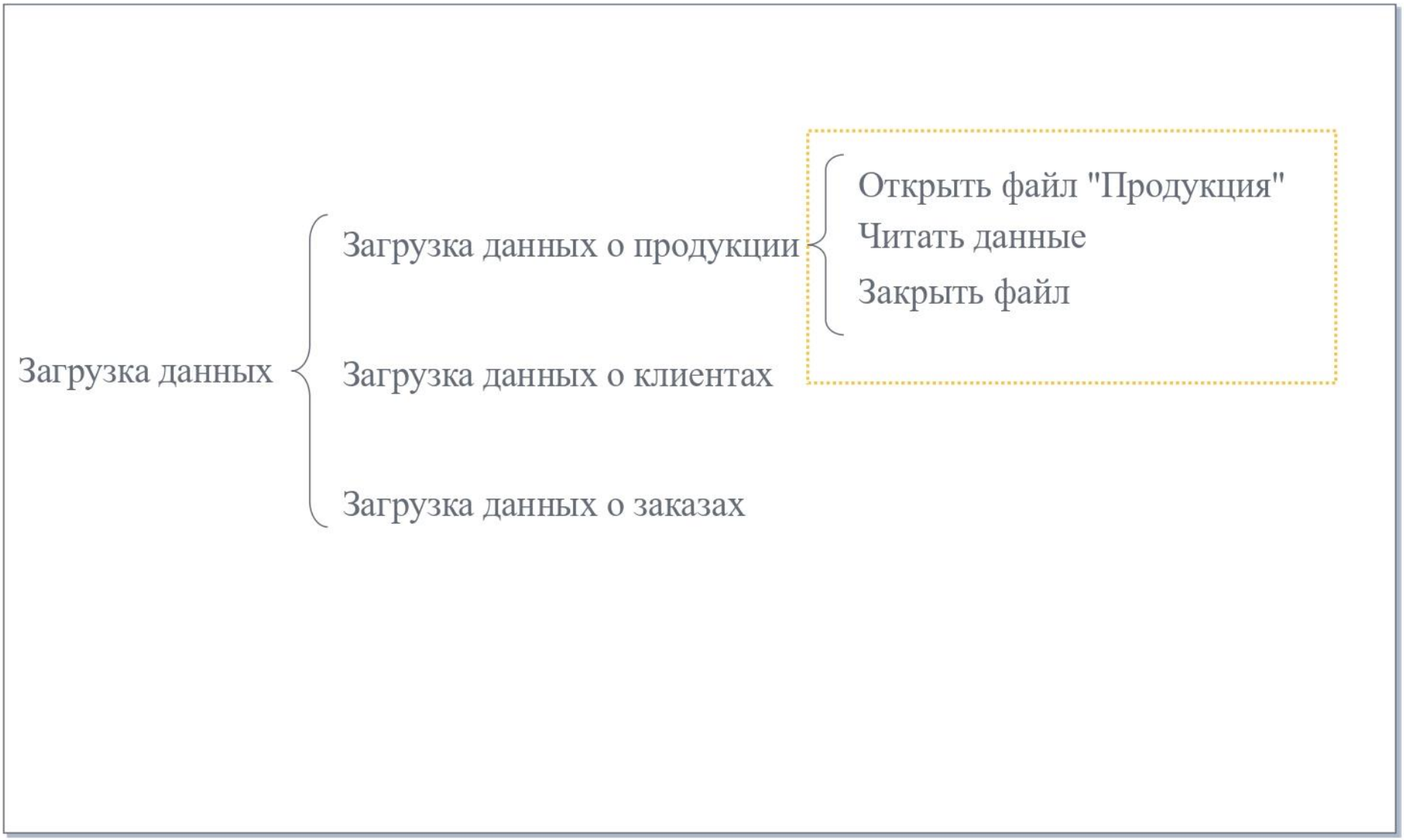
Подходы к проектированию программ

- Структурное программирование предполагает использование двух подходов к разработке программ – нисходящего (сверху-вниз) и восходящего (снизу – вверх) проектирования
- Нисходящее проектирование предполагает сначала разработку алгоритма в целом из крупных блоков, а затем пошаговую детализацию каждого блока
- При этом нереализованные блоки заменяются на заглушки, что делает возможным тестирование программы
- Восходящее проектирование используется, когда точный алгоритм программы еще не разработан, но уже есть возможность для написания отдельных функций, реализующих конкретные действия
- Есть проблемы с тестированием отдельных функций

Нисходящая и восходящая разработка



Пошаговая детализация текста программы



Модули в Python

- Модуль может быть написан на Python
- Модуль может быть написан на C и динамически подгружен во время исполнения, как модуль re (regular expression)
- Для создания модуля код Python сохраняется в отдельный файл с расширением .py
- Модуль может быть подключен с помощью import <имя модуля>, в этом случае объекты из модуля доступны с помощью префикса <имя модуля>.<имя объекта>

```
import math
print(math.pi)
```

- При подключении имя модуля может быть изменено

```
import random as rnd
```

Модули в Python

- Из модуля могут быть импортированы объекты с помощью

```
from <module_name> import <name1>, <name2>,….
```

Все объекты можно импортировать с помощью *

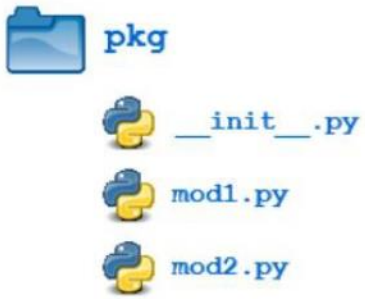
```
from <module_name> import *
```

- Для просмотра содержимого модуля используется функция **dir**
- Расположение модулей при поиске:
  - Текущая папка
  - Папки из списка в системной переменной PYTHONPATH
  - В списке папок, определённых и настроенных во время установки Python.
- Посмотреть список папок для поиска можно в переменной path из модуля sys

```
>>> import sys
>>> sys.path
['', 'C:\\Users\\john\\Documents\\Python\\doc', 'C:\\Python36\\Lib\\idlelib',
'C:\\Python36\\python36.zip', 'C:\\Python36\\DLLs', 'C:\\Python36\\lib',
'C:\\Python36', 'C:\\Python36\\lib\\site-packages']
```

Пакеты Python

- По мере роста количества модулей становится сложно управлять ими
- Пакеты позволяют иерархически структурировать модули с использованием точечной нотации. Таким образом, пакеты помогают избежать коллизий между именами модулей.
- Пакет – это просто папка с определенным именем, включающая несколько модулей.
- Если файл с именем __init__.py присутствует в каталоге пакета, он вызывается при импорте пакета или модуля в пакете



Архитектура пакета

Пакеты Python

- Код инициализации пакета может содержать переменную

```
__all__ = [  'mod1',  'mod2',  'mod3',   'mod4' ]
```

- Это список пакетов, которые импортируются при импорте *
- Пакеты могут содержать вложенные подпакеты и глубина вложенности не ограничена.

Иллюстрация иерархии подпакетов

Задачи

Задача 20.

```
figures
  |_ circle
        |_ __init__.py
        |_ code.py
  |_ square
        |_ __init__.py
        |_ code.py
  |_ triangle
        |_ __init__.py
        |_ code.py
  |_ __init__.py
```

Создайте пакет 'figures', состоящий из трех подпакетов: 'triangle', 'circle', 'square'.

В каждом подпакете будем иметь файл code.py, где создадим ряд функций.

Ваша итоговая задача – позволить человеку, загрузившему ваш пакет, иметь возможность напрямую импортировать все функции из подпакетов.
Например, он может написать так: 'from figure import circle_area'.

Также вы, как разработчик, после написания всей библиотеки решили поменять ее имя на 'figures'.

Постарайтесь сделать код таким, чтобы это не заставило вас переписывать все внутренние импорты с учетом нового именования.

– для пакета 'circle': функции circle_perimeter() – вычисляет длину окружности, circle_area() – вычисляет площадь окружности. Еще заведем переменную default_radius = 5, которая будет скрыта при импорте модуля. Ее назначение – дефолтный радиус для окружности, если пользователь не введет свой. Обе функции принимают на вход только радиус.

– для пакета 'triangle': функции triangle_perimeter() – вычисляет периметр треугольника, triangle_area() – вычисляет площадь фигуры. Дополнительно создадим три переменные (длины сторон треугольника): a = 7, b = 2, c = 8, которые также не будут видны при импорте.
На вход функциям передается длина трех сторон (если пользователь ничего не введет, то используются значения по умолчанию).

– для пакета 'square': функции square_perimeter() – вычисляет периметр квадрата, square_area() – вычисляет площадь фигуры. Дополнительная переменная a = 15 не доступна при импорте и принимается функциями, если пользователь не предоставил свои размеры стороны квадрата.

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru