

Конспект лекций по языку программирования Python

Автор конспекта – Александр Бердюгин

Конспект выполнен по лекции <https://youtu.be/5g-MHZ0MzZY>
(Учим python за 7 часов! Уроки Python – полный курс обучения
программированию на python с нуля).

Освоил все 7 часов лекций с 29 января по 12 мая 2022 г. (около 15 недель) с подробным конспектированием в тетради и созданием файлов.

Выполненные примеры (по этому видео и др.) находятся в
репозитории https://github.com/Alexan-7/Studying_Python.org_2021-2022.

Как пошутил один хакер про язык Python: «Этот червь забавен и гибок, но может задушить, если его вовремя не сбросить на землю».

Бонус: генератор паролей

```
import random

while True:

    x = input("Введите имя пользователя, получите
стенерированный пароль и запомните его: ")

    pas = ""

    for y in range(16):

        pas += random.choice("QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm0123456789!@#$%^&*<>")

    print(x, ", your password: ", pas)
```

Программирование на Python

29.01.

python.org - официальный сайт

2022

скачиваем my библиотеку, в которой есть
самый security (прокрутим вниз
на все Download → All releases)

Если bugfix, лучше не ставить
версия Windows: Most recent release -
Скачай → меню download (32x или
64x), Add Python 3.8! to PATH

однажды установлено

Абсолютноическая установка, но
нет поддержки для языка Python

Customize Installation -
See workers → Next → See worker.
как Python - прямо на диск C:

C:\Python38-64

создание текстовый документ →
→ загрузка архив. zip → наже-
тие расширение "txt" на "py"

В Python надо сразу настроить размер шрифта к фунции (options → Configure IDE → Size + Bold → OK)

31.01. Слово "print" надо писать с маленькой буквы.

2022 int(5,9) = 5 - не округляем, а отбрасываем дробную часть.

!= - оператор „Не равно“

С 35:40 иск. Без отступов (Tab) безнаклонная синтаксич. ошибки.
if .. : условие - конец услов оператора

21.02. При помощи условных операторов обозначим нам ног от ошибок
if - elif - else

Модуль подкастов слово import
"https://8ceillets.com" - это промокод опровергания.

модуль демод

gauze While
on 56:40

$x += 1$ - увеличиваем значение 02.02
переменной x на единицу.
2022

$y * count$ - умножаем значение y на $count$.

While True: флаг для бесконечного цикла в Таб (отступы) -
всё в цикле.

`len(x)` - показывает длину
последовательности x

`if ... continue` - прерывается
цикль, когда после `continue` не
исполняется при `if`, но цикл
работает дальше. - при
этой условии цикл не исполн-ся

`if ... break` - полностью прекра-
щает работу цикла, опера-
тор `else` не выполняется.

`\n` - знак переноса строке.

Отступы ТАБ - очень важны -
только в Python! Далее 1:21:24, for

3.02.2022 Число в цикле (cycle-in-cycle)
Функция range генерирует последовательность чисел.

Функции range три параметра:
range (start, stop, step)
range (1, 10, 2) выдаст
1 3 5 7 9 (в стеке)

Программа cycle-in-cycle:

x = 'абв... и тд'

y = input ('введите строку:')

for i in x: # передираем x
count = 0

for r in y: # передираем y,
же ввёз-я строка

Взяв одну букву из x запускается
второй цикл.

За одну сработку первого цикла
подкостю проштрафируется второй цикл.
Во втором цикле сравнивается,

if i == r:
count += 1 # увеличим на 1

len (n) - длина списка n.

Список - это коллекция? тип данных,

и т.д. - т.е. можно извлечь как
изготавливается

Функция **list** конвертирует

строку в список

Файл 1:41:29
Слайд 2

- Выведение текста \rightarrow Alt + 3 - 4.02.
закомментировано в выделенное. 202
- Выведение текста \rightarrow Alt + 4 -
снять комментарий с выделенного.

Когда интерпретатор (F5) активен,
можно посмотреть доступные методы:

переданная строка \rightarrow окно с методами -
то же (как в Delphi).

Синтаксис срезов:

b = n[5:] - от индекса 5 до конца списка

b = n[:5] - от начала списка до индекса 5 1.53

Кортеж - это неизменяемый тип
данных. Как и список, это запись
упорядоченная последовательность элементов

1. Кортежи быстрее итерируются циклами, чем списки;
2. Кортежи заначают меньше места;
3. Кортеж - неизменяемый тип данных (в списке можно изменять значения)

$x = [9, 8, 7, 6, 5, 4, 3]$ далее 2:01:13
 for i in range(len(x)):

$x[i] += 3$ # обработка списка
 print(x) # каждое значение

даём rez-m: [12, 11, 10, 9, 8, 7, 6]

НО! $x = (9, \dots, 3)$ - выдаёт ошибку
 т.к. изменим кортеж нельзя!

У кортежа 2 метода (в отличие от списка, окончно более менее).

Кортеж - неизменяемый, но у него есть
 2 метода ($h = (\dots)$) $h += (\dots)$) далее 2:08:42

`import os` - импортируем модуль по OS.
работе с операционной системой? 02.02.22

`os.walk` - позволяет генерировать путь к файлам на компьютере.

Функция `walk` возвращает кортеж из папок, блокированных в `os.walk ('путь')`
`time.time ()` - время с момента начала эпохи.

Парсер - это программа, сервис или скрапт, который собирает данные с указанной ресурсом, анализирует и возвращает в нужном формате

Функция `def`.

08.02.2022

После создания функции надо ее обозначить и вызвать им-функции ()
Создать ее можно и в начале. Потом вызвать в программе хоть сколько раз.

`return` - оператор, чтобы использовать -
вам значение из функции дальше в программе.

`return` вернет значение в том же
запуска функции.

Выражение `return` прекращает выполнение функции и возвращает указанное после выражения значение.

Далее - Codewars (практика) и 2:30
+ на конец Ютуба по темам Python-анализаторов

09.02. def - определение функции, после него 2022. `имя-функции ():`

В скобках после имени - параметры
При выводе результата надо указать
`print (имя-функция (значение_для_па-
раметра))`

`false` и `False` - разные! Булевое
значение - с большой буквой `False`

11.02 В некоторых случаях функции гд.
2022 универсального, чтобы передать в неё "для обработки" переменные
как - то аргументов.

Спсв. синтаксис: `def name (*args):`
Звёздочка `*` - наз-ся "Астериск"

Она позволяет указывать передаваемые аргументы в кортеже и делить
разделить с ними.

Можно передавать сколько угодно аргументов, когда в программе вызывается ф-я.

Если `def name(*args):`, то помимо при вызове ф-ции `name(1, 2, 3, 4)`, то сколько бы и не было позиционных параметров перед `*args`, они возьмут на себя по одному неиспользованному аргументу, все остальные будут упакованы в кортеж `*args`.

После `*args` - дополнительные параметры для работы. Записываются после `*args`, наз-ся «ключевые параметры»

В вызове функции „к. пар“ г.д. именованы
Список - в квадратных скобках
2.58.2

Кортеж - в круглых скобках

Позиционные параметры - запись са^з ПЕРЕД `*args`

Ключевые параметры - запись са^з ПОСЛЕ `*args`

В вызове функции к ключевым параметрам...
...нужно обратитьсь под имена

Благодаря параметру *args можно передавать в функцию разное кол-во аргументов, а при помощи ^{для обработки} копирования параметров можно сообщать в функцию аргументы с помощью которых управляет её работой.

2:56:28

12.02 Словом Global мы дадим команду, что мы хотим видеть изменения в глобальной переменной, которая будет видна в глобальной области видимости (не создав новую локальную переменную).

14.02 **NONLOCAL** - если мы хотим при помощи где-либо функции не создавать новую локальную для ее' переменную, а изменить значение залежащие переменные в материнской функции.

В Python есть где зорят видимости: глобальная и локальная.

К локальности з. ф. относ - ся переменные, обновляемые в функциях и классах.

Глобальная область видимости: переменные в наследуемом классе While : } гл. ОДн. Выг
чакел : for . : }

Пишнее использование Global или защищенные глобальной областью видимости НЕ приветствуются - нужно устремить, куда вносятесь из нее или из какой функции отдельной - на зоне видимости. далее - 3:08

Передача значений между функции - 16.02 2022

Каждая функция **def** возвращаем **return**

Задача на Codewars (бесплатная):

def multiply(a, b):

c = a * b

return c

далее 3:14:20

Код на Python надо писать так, чтобы чистоё языковое представление работало с глобальными глобальными видимостями.

Чтобы сделать проверку корректности глобальных переменных переделанных,

В конструкции `n = float(input('Введи
гиме радиуса цилиндра в см:')).re
place(',', '.'))` фраза `.replace(',', '.')` подразумевает не только удаление запятой, но только с точкой, но и с запятой

18.02. Словарь похож на список-хранилище с
2022 перечень каких-то объектов, имеющих

свои значения от списка в списке, что в списке все хранятся по индексу. В словаре используется

Словарь записывается в фигурных скобках

Ключ - любой незадекартический тип данных:

- число целое
- число с плавающей точкой
- строка
- кортеж

Словарь { Ключ : значение } 3:25:25

Значение нужно в словаре, когда есть ошибки?

Но! Иногда необходимо обратиться к эл-модулю например, get()

После запуска программы от 18.02.2022
появляется словаря и функции

Видим значения вложенного словаря
словаря.

F5 → Users ['Bob 33'] ['id'] или
Users ['Alex 7'] ['password']

IDLE → Options → Settings → F5

Словарь в Python как строка,
список или кортеж, являющиеся
империрующимися объектами, т.е.
его можно преобразовать
(перебрать) при помощи цикла
`for`.

Найден словарь с записью о цене бокса

Price = {} - втыкаем ошибку

Найден словарь с прописной ценой бокса

Price = {} - исправляем.

round(price, 2) - округляем (2 знака после)

3:41:
40

Метод `x = price.items()`, написанный в интерпретаторе (`F5, 5, x`) создает не список из кортежей, а это только отображение каждого слова в виде huge для возможного его изменения. `type(x) → class 'dict-items'` - Это неподдающееся списку,

Рассказ `list(x)` подсказывает преобразование слова в список, если на него ^{данее} ~~дано~~ метод `.values(x)` в интерпретаторе ^{3:49:51} возвращает представление, но отличное от `items()` будут представлять ^{данее} в huge списка значений слова.

Метод `.keys()` выдает колону слов в виде huge списка. ^{данее} 3:49:21

05.03. Умение, запись кодировка

2022

Код из урока "Парсер файлов" "Parsers" по всему диску C: подключается функция `Walk` (`импорт os`)

... и добавим в список [...]

Считывание информации из файла.

`h = r.read()` - прочитать весь файл
одинаково

`h = r.readline()` - прочитать построк-
но (каждую строку)

`h = r.readlines()` - прочитать построки
но в `h` будет список из всех строк в
файле

Прочитать через цикл - более

простой и удобный способ (`for i in r..`)

Бинарный режим работы - позво-
ляет читать и записывать

бинарные файлы. (кроме текстовых)

Файл может быть 1 бб. и считать его
весь в оперативную память не надо.
помимо с циклом `while` будем записы-
вать его кусками.

Разберёмся с ошибкой на 4:00:00 с. - была ошибка

Метод `__sizeof__()` позволяет по - 12.03.

считывать сколько оперативной

памяти занимает certain объект

`if var.sizeof() == 17:` 17 - это не const,
`break` может отличаться
в зависимости от операт. систем

Кодировка файлов. Когда мы сажаем
считываем и запускаем в файл,
то никаких проблем нет.

Но если файл был создан на другом
компьютере, то может быть потер-
янно загада кодировки.

`r = open('file2', 'w')` - будем записывать
(загадав) 'w' файл, т.к. его нет.

`encoding = 'UTF-8'` - дополнительный
аргумент после 'w' - кодировка файла

Но при чтение из файла надо **ТОЖЕ**
указать кодировку!

14.03. Множества - тип данных

2022

М. создаются при помощи функции `set()`,
куда передается итерированная последовательность
кортеж - строка. Или при помощи фильтра-
ции скобок.

Слово - просто загруженные скобки (р.с.)

Множество - перечисленные значения через
запятую в скобках.

Множество может состоять только
из независимых между собой значений:

- только ЧИСЛО, КОРТЕЖ, СТРОКА

Из списка множество состоит НЕ более

Множество - это неупорядоченная коллекция элементов. Все эти мы **ПЕРЕМЕШАНЫ**

Если добавим в множество одинаковые элементы, после F5 они будут содержать только уникальные значения

Операции над множествами намного быстрее, чем аналогичные операции со списками
(см. код от 15.03.2022 → F5)

Однаковых элементов в множестве быть не может

Метод строк `.split()` вернет список из всех слов. Всёе пробелы

и переносы строк откусывает

`set(r.read().split())` - список превратится во множество без повтор. слов.

Пример с SEO-оптимизацией - в файлах от 2022.03.17.

Использование множеств позволяет быстрее при исключении когда реалистичные поставляются запросы (например, анализ текста). Давно 1:22:30

18.03.
2022

Строки, экранирование.

Двойные кавычки и апострофы
отличаются, когда надо символы
в одинарных апострофах встав-
ть в строку, которая в двой-
ных кавычках.

Число по краям одинарных кавычек а симв-
олы в строке - в двойных.

Если в строке есть апостроф и он разрыв-
ает строку в одинарных кавычках,
то обратный слеш | перед апострофом
подводит экранирование.

Обратный слеш | подводит переход символов
одной строки на следующую. Интерпре-
татор получает одну строку

|n - экранированный символ, переходим
строку в интерпретаторе.

Если экранирование внутри строки, то
перед строкой добавим итеракт r -
запускаем, только где "старт".

В начале строки зажимаем прямой слеш |
Далее h:30

Строки - часть II. Методы.

21.03.
2022

Все знаки в строке проиндексированы от 0 и т.д.

Среди `S[0:5]` берут элементы от 0 - 20 из 5 - 20 (но их 15 штук)

Рассмотренные методы в файле от 21.03. Далее - практика, от 4:41:30

Метод строк `.join()` позволяет объединять слова, разделять ранее разделенные в строке.

Методы форматирования строк есть не для них все (окно с помощью нее `format`). Далее 4:43:43

F-строка. Форматирование строк.

`enter = input('Запрос:')`

`S = 'Text... %s' % enter` - самый старый способ

`S` после `%` - это константа, которую указываем т.к. то, как отформатирована `X`, который подставляется после второго знако процента. (в данном случае подставление строки: `text` везде одинаково).

`S = 'Hello %s, I am %s' % (enter, 'Python')`

`%` - марка, в которую будем подставлять значения (...), т.к. более 1 знака - надо указывать кортежем 4:46:01

4:46:39 $s1 = 'Hello, \{ \}, I am \{ \}'. format(en-$
ter, 'Python') - более новой, но не
очень удобный метод. Раньше
скобки $\{ \}$ брали знаков \$ \$ = мак
буквам MR, что хотели передать
из скобок (цифры 0 и 1 подставлять)
F-строки - самый новый способ
записывать различные строки. (с версии Python 3.6)
 $f'Hello, \{перем-я\}, I can do it in f-$
 $string \{ 2 + 2 \}'$ → переменные подставляются
в строку
→ возникло при фор-
мировании строк.
f-строка быстрее и удобнее.

4:50:10

28.03. Обработка ошибок try-except-finally
2022 Ошибки бывают предсказуемые и не,
которые мы предугадать не можем.
В Delphi обработку try → except →
→ end мы называем Ракетой Мухамед.
Ошибка происходит, но не ошибка -
емка (это я кое-что).

else: - дополнительный необязательный оператор. Используется в том случае, если не было ошибки ИЗНАЧАЛЬНО в блоке try;. Но break после try - неизвестно, прерывает цикл.

Даже если произошла некоторая ошибка, то **finally** всё равно сработает. В finally можно прописать сюда - некие действия, корректирующие ошибку, закрывающие, соединяющие с бло-
гой ошибкой. Далее - 5:00:45

while True:

try:

break - прерывает бесконечный цикл, если выполнено условие try

except Exception - общая группа ошибок математик - k (Type Err, Value Err, File Err or m.g.). + ошибку

Список.append - добавляет информацию в список под её текущий.

Оператор `continue` в Python про-
пускает оставшуюся часть цикла
в блоке цикла и передаёт управле-
ние в его начало, т.е. началу
этого цикла.

После `finally` код ошибки, который
не не прерывает выполнение — закрытие
программы. ~~закрытие~~ — 5.10.00.

код `finally`: исполнение бегла.
но ~~закрытия~~. Рабочий файл сохранен все
равно ~~закрытие~~, и запускается скрипт `fail`.

31.03. With - as: суть применения.

2022. Коммекстный менеджер `With - as` позволяет
автоматически открывать файлы, если
в коде программы ~~пред~~ вспомогатель-
ная

передается с флагом `r = open ('file.txt', 'a')` можно запускать
как `with open('file.txt', 'a') as r:`

~~all~~ mog . close прописывается не отдельно, т.к. оператор with открывает файл а одновременно закрывает его, когда завершится работу с файлом.

Если произошла ошибка, то программа "закрёпится", но этот исключение контекста with закроет файл безотказно. и какие-то другие исключения останутся,

K. d. With - as познакомим [#] наше работать ресурсами требований к закрытию
необходимо закрыть в случае ошибки
Сама ошибку мы не обрабатываем
Можно и через try-finally, но код получается небезразличной и много лишней.

Import модуля



Далее 5.16.



02.04.
2022

Import модулей, if name == "main".
Модули лежат в папке
Python → Lib DLL написаны на языке
C и Python.
Модуль файл с расширением *.py –
это уже модуль.

Date 5:17:31

Принцип кавычек "" в Python:

1. Внутри них можно использовать одинарные ' и двойные " кавычки;
2. Можно разделять строку на несколько строк;
3. Стандартный уголок практикой придерживается десятичного разделителя.

Как фигуранты скобки {} или константные & фарши {...} в Delphi

Чтобы узкакие, что есть в модуле, который импортируется, надо передавать в функцию dir() имя модуля.

print(dir(имя_модуля))

print(help(имя_модуля)) - справка по модулю

Посмотреть справку по функции - можно
далее закомментировать и вызывать после
Import ини-модуль → Ини-модуль.функция.

Можно использовать - не конкретные имена-модулей
из модуля!
`from модуль import *` - import-ся
все имена, какие там есть. И они все
находятся в той области видимости, с ко-
торой работает. Далее 5.23.30

И можно обращаться к этим именам,
а не указывать явно имена модулей.

Что можно вызвать функции и тд и тп
нужно указывать явно имена модулей, а спра-
шивая есть ли в работе программы.

Можно кроме всего import * использовать-
работки отдельные фунекции, например:

`from ини-модуль import newf`

Можно дать именование модулю, если это
имя отличное:

`import ини-модуль as m`
и дальше использовать короткое имя.

Модулей очень много. Выбрать модуль
под опред-е нужды и узнать его функции
также есть присторы,
лучше через поисковик, или
проще поиски dir и help. Лучше си.
в Google и Яндекс. Далее - 5.29.50

07.04. if -name- == 'main'
2022 Когда мы пишем код (модуль), мы
не догреваемся, что он может исполь-
зоваться самостоимельно, т.е. сам
по себе представляет како-[?] -то
функционал. И он может быть им-
портирован куда-то в другой модуль.

Когда импортируется модуль, Python его
сразу читает и запускает исполнением.
Есть такой параметр, как -name-.
Когда мы запускаем модуль как само-
стоеимельную программу Python присва-
ивает в этот параметр value -main-
(можно воспользоваться print (-name-))

Если мы импортируем этот модуль и
там есть такой запрос if -name- == 'main',
то его код исполнен не будем. Он
просто не выполнится, потому что его испол-
няет не main. Когда импортируется
то в параметр name присваивается
просто имя модуля. Проверка
не срабатывает и код не выполн-ся.

Можно проверить: в программе, когда импортируешь, записать после `import`
`print(модуль.__name__)`

импортировав, использовать его функции.

Игра в досиню. на tkinter

дате - 5.3н.58

08.04

2022

Tkinter - кроссплатформенное

библиотека (модуль) для разработки графического интерфейса на языке Python.

`from tkinter import *` - импортирует все
`root = Tk()` - определяет главное окно,
`root.mainloop()` - запускает цикл на-
шего окна

При получении метода `root.iconphoto` можем разместить в кадре саму иконку изображение `*.png`, а не `*.ico`.

Вызываем класс `Label` из Tkinter

указывая `.pack()` размещаем любой элемент `label` на окне.

`lab1` и `lab2` - создают где мен-
ю, в которых будут отображаться се-
кции икошки

Можно создать меню просто так
(как в Label (root, image = font)).

1. pack(). Можно задать ее через
переменную (Lab1 = Label (root)),
если дальше по программе к ней будем
обращение.

Метод .place позволяет нам точно расположить на окне метку.

Метод .place размещаем по схемке
 $y \in [0;1]$ и $x \in [0;1]$ в верхней четверти.

PhotoImage - класс открытия картинки.
PhotoImage (file = (bros ())) - передаёт
в параметр "file" ранее созданную
фотографию "Бросок" через random.

Обращается к меткам (ранее
созданным) Lab1 и Lab2 в функции
def img ().

Все атрибуты метода хранятся
в словаре. Поэтому параметру Image
присваивается ранее созданное изображение.

`root.update()` - обновление окна.

Вместо кнопки писалась обработка -
так, событие `root.bind('<1>', img)` - реакция на какое-либо событие,
здесь, где '`<1>`' - событие Клик мыши.
и перед запуском `img` - вызываемая функция.

Картинки пока не отображаются,
т.к. окно их не видит из-за функ-
ции `def img(event):`. Поэтому
записал `global b1, b2`, чтобы не-
рассматривать не было сокрытое.

В функции `def img(event):` вто-
гий цикл `for i in range(10)`, чтобы
картинки передвигались после каждого
клика мыши (но 10 раз).

На youtube надо ставить качес-
тво видео 1080p HD

Но когда запускаешь программу - есть две
пустые метки. Чтобы сразу было передви-
жение.

Потому запускаем функцию `img('event')`
в конце программы (с абсолютным назначе-
нием, чтобы ее было видно вперед).

`pack()` - метод для позиционирования
элементов в окне.

Для перевода из *.py в *.exe уста-
навливается pyinstaller. Для этого

В командной строке пишем
`pip install pyinstaller`

Проверяется версия pyins. Пишем:
`pyinstaller --version`

Далее в командной строке пишем:

`... > pyinstaller путь\файл.py`

В папке C:\Users\Brdgn у меня есть проект (созданась) папка `build`,
папка `dist`, файл `*.spec`.

Можно удалить `build`, `-pcache` и
`*.spec`. В папке `dist` лежит папка
с файлами нашей программы. Куда бы
туда картинки из программы.

Надо добавить `-W` перед путём файла,
чтобы не было командной строки (флаг)

`... > pyinstaller -W путь\файл.py`
далее 5.56.53. Декораторы

13.04.
2022.

@ Декораторы.

Декоратор - это функция, кото-
рая подгружает содержимое другого
файла и вставляет его в свой код, т.е. синтакси-

модифицировав её поведение.

Декоратор позволяет нам модифицировать код функции так, как нам нужно, не вынимаясь в её код.

Когда работаем над своим небольшим проектом, декораторы НЕ нужны, чтобы это исправить.

НО в масштабном групповом проекте может код нельзя и поправить функцию можно через ДЕКОРАТОРЫ.

Exception - с заглавной буквой

Except Exception - общая группа ошибок.

Декораторы также позволяют избежать дублирования одного и того же кода внутри каждой функции.

Перед функцией пишется @decor. Это значит функция = decor(функция) аргумент

Декоратор позволяет запустить функцию внутри другой функции, сохранив обещанное в ней.

далее - 6.10 - генера

Генераторы списков, словарей, множеств.

25.04
2022

Когда нужно создать список на основе другого списка, обычно запускаем цикл for

НО вместо этого есть синтаксис генератора списков

$h = [9, 8, \dots, 3, 2, 1]$

`newh2 = [x * 2 for x in h]`
`print(newh2)`

Множество h можно передавать не
без аптератора именуя последнюю:
строка, список, кортеж, словарь

Множество не содержит дублиру-
ющих элементов (повторений)

В генераторе можно записать условие.

`g = [x for x in h if x % 2 == 0]`

Получаемся генератор проходящий по спис-
ку `for x in h` с только значениями,
которые соответствуют условию
`if x % 2 == 0` будем обрабатывать
все `[x...]` далее - 6.20.10

Если запись получается длинной то можно
и то перенести на другую строку.

Синтаксис генераторов позволяет
писать НЕ в одну строку

`g = [os.path.join(x, i)`
`for x, i in os.walk('C:\\"')`
`for i in c if '.txt' in i`

Такой способ (через генератор) более компактный и работает быстрее чем код из урока про парсер файлов.

Генератор словаря на основе - 28.
или другого словаря 04.

Заводские перевесы на new_d. 11b 2022
ригурных скобках...

Надо запасать килог : значение.

Объяснение в файле 2022.04.28_Генераторы

Получим один результат при получении двух разных костей.

Но через генератор - быстрее и записей в одну строку.

Ген-ры списков - в ~~квадратных~~ скобках.

Ген-ры множеств - в фигурных скобках

Ген-ры словарей - в фигурных скобках,
но кило : значение

Выражение - генератор - это другое. ganz

Выражение - генератор - это 01.05. 6:30
така оптимизация программы. 2022

Метод .split возвращает список из
строк, разделенных по разделяителю

(скобки после split).

В выражение - генератор записывается в круглых скобках, 6:34:40

Функция eval() в Python используетась для симуляции выполнения кода из строки - выражения, как выражения Python и последующего его выполнения.

Без функции eval() мы НЕ можем выполнить команды, выделяемые подзаголовком. Стока функции eval() заканчивается в дальнейшем выполнении операторов

Выражение - генератор будет обработано по одному элементу из первого списка за одно выражение к нему.

Если итерировать генератор списка с циклом for, то генератор списка скажет поочередно создаст новый список, загрузив его в память, после этого вернется пакетом for.

Выражение - генератор будет генерировать по одному результату и забывает его за одно выражение.

`n[1]` - возврат значение по индексу
у списка - можно

`z[1]` - по индексу у бордс.-генератора-
Нельзя error, так как у генера-
тора нет индексов.

`help(next)` в интерпретаторе. Возвращаем по одному значению из объекта,
который можно проконтролировать.

Ген-р берёт по 1-му значению и это
кончит наше паслов.

далее - 6:41:53
В интерпретаторе написали:

`n.sizeof()` — } указать, сколько заня-
`z.sizeof()` — } иает памяти пере-

`next(z)` - каждый раз обращается
к бордсии - генератору.

`next(z)` - возврат по одному элемен-
ту запр. Экономится паслов.

6:43:03

06.05
2022

Следующий - генератор.

Оператор `yield`.

Установили стандартный редактор кода
`VS Code`. Ищем в поиске, где -
на сайте `code.visualstudio.com`

Узнать разработчики самое:

мой компьютер \rightarrow свойства \rightarrow так система
мог ($32x$ или $64x$).

Галочка добавить в PATH - обязательно
поставить.

Оператор **yield** даёт понять интерпретатору Python, что данная функция, когда
вызоврашает какое-то значение. Из этой функции создается
объект-генератор при помощи **yield** данс - 6:49:38

Использование генераторов облегчает работу программы, инициализируя её подбирает + получив ка-
кое-то значение, сразу обрабатывает его.

Lambda - функция

11.05

2022

Запись lambda аналогична записи def some ():

Параметры через пробел
(у def они в скобках)

После параметра - возвращение.
Оператор return не нужен -
он стоит по фальстаки.

После возвращения : пишется выражение, которое используется в результате вернёт lambda-функцию
Позволяет в одну строку до 79 символов записать.

Синтаксис lambda'и позволяет передавать одну логическую единицу на две функции.

Чтобы запустить lambda-функцию запишите её в некое переменное **Var = lambda x: x/5** и далее запустить через переменную саму функцию.

from dis import dis - из модуля dis импортируется функция dis, которая показывает, как в байт-коде будет че-

терпретатору заключено функцию

Нужно правильное применение функции `lambda`, чтобы она могла участвовать в контексте нашего кода.
(а не посторонне). Далее - 6:58:12

`lambda` очень удобно использовать в контексте функции - на фильтр и сорт.

Функция `sorted` вернёт нам новый отсортированный список.

`r = sorted (сортиментный_список, key =)`
`key = - кий сортировки.`

Функция `filter` использует, когда нужно на основе списка создать новый список, используя какую-то залогу.

`x = filter (lambda x: x[1] > 18,
list_of) - рассмотрим подробнее.`
`list_of - список, который будем фильтровать.`

Lambda: - возвращаем True или False.

Если True, то элемент списка добавляется в новый список X.

Вместо определения новой функции кратко записьем с lambda.

X: - выделим из списка.

X[1]>18 - условие, которое г.д. True, можно добавить в список.

Фн.к. фильтр - это объект, то добавляя из него список при помощи list.

X = list(filter(...))

С 29.01.2022 по 12.05.2022

Данее - codewars и канал youtube

Python Hub Studio

Генератор паролей

16.05.
2022

import random - импортируем
random - получаем случайн. выбора.

Сам файл random в папке lib- содержит 815 строк кода!

А сам скрипт генератора паролей? со-
держит 11 строк.

В VS Code можно нажимая
клавиши Python - все. чтобы Ex-
tension (Ctrl + Shift + X) **закрыть**

Ctrl = u Ctrl - разбиваем
уменьшаем или увеличиваем Zoom.

Python DDT

Слабый UI. - Число сложенное с
загрузкой.

Начальное - обучение - кака
о more, как засадить
или вырастить и действовать
самостоятельно без прокоманды
человека.

На каких языках можно это?

1. Минимал препарсинг -
позволит спросить что будущее

2. Максимал Байеса - позво-
лить отвечать на подтиповые
вопросы, основываясь на
предикции знатоков