Занятие 9

Вложенные словари

Работа с текстовыми файлами

Что напечатает?

```
print(sorted([1, -5, 2, -4, 3, float('inf')], key = lambda x: -abs(x)))

print(sorted([1, 2, 3, 111, 222, 333], key = lambda x: len(str(x))))

print(sorted([1, 0, [], [0], (), (0,), (0)], key = lambda x: 1 - bool(x)))

print(sorted([‘Hi', ‘These', 'Craziest', 'Worlds’], key = lambda x: len(x)))

print(sorted(['Hello', 'This', 'Crazy', 'World'], key = lambda x: x[::-1]))
```

Задача 8-1

На вход программе подается строка генетического кода, состоящая из букв А (аденин), Г (гуанин), Ц (цитозин), Т (тимин).
Подкорректируйте код.
Если рядом стоят А и Г, то поменяйте их местами.
Если рядом стоят Ц и Т, то поместите АГ между ними.

Задача 8-2

На вход подается список, состоящий из списков чисел, например:

lst = [[1,5,3], [2,44,1, 4], [3,3]]

Отсортируйте этот список по возрастанию общего количества цифр в каждом списочке.

Каждый списочек отсортируйте по убыванию.

Такие словари называются вложенными. И для указания элементов необходимо указывать несколько индексов, например:

lst[0][0] = 1, lst[0][1] = 5, lst[1][0] = 2, lst[1][3] = 4

Что напечатает print(lst[1][1])?

Задача 8-3

Дан список слов. Отсортируйте его по количеству уникальных букв в каждом слове в обратном порядке.

Например: [‘abab’, ‘xx’, ‘aaaaaaa’, ‘abcbab’].

Результат: [‘abcbab’, ‘abab’, ‘aaaaaaa’, ‘xx’]

Если число уникальных букв одинаково, то порядок алфавитный.

Lambda

```
spisok = [222, 31, 1, 711, 82, 322]

# Как отсортировать список по первой цифре каждого элемента?

print(sorted(spisok, key = lambda x: ??????))

# Найти число из списка, у которого сумма цифр наименьшая

print(min(spisok, key = lambda x:  ????? ))
```

Lambda в sort, sorted, max, min

- max(lst, key = abs)
- max(lst, key = lambda x: abs(x))

Но можно и более сложные функции:

- Отсортировать список целых чисел по возрастанию, но сначала четные числа, а потом нечетные.
- sorted(lst, key = lambda x: (x%2, x))

- Что надо изменить, чтобы сначала были нечетные числа, а потом четные?

- Отсортируйте список слов не зависимо от регистра, например:

Вход: [‘b’, ‘A’, ‘Z’, ‘x’] Выход: [‘A’, ‘b’, ‘x’, ‘Z’]

Тернарный оператор

- x = 1
- y = 2
- maximum = x if **x > y** else y

Например, можно так, но очень громоздко:

```
def abs(number):
    if number >= 0:
        return number
    return –number
```

А можно так, более лаконично:

```
def abs(number):
    return number if number >= 0 else -number
```

Можно ли в лямбда-выражениях использовать
стандартные операторы управления if, for, while?

НЕТ!

НО !

Можно использовать тернарный оператор.

```
lower = (lambda x, y:    x if x < y else y)
# Вызов 1 способ
(lambda x, y:     x if x < y else y)(10,3)

# Вызов 2 способ
lower(10,3)
3
```

Задание

Создайте лямбда функцию, которая рассчитывает подоходный налог в зависимости от суммы дохода.

Если доход не больше 5 000 000, то 13%.

Если доход больше 5 000 000, то 13% от 5000000 и 15% с суммы превышающей 5 000 000

Задание

Дан список из кортежей (Фамилия, премия).

Напечатать эти кортежи в порядке убывания премии.

Тех, у кого одинаковая премия, то печатать в алфавитном порядке фамилий.

Например:

Ввод: [(Иванов, 100), (Петров, 200), (Сидоров, 200), (Воробьев, 100), (Лунин, 200)]

Результат:

Лунин 200

Петров 200

Сидоров 200

Воробьев 100

Иванов 100

Методы словарей

- **dict.clear**() - очищает словарь.
- **dict.copy**() - возвращает копию словаря.
- classmethod **dict.fromkeys**(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None).
- **dict.get**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).
- **dict.items**() - возвращает пары (ключ, значение).
- **dict.keys**() - возвращает ключи в словаре.
- **dict.values**() - возвращает значения в словаре.
- **dict.pop**(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).
- **dict.popitem**() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.
- **dict.setdefault**(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ со значением default (по умолчанию None).
- **dict.update**([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).

Вложенные словари

Ключами (keys) словаря могут быть только неизменяемые объекты: числа, строки, кортежи, True, False и некоторые другие.

{True:1, False:2, 1:2, '1':2, frozenset((1,2)):[1,2]}

Значениями (values) может быть все, что угодно: числа, строки, кортежи, True, False, а также: списки, множества, функции, лямбда функции…

a = {True:abs, False:(lambda x:x * x), 3:{1,2,3}, '1':[1,2], 'Город':'Санкт-Петербург’}

Что напечатает?

a[True](-123)

a[False](5)

А могут быть и словари!!!

```
students = { 0: {‘name’: ‘Иванов’, ‘age’: 22},
             1: {‘name’: ‘Петров’, ‘age’: 23},
             2: {‘name’: ‘Сидоров’, ‘age’:24}}
```

Что напечатает students[0]?

students[1]?

students[0][‘name’]?

students[1][‘age’]?

Для обращения к элементам внутренних словарей нужны 2 ключа!

Напечатайте ключи и значения всех словарей (словарь имеет два уровня вложенности)

Задание

Дан словарь dct с не более, чем двумя уровнями вложенности.

Введите ключ x и напечатайте значения всех словарей, у которых ключ совпадает с x.

Например:

dct = {1: 123, 2:234, 3:{1:111, 2:222}, 4:{1:’abc’, 2: ‘def’}}

x = 1

Результат: 123 111 abc

Подсказка: функция type() определяет тип аргумента: int, str, list, dict и др.

import collections

collections.Counter - вид словаря, который позволяет нам считать количество неизменяемых объектов (в большинстве случаев, строк).

collections.deque(iterable, [maxlen]) - создаёт очередь из итерируемого объекта с максимальной длиной maxlen. Очереди очень похожи на списки, за исключением того, что добавлять и удалять элементы можно либо справа, либо слева.

collections.defaultdict ничем не отличается от обычного словаря за исключением того, что по умолчанию всегда вызывается функция, возвращающая значение

collections.OrderedDict - ещё один похожий на словарь объект, но он помнит порядок, в котором ему были даны ключи (после версии 3.6 любой словарь помнит этот порядок)

collections.namedtuple позволяет создать тип данных, ведущий себя как кортеж, с тем дополнением, что каждому элементу присваивается имя, по которому можно в дальнейшем получать доступ

collections.Counter

```
import collections
a = collections.Counter('aabbbcccddddeeeeeabcdef')
b = dict(a)
print(b)
{'a': 3, 'b': 4, 'c': 5, 'd': 5, 'e': 5, 'f': 1}
```

x = “С помощью Counter определите, какая буква содержится больше всего в строке x”

Работа с файлами

TXT

.txt — это формат файлов, который содержит текст, упорядоченный по строкам.

Текстовые файлы отличаются от двоичных файлов, содержащих данные, не предназначенные для интерпретирования в качестве текста (закодированный звук или изображение).

.py – это тоже текстовые файлы)

Что мы можем делать с файлом?

- Открыть
- Прочитать
- Дописать
- Переписать
- Закрыть!!!

Открытие файла

Прежде, чем работать с файлом, его надо открыть.

Для этой задачи есть встроенная функция open:

```
f = open("test.txt",  encoding="utf-8")
```

Результатом работы функция open возвращает специальный объект, который позволяет работать с файлом (файловый дескриптор)

Создайте в PyCharm текстовый файл test.txt, введите туда 4-5 строк:

First string

Second string

Третья строка

Четвертая строка

Синтаксис функции open()

fp = open(**file, mode='r',** buffering=-1, **encoding**=None, errors=None, newline=None, closefd=True, opener=None)

Параметры:

- **file** - абсолютное или относительное значение пути к файлу или файловый дескриптор открываемого файла.
- **mode** - необязательно, строка, которая указывает режим, в котором открывается файл. По умолчанию 'r'.
- **buffering** - необязательно, целое число, используемое для установки политики буферизации.
- **encoding** - необязательно, кодировка, используемая для декодирования или кодирования файла.
- **errors** - необязательно, строка, которая указывает, как должны обрабатываться ошибки кодирования и декодирования. Не используется в бинарном режиме
- **newline** - необязательно, режим перевода строк. Варианты: None, '\n', '\r' и '\r\n'. Следует использовать только для текстовых файлов.
- **closefd** - необязательно, bool, флаг закрытия файлового дескриптора.
- **opener** - необязательно, пользовательский объект, возвращающий открытый дескриптор файла.

Имя файла, какой файл, что делать

У функции **open()** много параметров, нам пока важны 3 аргумента:

Первый, это имя файла.

Путь к файлу может быть относительным или абсолютным.

Второй аргумент - это режим, mode, в котором мы будем открывать файл. Режим обычно состоит из двух букв, первой является тип файла - текстовый или бинарный, в котором мы хотим открыть файл, а второй указывает, что именно мы хотим сделать с файлом.

Третий аргумент – кодировка файла

Первая буква режима:

"b" - открытие в двоичном режиме.
"t" - открытие в текстовом режиме (является значением по умолчанию).

Второй буква режима:

"r" - открытие на чтение (является значением по умолчанию).
"w" - открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
"x" - эксклюзивное создание (открытие на запись), бросается исключение FileExistsError, если файл уже существует.
"a" - открытие на дозапись, информация добавляется в конец файла.
"+" - открытие на чтение и запись

Примеры

Режим "w" открывает файл только для записи.

Перезаписывает файл, если файл существует.

Если файл не существует, создает новый файл для записи.

f = open("test.txt", mode="w" **encoding=**"utf-8")

Открывает файл в бинарном режиме для записи и чтения.

Перезаписывает существующий файл, если файл существует.

Если файл не существует, создается новый файл для чтения и записи.

f = open("music.mp3", mode="wb+")

По всем режимам см. документацию open()

Закрыть файл

После того как вы сделали всю необходимую работу с файлом - его следует закрыть.

```
f = open("text.txt",  encoding="utf-8")
# какие-то действия
f.close()
```

Чтение файла

Теперь мы хотим прочитать из него информацию.

Для этого есть несколько способов, но большого интереса заслуживают лишь два из них.

Первый - метод **read,** читающий весь файл целиком, если был вызван без аргументов, и n символов, если был вызван с аргументом (целым числом n).

```
f = open("test.txt", "r")

print(f.read(5))
print(f.read(5))
print(f.read(4))
print(f.read())

f.close()
```

Фунция readlines()

Файлы можно читать не только целиком или посимвольно, но и построчно.

Для этого у объекта файла есть метод readlines, который возвращает список из строк файла.

```
f = open("test.txt", "rt")
print(f.readlines())
f.close()
```

Обратите внимания, что каждая строка в списке имеет в конце символ `\n`.

Задание

Прочитайте содержимое файла с помощью функции readlines()

Присвойте ее результат переменной lst

Напечатайте пронумерованный список строк.

Постарайтесь избавиться от лишних пустых строк.

Фунция readline()

Функция readlines() загружает все строки целиком и хранит их в оперативной памяти,
что может быть очень накладно, если файл занимает много места на жёстком диске.
Можно читать файл построчно с помощью функции readline()

```
f = open("text.txt", "rt")
print(f.readline())
print(f.readline())
f.close()
```

Также обратите внимание, что возвращённые строки имеют в конце символ `\n`.

Задание

Прочитайте содержимое файла с помощью функции readlines()

Присвойте результат переменной lst

Напечатайте пронумерованный список строк.

Постарайтесь избавиться от лишних пустых строк.

Итерирование файла

Ещё один способ прочитать файл построчно - использовать файл как итератор. Такой вариант считается самым оптимизированным

```
f = open("text.txt")
for line in f:
    print(line)

f.close()
```

Запись

Теперь рассмотрим запись в файл.

Для того чтобы можно было записывать информацию в файл, нужно открыть файл в режиме записи.

Для записи в файл используется функция write.

При открытии файла на запись из него полностью удаляется предыдущая информация.

```
fout = open("test.txt", "wt")
fout.write("New string")
fout.write("Another string")
fout.close()
```

Если вы откроете файл в текстовом редакторе, то увидите, что строки "New string" и "Another string" склеились.

Так произошло, потому что между ними нет символа перевода строки.

writelines()

Также в файлах, открытых на запись, есть метод writelines, который позволяет записать несколько строк в файл

```
f = open("text.txt", "wt")
lines = [
    "New string\n",
    "Another string\n",
]
f.writelines(lines)
f.close()
```

Задание

Прочитать информацию из файла test.txt.

Записать в файл test1.txt, только те строки, которые содержат цифры.

Например:

Hello!

This is the 1st letter.

Bye

Результат: This is the 1st letter.

Задание

Откройте текстовый файл.

Каждый второй знак этого файла перенесите в другой файл.

Задание

Прочитать строки текста из одного файла,
отсортировать слова внутри строки по возрастанию и
записать обновленные строки в другой файл.

Задача 9-1

Дан генетический код ДНК (строка, состоящая из букв G, C, T, A)

Постройте РНК, используя принцип замены букв:

- G → C;
- C → G;
- T → A;
- A → T

Напишите функцию, которая на вход получает ДНК, и возвращает РНК, для этого постройте словарь для замены букв.

Например:

Вход:GGCTAA Результат: CCGATT

Задача 9-2

Напишите программу, которая определяет и печатает «похожие» слова.

Слово называется **похожим** на другое слово, если его гласные буквы находятся там же, где находятся гласные буквы другого слова, например: дорога и пароход - похожие слова (гласные буквы на втором, четвертом и шестом местах), станок и прыжок - похожие слова, питон и удав непохожие слова. Считаем, что в русском языке 10 гласных букв (а, у, о, ы, и, э, я, ю, ё, е).

Ввод: x –первое слово, например, питон.

n – количество слов для сравнения, например 6.

Дальше вводятся 6 слов, например: поросенок, титан, итог, лавка, погост, кино.

Вывод - слова, **похожие** на питон:

титан, погост, кино

Задание 9-3

Произвести частотный анализ текста из файла.

Сосчитать с помощью словаря и функции get сколько раз встречается каждый символ в тексте (включая буквы, цифры и служебные символы, включая пробелы), не учитывая регистр.

Отсортировать по убыванию и напечатать первые 10 символов, и их частоты. При равенстве частот отсортировать символы в алфавитном порядке

Например, текст «Мама мыла раму»:

а – 4

м – 4

л – 1

И т.д.