Занятие 9

Лямбда-функции

Вложенные словари

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([‘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 (или Г и А), то поменяйте их местами.
Если рядом стоят Ц и Т (или Т и Ц), 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.

Функции для работы со строками

- **str**(n) — преобразование числового или другого типа к строке;
- **len**(s) — длина строки;
- **chr**(s) — получение символа по его коду ASCII;
- **ord**(s) — получение кода ASCII по символу;
- **find**(s, start, end) — возвращает индекс первого вхождения подстроки в s или -1 при отсутствии. Поиск идет в границах от start до end;
- **rfind**(s, start, end) — аналогично, но возвращает индекс последнего вхождения;
- **replace**(s, new) — меняет последовательность символов s на новую подстроку new;
- **split**(x) — разбивает строку на подстроки при помощи выбранного разделителя x;
- **join**(x) — соединяет строки в одну при помощи выбранного разделителя x;
- **strip**(s) — убирает символы с обеих сторон;
- **lstrip**(s), **rstrip(s)** — убирает символы только слева или справа

Функции для работы со строками

- **lower**() — перевод всех символов в нижний регистр;
- **upper**() — перевод всех символов в верхний регистр;
- **capitalize**() — перевод первой буквы в верхний регистр, остальных — в нижний.
- **isdigit**() - состоит ли строка из цифр
- **isalpha**() - состоит ли строка из букв
- **isalnum**() - состоит ли строка из цифр или букв

index(s, start, end)

- Метод выдает индекс первого вхождения.
- txt = "Hello, welcome to my world."
- x = txt.index("welcome")
- print(x)

- # В отличии от find выдаст ошибку
- txt = "Hello, welcome to my world."
- x = txt.index("goodbay")
- print(x)
- **ValueError:** substring not found

replace(oldvalue, newvalue, count)

- Параметры:
- oldvalue - строка для поиска
- newvalue – строка замены
- count – сколько вхождений заменить, по умолчанию = все

Что напечатает?

- txt = "I like bananas"
- x = txt**.replace**("bananas", "apples")
- print(x)

- txt = "I like bananas"
- x = txt.replace("a", "o", 2)
- print(x)

join(iterable)

Что напечатает?

- myTuple = ("John", "Peter", "Vicky")
- x = "#".join(myTuple)
- print(x)

- myDict = {"name": "John", "country": "Norway"}
- mySeparator = "_"
- x = mySeparator.join(myDict)
- 'name_country’

print('-'.join('abc-xyz-fgh'.split('-’)))

strip(characters)

Параметры:

Characters – опциональный, устанавливает символы для удаления из текста

```
# удаление пробелов
>>> text = "  test "
>>> text.strip()
'test'


txt = ",,,,,rrttgg.....banana....rrr"
x = txt.strip(",.grt")
print(x)
banana
```

lstrip(characters) **rstrip**(characters)

Параметры:

characters – опциональный, устанавливает символы для удаления из текста

удаление символов ‘.’ слева

>>> text = "...test..."

>>> text.lstrip(".")

'test...'

>>> text = "...test..."

>>> text.rstrip(".")

'...test'

Таблица "Функции и методы строк"

Функция или метод	Назначение
S = r"C:\temp\new"	Неформатированные строки (подавляют экранирование)
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
str.**partition**(*sep*) 'abdegh'.partition('de')	Делит строку на три части, до sep, sep и после sep. ('ab', 'de', 'gh')

Функция или метод	Назначение
S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Начинается ли строка S с шаблона str
S.endswith(str)	Заканчивается ли строка S шаблоном str
S.join(список)	Сборка строки из списка с разделителем S
ord(символ)	Символ в его код ASCII
chr(число)	Код ASCII в символ
S.capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
S.center(width, [fill])	Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию)
S.count(str, [start],[end])	Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)

Лямбда Функции

Лямбда-функции, анонимные функции

Раньше мы использовали функции, обязательно связывая их с каким-то именем.

В Python есть возможность создания однострочных анонимных функций

Конструкция:

lambda [param1, param2, ..]: [выражение]

lambda - функция, возвращает свое значение в том месте, в котором вы его объявляете.

Пример

```
def fun1(x):
    return x % 10

spi = [222, 21, 1, 111, 12, 322]

print(sorted(spi, key = fun1))

print(sorted(spi, key = lambda x: x % 10))
```

Вызов lambda

Для вызова lambda обернем функцию и ее аргумент в круглые скобки. Передадим функции аргумент.

```
>>> (lambda x: x + 1)(2)
3
```

Задание

Используя лямбда-функцию напишите оператор в одну строку, который печатает квадраты чисел от 0 до 9.

Подсказка: использовать map, в качестве первого параметра написать лямбда функцию, которая возвращает квадрат числа, в качестве второго параметра range(10)

Именование lambda

Поскольку лямбда-функция является выражением, оно может быть именовано.

Поэтому вы можете написать код следующим образом:

```
>>> add_one = lambda x: x + 1
>>> add_one(2)
3
```

Аргументы

Как и обычный объект функции, определенный с помощью def, лямбда поддерживают все различные способы передачи аргументов.

Это включает:

Позиционные аргументы

Именованные аргументы (иногда называемые ключевыми аргументами)

Переменный список аргументов (часто называемый *args)

Переменный список аргументов ключевых слов **kwargs

Аргументы функции

Функции с несколькими аргументами (функции, которые принимают более одного аргумента) выражаются в лямбда-выражениях Python, перечисляя аргументы и разделяя их запятой (,), но не заключая их в круглые скобки:

```
>>> full_name = lambda first, last: f'Full name: {first.title()}{last.title()}'

>>> full_name('guido', 'van rossum')
'Full name: Guido Van Rossum'
```

Задание

Создайте лямбда функцию, которая принимает один параметр – строку.

Переводит все буквы в нижний регистр и переворачивает их в обратном порядке.

Пример:

Вход: ‘ACbdzYx’

Результат: ‘xyzdbca’

Именованные параметры

Как и в случае **def**, для аргуменов **lambda** можно указывать стандартные значения.

```
>>>strng = ( lambda a='He', b='ll' , c='o': a+b+c)
strng(a='Ze')
'Zello'
```

Для чего используется lambda ?

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: (??????))

Отсортируйте список слов не зависимо от регистра, например:

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

Задание

Дан список чисел lst и число p. Найти и напечатать самое близкий

к числу p элемент списка lst.

Например: lst = [1, 10, 21, 30]

Наиболее близкое к числу 16 является 21:

16 – 1 = 15, 16 – 10 = 6, 21 – 16 = 5, 30 – 16 = 14

Какую лямбда-функцию лучше всего здесь использовать в операторе min()?

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

Лямбда-выражения как элементы кортежей

Формируется кортеж, в котором элементы умножаются на разные числа.

```
import random
# Кортеж, в котором формируются три литерала-строки
# с помощью лямбда-выражения
T = ( lambda x: x*2,
    lambda x: x*3,
    lambda x: x*4 )
# Вывести результат для строки 'abc'
for t in T:
    print(t('abc'))
```

Результат:

```
abcabc
abcabcabc
abcabcabcabc
```

Использование лямбда-выражения для формирования словаря функций

```
# Словарь, который есть таблица переходов
Dict = {
    1 : (lambda: print('Monday')),
    2 : (lambda: print('Tuesday')),
    3 : (lambda: print('Wednesday')),
    4 : (lambda: print('Thursday')),
    5 : (lambda: print('Friday')),
    6 : (lambda: print('Saturday')),
    7 : (lambda: print('Sunday'))
    }

# Вызвать лямбда-выражение, выводящее название вторника
Dict[2]() # Tuesday
```

Совместное использование lambda-функции со встроенными функциями

Функция **filter()** принимает два параметра — функцию и список для обработки.

В примере мы применим функцию list(), чтобы преобразовать объект filter в список.

```
numbers=[0,1,2,3,4,5,6,7,8,9,10]
list(filter(lambda x:x%3==0,numbers))
[0, 3, 6, 9]
```

Код берет список numbers, и отфильтровывает все элементы из него, которые не делятся нацело на 3.

При этом фильтрация никак не изменяет изначальный список.

Функция map()

Функция **map()** в отличие от функции **filter()** возвращает значение выражения для каждого элемента в списке.

#Пример 1

numbers=[0,1,2,3,4,5,6,7,8,9,10]

list(**map(**lambda x:x%3==0,numbers)**)**

[True, False, False, True, False, False, True, False, False, True, False]

#Пример 2

list(map(**lambda** x: x.capitalize(), ['cat', 'dog', 'cow']))

['Cat', 'Dog', 'Cow']

Задание

Дан список чисел.

Превратить его в список сумм цифр каждого числа.

Например.

Вход: lst = [123, 234, 345, 456]

Результат: [6, 9, 12, 15]

Используйте list(map(lambda x: ?????, lst))

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:  ????? ))
```

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.

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

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()
```

Фунция read()

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", "r")
print(f.read())
f.close()
```

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

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", "r")
print(f.readlines())
f.close()
```

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

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()
```

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.

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.