Занятие 8

str

lambda

Начнем с ...

- 1. Разминка (Что напечатает?)
- 2. Обсуждение домашнего задания

Что напечатает?

```
def fu1(x):    return abs(x)
print(sorted([1, -5, 2, -4, 3, float('inf')], key = fu1))

def fu2(x):    return str(x)
print(sorted([1, 2, 3, 111, 222, 333], key = fu2))

def fu3(x):    return bool(x)
print(sorted([1, 0, [], [0], (), (0,), (0)], key = fu3))

print(sorted(['Hello', 'This', 'Crazy', 'World']))

def fu4(x):    return x[::-1]
print(sorted(['Hello', 'This', 'Crazy', 'World'], key = fu4))
```

Scope (область видимости)

- **Область видимости** указывает интерпретатору, когда наименование

(или переменная) видима. Другими словами, область видимости определяет, когда и где вы можете использовать свои переменные, функции и т.д.

- Если вы попытаетесь использовать что-либо, что не является в вашей области видимости, вы получите ошибку NameError.

- Python содержит три разных типа области видимости:

  * Локальная область видимости
  * Глобальная область видимости
  * Нелокальная область видимости (была добавлена в Python 3)

Полезные встроенные функции (built-in)

print, len, str, int, float, list, tuple, dict, set, range

bool, enumerate, zip, reversed, sum, max, min, sorted, any, all

type, filter, map, round, divmod, bin, oct, hex, abs, ord, chr, pow

Если не знаете или забыли, что за функция, то набираете

help(<имя функции>), например: help(print)

sorted(*iterable*, *key=None*, *reverse=False*)

Отсортируйте список целых чисел по возрастанию последней цифры.

Для этого надо воспользоваться операцией sorted и написать правильную функцию для key=.

Более сложный вариант, отсортировать по последней цифре, внутри группы с одинаковыми цифрами – по возрастанию самих чисел.

```
def fun1(x): return x % 10
def fun2(x): return (x % 10, x)

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

print(sorted(spi, key = fun1))     #
print(sorted(spi, key = fun2))     #
```

Список	222	21	1	111	12	322
fun1	2	1	1	1	2	2
fun2	(2, 222	(1, 21)	(1, 1)	(1, 111)	(2, 12)	(2, 322)

21, 1, 111, 222, 12, 322

1, 21, 111, 12, 222, 322

eval exec

Что напечатает?

```
# eval – evaluation (оценка, вычисление)
x = 5
print(eval('12 + 36'))
print(eval('x + 36'))
print(eval('divmod(x, 2)’))

# exec – execution (выполнение)
a = 'x = 5'
exec(a)
b = 'print(x)'
exec(b)
exec('''
for i in range( 5 ):
    print(i)
''')
```

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)

Задание

На вход программе подается строка генетического кода, состоящая из букв А (аденин), Г (гуанин), Ц (цитозин), Т (тимин).

Еще подается число – n.

Найдите и напечатайте самую часто встречаемую последовательность n букв в строке кода.

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
def fun2(x): return (x % 10, x)

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

print(sorted(spi, key = fun1)) #   print(sorted(spi, key = lambda x: x % 10))

print(sorted(spi, key = fun2)) #   print(sorted(spi, key = lambda x: (x % 10, x)))
```

Функция тождества (identity function)

- # функция, которая возвращает свой параметр
- **def** identity(x):
- return x
- # identity() принимает передаваемый аргумент в x и возвращает его при вызове.

- Лямбда-функция :
- **lambda** x: x

- Ключевое слово: lambda
- Параметр: x
- Выражение(тело): x

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 цикл, который печатает квадраты чисел от 0 до 9.

Именование lambda

- Поскольку лямбда-функция является выражением, оно может быть именовано.
- Поэтому вы можете написать предыдущий 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.

- >>>str = (**lambda** a='He', b='ll' , c='o': a+b+c)
- str(a='Ze')
- 'Zello'

Пример

- >>> (**lambda** x, y, z: x + y + z)(1, 2, 3)
- 6
- >>> (**lambda** x, y, z=3: x + y + z)(1, 2)
- 6
- >>> (**lambda** x, y, z=3: x + y + z)(1, y=2)
- 6
- >>> (**lambda** *args: sum(args))(1,2,3)
- 6
- >>> (**lambda** **kwargs: sum(kwargs.values()))(one=1, two=2, three=3)
- 6

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 x. Найти и напечатать самое близкий

- к числу x 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:  ????????? ))
```

Использование лямбда-выражения как литерала списка.

- import random
- # Словарь, в котором формируются три случайные числа
- # с помощью лямбда-выражения
- L = [**lambda** : random.random(),
- **lambda** : random.random(),
- **lambda** : random.random()]
- # Вывести результат
- for i in L:
- print(i**()**)

Лямбда-выражения как литералы 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 таблиц “переходов”

- # 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

Таблица “переходов”, в которой вычисляется площадь фигур

```
import math
Area = {
    'Circle' : (lambda r: math.pi*r*r), # окружность
    'Rectangle' : (lambda a, b: a*b),   # прямоугольник
    'Trapezoid' : (lambda a, b, h: (a+b)*h/2.0) # трапеция
    }
# Вызвать лямбда-выражение, которое выводит площадь окружности радиуса 2
print('Area of circle = ', Area['Circle'](2))
# Вывести площадь прямоугольника размером 10*13
print('Area of rectangle = ', Area['Rectangle'](10, 13))
# Вывести площадь трапеции для a=7, b=5, h=3
areaTrap = Area['Trapezoid'](7, 5, 3)
print('Area of trapezoid = ', areaTrap)
```

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.
- # Пример 1
- 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.

filter()

- # Пример 2
- even = **lambda** x: x%2 == 0
- list(**filter**(even, range(11)))
- [0, 2, 4, 6, 8, 10]

- # Обратите внимание, что filter() возвращает итератор, поэтому необходимо вызывать list, который создает список с заданным итератором.
- Реализация, использующая конструкцию генератора списка, дает одинаковый результат:
- >>> [x for x in range(11) if x%2 == 0]
- [0, 2, 4, 6, 8, 10]

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 суммы цифр каждого числа.
- Например.
- Вход: lst = [123, 234, 345, 456]
- Результат: [6, 9, 12, 15]

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

functools.**reduce**(*function*, *iterable*[, *initializer*])

- функция reduce() принимает два обязательных параметра — функцию и список.
- Сперва она применяет стоящую первым аргументом функцию для двух начальных элементов списка, а затем использует в качестве аргументов этой функции полученное значение вместе со следующим элементом списка и так до тех пор, пока весь список не будет пройден, а итоговое значение не будет возвращено.
- Например: reduce(lambda x, y: x + y, [1,2,3,4,5]) вычисляет ((((1+2)+3)+4)+5)

- Для того, чтобы использовать reduce(), вы должны сначала импортировать ее из модуля functools.

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

Заключение

- Избегать чрезмерного использования лямбд
- Использовать лямбды с функциями высшего порядка или ключевыми функциями Python
- Функции более высокого порядка, такие как map(), filter() и functools.reduce(), могут быть преобразованы в более элегантные формы с небольшими изменениями, в частности, со списком или генератором выражений.

Задание

Дан список из кортежей (Фамилия, премия).

Напечатать эти кортежи в порядке убывания премии.

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

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

Результат:

Лунин 200

Петров 200

Сидоров 200

Воробьев 100

Иванов 100

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:

[[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.

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.