Занятие 8

модули

str

lambda

Что напечатает?

```
def func(arg, *args, **kwargs):
        return len(args) + len(kwargs)
print(func(1))
print(func(1,2,3)
print(func(1,2,3,a=1,b=2,c=3))


tes = {1,2,3,4}
tes.discard(1);print(tes)
tes.discard(1);print(tes)
tes.remove(2);print(tes)
tes.remove(2); print(tes)
```

Задача 7-1

Напишите программу, которая рассчитывает НОК для списка натуральных чисел.

Задача 7-2

Напишите функцию, которая шифрует строку, содержащую латинские буквы с помощью шифра Цезаря. Каждая буква сдвигается на заданное число n позиций вправо. Пробелы, знаки препинания не меняются.

Например, для n = 1.

a -> b, b -> c, p -> q, y -> z, z -> a

A -> B, B -> C, Z->A

Т.е. заголовок функции будет def code(string, n):

В качестве результата печатается сдвинутая строка.

Задача 7-3

Дан x -двумерный массив чисел в виде списка, содержащего строки в виде списков. Размер массива n – строк и m – столбцов.

Напишите функцию, которая принимает этот массив как аргумент и в качестве результата выдает отсортированный список трех самых больших чисел.

Пример: n = 2, m = 3, x = [[1,6,3], [4,5,4]]

Результат: [4, 5, 6]

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)

zip

```
for k in zip([1,2,3,4,5], ‘abcdef’):
        print(k)
```

(1, ‘a’)

(2, ‘b’)

(3, ‘c’)

(4, ‘d’)

(5, ‘e’)

Останавливается на наименьшем итерируемом объекте, можно указывать несколько объектов

Сделайте zip((1,2,3,4), {1:111, 2:222}, {123, 456,789})

reversed

```
for k in reversed([1, 5, 2, 4, 3]):
    print(k)
```

```
3
4
2
5
1
```

А как еще можно перевернуть список?

filter

```
print(list(filter(bool, [0, 1, 2])))
```

[1, 2]

На месте bool может быть любая функция, которая выдает True или False, например, определенная нами.

```
def chet(x):
    if x % 2 == 0: return True
    return False
```

Что напечатает?

```
print(list(filter(chet, [0, 1, 2, 3, 4, 5, 6, 7])))
```

any all

```
print(any((True, False, False)))
print(all([True, False, False]))
```

Что напечатает?

```
print(all([]))
print(all([[]]))
print(all([[[]]]))
```

map

```
s = map(int, input().split())
```

Вводим 1 2 3

Получаем список s = [1, 2, 3]

Что напечатает?

```
k = 123
print(sum(map(int, list(str(k)))))
```

sorted

Все превращает в отсортированный список.

print(sorted([1, -2, 3, -4, 5]))

Результат: -4, -2, 1, 3, 5

Что напечатает?

print(sorted([1, -2, 3, -4, 5], key = abs))

Можно создать свою функцию и использовать ее для сортировки.

Напишите такую функцию, чтобы список был отсортирован по убыванию.

Подсказка: используйте в вашей функции abs, но со знаком минус.

Все вместе

***args -** произвольное число позиционных аргументов

****kwargs** - произвольное число именованных аргументов

Параметр с одним префиксом * может захватывать любое количество позиционных аргументов в кортеж.

Параметр с двойным префиксом ** может захватывать любое количество ключевых аргументов в словарь.

```
def many_all(var1, *args, **kwargs):
    print(var1)
    print(args)
    print(kwargs)
many_all(10, 34, 77, name='Piter', age=20)
```

```
10
(34, 77)
{'name': 'Piter', 'age': 20}
```

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)

Локальная область видимости

Локальная область видимости (local) — это блок кода или тело любой функции Python.

Эта область Python содержит имена, которые вы определяете внутри функции. В этих областях каждый разработчик может использовать одни и те же переменные, независимо друг от друга.

```
x = 100

def doubling(y):
    z = y*y


doubling(x)

print(z)
```

NameError: name 'z' is not defined

Где какая переменная s? Что будет напечатано?

```
s = 0
print('000', s)

def a():
    s = 1
    print('111', s)

    def b(): # Функция, вложенная в функцию a()
        s = 2
        print('222', s)

    b()
    print('333', s)

a()
print('444', s)
s = 3
print('555', s)
```

Глобальная область видимости (global)

В Python есть ключевое слово global, которое позволяет изменять изнутри функции значение глобальной переменной.

Оно записывается перед именем переменной, которая дальше

внутри функции будет считаться глобальной.

```
x = 100
def doubling(y):
        global x
        x = y*y
doubling(x)
print(x)
```

10000

Не используйте global без острой необходимости!

Нелокальная область видимости

В Python 3 было добавлено новое ключевое слово под названием **nonlocal**.
С его помощью мы можем добавлять переопределение области видимостей функций.

```
def counter():
    num = 0

    def incr():
        num += 1 # num = num + 1
        return num

    x = incr()
    return x
```

Если вы попробуете запустить этот код, вы получите ошибку UnboundLocalError, так как переменная num ссылается прежде, чем она будет назначена в самой внутренней функции.

Нелокальная область видимости

Добавим nonlocal в наш код:

```
def counter():
    num = 0

    def incr():
        nonlocal num
        num += 1
        return num

    x = incr()
    return x


inc = counter()
  print(inc)
1
```

Модули

Система модулей позволяет вам логически организовать ваш код на Python.

Группирование кода в модули значительно облегчает процесс написания и понимания программы.

Модуль в Python это **просто файл**, содержащий код на Python.

Каждый модуль в Python может содержать **переменные**, **объявления классов и функций**.

Кроме того, в модуле может находиться **исполняемый код**.

Команда **import**

Вы можете использовать любой питоновский файл как модуль в другом файле, выполнив в нем команду import.

Команда import в Python обладает следующим синтаксисом:

import *module_1*[, module_2[,... module_N]

Когда интерпретатор Python встречает команду import, он импортирует этот модуль, если он присутствует в пути поиска Python.

Путь поиска Python это список директорий, в которых интерпретатор производит поиск перед попыткой загрузить модуль.

math

Например, чтобы использовать модуль math следует написать:

```
import math


# Используем функцию sqrt из модуля math
print(math.sqrt(9))
# Печатаем значение переменной pi, определенной в math
print(math.pi)
```

Важно знать! Модуль загружается лишь однажды, независимо от того, сколько раз он был импортирован. Это препятствует цикличному выполнению содержимого модуля.

from *module* import *var*

Выражение from ... import ... не импортирует весь модуль, а только предоставляет доступ к конкретным объектам, которые мы указали.

Импортируем из модуля math функцию sqrt

from math **import** sqrt

Выводим результат выполнения функции sqrt.

Обратите внимание, что нам больше незачем указывать имя модуля

print (sqrt(144))

Но мы уже не можем получить из модуля то, что не импортировали !!!

print (pi) # Выдаст ошибку

from *module* import *var, func, class*

Импортировать из модуля объекты можно через запятую.

from **math** import pi, sqrt
print(**sqrt**(121))
print(pi)
print(e)

from *module* import *

В Python так же возможно импортировать всё (переменные, функции, классы) из модуля, для этого используется конструкция **from** ... **import** *

from math **import** *

Теперь у нас есть доступ ко всем функция и переменным, определенным в модуле math

print(sqrt(121))

print(pi)

print(e)

Не импортируются объекты с именами, начинающимися с ‘_’ (***_var)***

if __name__ == "__main__":

```
# test.py
def function():
    print("Function is called")

if __name__ == "__main__":
    print("Script is run directly")
    function()
else:
    print("Script is imported")
```

Некоторые функции из math

Тригонометрия: acos acosh asin asinh atan atan2 atanh cos cosh sin sinh tan tanh degrees radians

Округления: ceil floor trunc

Экспонента и логарифмы: exp log log10 log1p log2

Арифметика: dist remainder sqrt factorial gcd isqrt pow prod fsum

Великие постоянные: e pi inf nan isinf isnan isfinite

Задание

Напишите функцию, которая рассчитывает НОК (наименьшее общее кратное) двух чисел.

Подсказка. Попробуйте использовать функцию math.lcm

import *module_1* [,module_2]

За один раз можно импортировать сразу несколько модулей, для этого их нужно перечислить через запятую после слова import

import math, os

print(math.sqrt(121))

print(os.env)

Площадь круга

Напишите функцию, которая рассчитывает площадь круга.

Кто помнит формулу?

import *module* as *my_alias*

Если вы хотите задать псевдоним для модуля в вашей программе, можно воспользоваться вот таким синтаксисом

```
import math as matan
print(matan.sqrt(121))
```

```
import pandas as pd
import numpy as np
```

Получение списка всех модулей Python установленных на компьютере

Для того, чтобы получить список всех модулей, установленных на вашем компьютере достаточно выполнить команду:

```
>>>help("modules")
```

Через несколько секунд вы получите список всех доступных модулей.

Создание своего модуля в Python

Чтобы создать свой модуль в Python достаточно сохранить ваш скрипт с расширением .py
Теперь он доступен в любом другом файле.
Например, создадим два файла: module_1.py и module_2.py и сохраним их в одной директории.

```
# В первом запишем:
# module_1.py
def hello():
    print("Hello from module_1")


# А во втором вызовем эту функцию:
# module_2.py
from module_1 import hello
hello()
#Убедитесь, что это работает, вызовите функцию одного модуля из другого.
```

Сортировка кортежей

t = [(1,2), (0,2), (1,1), (0,0), (0,1), (1,0)]
#что напечатает?
print(sorted(t)) # → [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2)] – все по возрастанию
print(sorted(t, reverse = True)) # → [(1, 2), (1, 1), (1, 0), (0, 2), (0, 1), (0, 0)] – все по убыванию

Отсортируйте кортежи по первому элементу по возрастанию, а по второму по убыванию.

Определите функцию:
def fu(p):
    return p[0], -p[1]
И используйте ее в сортировке (key = fu)

А теперь наоборот, первый параметр по убыванию, а второй по возрастанию.

Задание

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.

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=.

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

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

print(sorted(spi, key = fun1))     #                    21, 1, 111,    222, 12, 322
```

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

Отсортировать по последней цифре, внутри группы с одинаковыми цифрами – по возрастанию самих чисел.

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

```
def fun2(x):
        return ????????

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

print(sorted(spi, key = fun2))        #                    1, 21, 111,    12, 22, 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

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

print(sorted(spi, key = fun1))

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

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:

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

Отсортируйте этот список по возрастанию общего количества цифр (не чисел!) 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.