Лекция 1 Знакомство с Python.

1. Python – язык с динамической типизацией.

value = 2809

name = 'Sergey'

При делении типа 5/2 будет возвращать 2,5 – т.е. тип динамически поменяется на float

Если необходима пустая переменная использовать конструкцию a = None

Переменные можно присваивать след образом:

A, b, c = 1, 2, 3

1. Комментарий начинается с # (# комментарий)
2. Ввод – вывод:

print('Введите а')

a = input()

print('Введите b')

b = input()

print(a, b)

print('{} -- {}'.format(a, b)) или print(f’{a}—{b}’)

Если нужно input c явным типом - a = int(input())

1. Арифметические операции

Стандартные, но есть:

// - аналог целочисленного деления в C#, т.е. 5/2 = 2

\*\* - возведение в степень

1. Особенность вывода вещественного числа, полученного при делении

Может вывести результат типа 5/2 = 2,500000000000000000000004

Чтобы этого не было можно использовать метод round( число, кол-во знаков после зпт)

1. Аналогично C# a = a + 1 можно заменить a +=1 – это работает со всеми арифметическими операциями
2. В Python нет массивов, но есть списки list

S = [] – пустой список

В списках могут быть данных различных типов (но это не рекомендуется!)

Можно выводить список с помощъю print()

Примеры объявления:

numbers = list(range(1, 6))

numbers = [1, 2, 3, 4, 5]

Чтение списков:

colors = ['red', 'green', 'blue']

for e in colors:

print(e) # red green blue

for e in colors:

print(e\*2) # redred greengreen blueblue

1. Лог. Операции
2. >, >=, <=, ==, != not, and, or – не путать с &, |, ^ Кое-что ещё: is, is not, in, not in
3. Пример конструкции if else

username = input('Введите имя: ')

if(username == 'Маша'):

print('Ура, это же МАША!')

else: print('Привет, ', username)

Важны отступы!

Else if в python выглядит как ELIF

1. Пример while

original = 23

inverted = 0

while original != 0:

inverted = inverted \* 10 + (original % 10)

original //= 10

print(inverted)

while может быть дополнен else

1. Конструкция FOR

Если знаем что хотим:

for i in 1, -2, 3, 14, 5:

print(i)

заметки по range:

r = range(100, 0, -20) #100 – стартовое число, 0 – минимальное число, -20 шаг

for i in r:

print(i) # 100 80 60 40 20

for i in range(5):

print(i) # 0 1 2 3 4

Циклы можно вкладывать друг в друга

1. Строковые операции:

text = 'съешь ещё этих мягких французских булок'

print(len(text)) # 39

print('ещё' in text) # True

print(text.isdigit()) # False

print(text.islower()) # True

print(text.replace('ещё','ЕЩЁ'))

print(text[0]) # c

print(text[1]) # ъ

print(text[len(text)-1]) # к

print(text[-5]) # б

print(text[:]) # print(text)

print(text[:2]) # съ

print(text[len(text)-2:]) # ок

print(text[2:9]) # ешь ещё

print(text[6:-18]) # ещё этих мягких

print(text[0:len(text):6]) # сеикакл

print(text[::6]) # сеикакл

text = text[2:9] + text[-5] + text[:2] # ...

1. Функции

Объявление:

def function\_name(x):

# body line 1

# . . .

# body line n

# optional return

Использование function\_name(x)

Лекция 2 Данные, функции, модули в python

1. Работа с файлами:

- Python может считывать и записывать данные в файлы

- Для работы с файлами необходимо связать файловую переменную определив модификатор:

a – открытие для добавление данных (если его нет, то он создастся)

r – открытие для чтения

w – открытие для записи (если его нет, то он создастся)

w+,r+

Примеры кода:

Data = Open(path, ‘r) – Открытие файла только для чтения

Data.close() – Закрытие файла

Exit() – выход

1. Функции и модули

Разбитие приложения:

Существует def func, чтобы сослаться на нее нужно использовать import func.f()

Как вариант использовать import func as fu, пример кода использования print(fu.f())

Значения по умолчанию: в функции можно исп значения по умолчанию, нпример

Def new\_string(symbol, count = 3) – в таком случае если при вызове функции мы не укажем count, а только строку – будет подставлено значение 3.

Передача неогр.кол-ва аргументов: перед аргументом ставится \*, например def func(\*params)

Рекурсия аналогична – необходимо условие выхода!

1. КОРТЕЖИ

- неизменяемый список для создания пар значений, например:

a = (1, 2)

Обращение аналогично спискам a[0] даст 1, отрицательные индексы могут использоваться.

ДЛЯ КОРТЕЖЕЙ НЕЛЬЗЯ МЕНЯТЬ ЗНАЧЕНИЯ!

Чтобы создать кортеж с одним элементом необходимо ставить запятую, например a = (3,)

Кортежи перебираются след образом:

B = (1, 2, 3)

For item in B: и так далее

Список можно преобразовать в кортеж:

T = tuple([a,b,c])

a, b, c = t

далее их можно использовать как обычные переменные

1. СЛОВАРИ

Это неупорядоченные коллекции произвольных объектов с доступом по ключу

Dict = {}

Dict =\

{

‘up’: ‘1’,

‘down’:’2’

}

Print(dict[up]) – выведет ‘1’

Перебор словаря:

For k in dict.keys(): - только ключи

For k in dict.values(): - только значения

For k in dictionary: - общий

Изменяется словарь по ключу, по аналогии со списком

1. МНОЖЕСТВА

Это переменная в которой содержится несколько значений

Например:

Colors = {“red”, “green”, “blue”)

Добавление элемента в множество – colors.add(“gray”)

Если мы попытаемся добавить в множество какой-то элемент, по значению одинаковый как в множестве – ЭЛЕМЕНТ НЕ ДОБАВИТСЯ!

Удаление – colors.remove(“red”), но если такого элемента в множестве не существует, будет ОШИБКА

Для обхода исключения при удалении несущ. Элемента исп. Colors.discard(“red”)

Очистка множества – colors.clear() – вернет пустое множество

Копирование множества c = a.copy()

Объединение с = a.union(b) – вернет все значения из a и b, при этом повт значения будут удалены!

Перечение c = a.intersection(b) – вернет все значения из a и b которые совпадают в обоих множествах

Разность .difference()

1. СПИСКИ – ДОПОЛНЕНИЕ!

Если списки взаимосвязаны с помощью = - их значения будут меняться вместе. Т.е. если в одном из таких списков мы поменяли значение, то тоже значение попадет и в другой список, при этом не имеет значения в каком этапе произошло изменение списка.

List.pop() – удаление последнего элемента с уменьшение длины списка

List.pop(2) – удаление элемента в индексе 2

List.insert(2, элемента) – добавление элемента на нужный индекс

Лекция 3 Ускоренная обработка данных Lambda, Filter, Map, Zip, Enumerate, List Comprehension

1. **LAMBDA**

Lambda функции – анонимные, используются один раз!

Пример без Lambda:

Def f(x):

Return x\*\*2

Print(f(2))

Пример с переменной хранящей ссылку на функцию:

Def f(x):

Return x\*\*2

G = f

Print(G(2))

Функцию можно передать в функцию как переменную и использовать ее

Def Math(func,x)

Print(func(x))

Пример LAMBDA:

Было:

def sum (x, y):

return x + y

Стало

Sum = lambda x, y: x + y

Было calc(sum, 5, 6), но можно calc(lambda x,y: x+y, 5, 6)

1. **LIST COMPREHENSION**

Нужно для быстрого создания списка

Было:

List []

For I in range(1, 101):

If (i%2 ==0):

List.append(i)

Print(list)

Стало:

List = [I for I in range (1, 101) if i%2 ==0]

Print(list)

Пример с кортежем:

List = [(I,I) for I in range (1, 101) if i%2 ==0]

Print(list)

Пример с функцией:

Def f(x):

Return x\*\*2

List = [(i,f(i)) for I in range (1, 5) if i%2 ==0]

Вывод: [(2, 4), (4,16)]

Пример с лямбдой:

List = [(I, lambda i: I\*\*2) for I in range (1,5) if i%2 ==0]

1. **MAP**

Данная функция применяет указанную функцию к каждому элементу итерируемого объекта и возвращает итератор с новыми объектами.

List = list(Map(lambda i: I\*\*2, [1,2,3])) \*вместо лист может быть set и тд

Результат: [1,4,9]

1. **FILTER**

Данная функция применяет указанную функцию к каждому элементу итерируемого элемента и возвращает итератор с теми обхектами, для которых функция вернула TRUE

Data = [x for x in range(10)]

Res = set(filter(lambda x: x%2==0, data)) или Res = set(filter(lambda x: not x%2, data))

Результат:

(2, 4, 6, 8, 10)

1. **ZIP**

Функция применяется набору объектов и возвращает итератор с кортежами из элементов входных данных.

A = [1,2,3]

B = [a,b,c]

Zip(A,B)

Результат [(1,a), (2,b), (3,c)]

Кол-во элементов = мин кол-ву элементов одного из входного набора

Дважды пройтись нельзя!

1. **ENUMERATE**

Выводит список с кортежами (итератор, элемент списка):

Enumerate([a,b,c])

Результат: [(1,a), (2,b), (3,c)]

ЛЕКЦИЯ 4 Python: от простого к практике

-На практике лучше придерживаться универсальности программ, модульности и некоторые функции которые могут использоваться несколько раз – оборачивать в отдельные программы – методы.

Например: для сложения – один метод, для кнопки иной метод

-Все методы вызываются в main – главной программе.

Пример модульности:  
Дано задача:

Система собирает информацию с датчиков, в ней есть модуль логирования, который заносит в журнал все обращения к датчикам. В системе есть модуль, выполняющий обращения для получения данных с датчиков и модуль генерации htmlпредставления. Запуск системы осуществляется из головного модуля.

Задача разделяется на модули:

Модуль 1: сбор информации с датчиков data\_provider

Модуль 2: логирование logger

Модуль 3: UI user\_interface

Модуль 4: HTML-генератор html\_creater

Модуль 5: главный модуль main

В каждом модуле будут методы:

data\_provider

get\_temperature, get\_pressure, get\_wind\_speed

logger

temperature\_logger, pressure\_logger, wind\_speed\_logger

user\_interface

temperature\_view, pressure\_view, wind\_speed\_view

html\_creater

create

main

ЛЕКЦИЯ 5 Возможна ли жизнь без PIP?

Для установки дополнительных расширений или модулей нужно установить PIP:

<https://otus.ru/journal/ustanovka-pip-na-python-3-bazovye-komandy/>

Далее для установки расширения использовать (пример Kivy)

pip install "kivy[full]" kivy\_examples