MACHINE LEARNING

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON И РАЗРАБОТКА ПРОГРАММ ДЛЯ МАШИННОГО ОБУЧЕНИЯ

ЛЕКЦИЯ III

План занятия

Библиотека Math

Циклы:

- while
- for

Строки:

- последовательности
- срезы
- индексация
- массивы
- кортежи
- специальные функции
- строковые методы

Библиотека Math

```
import math #подключение библиотеки math
```

```
math.sin(x)
y=math.sin(x)
print(math.sin(math.pi/2))
```

```
from math import *
x=pi/2;
y=sin(x);
print(y)
```

Библиотека Math

Функция	Описание
Округление	
`int(x)`	Округляет число в сторону нуля. Это стандартная функция, для ее использования не нужно подключать модуль `math`.
`round(x)`	Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
`round(x, n)`	Округляет число `x` до `n` знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль `math`.
`floor(x)`	Округляет число вниз (“пол”), при этом `floor(1.5) == 1`, `floor(-1.5) == -2`
`ceil(x)`	Округляет число вверх (“потолок”), при этом `ceil(1.5) == 2`, `ceil(-1.5) == -1`
`trunc(x)`	Округление в сторону нуля (так же, как функция `int`).
`abs(x)`	Модуль (абсолютная величина). Это - стандартная функция.
`fabs(x)`	Модуль (абсолютная величина). Эта функция всегда возвращает значение типа `float`.

Библиотека Math

Корни, степени, логарифмы	
`sqrt(x)`	Квадратный корень. Использование: `sqrt(x)`
`pow(a, b)`	Возведение в степень, возвращает a^b. Использование: `pow(a,b)`
`exp(x)`	Экспонента, возвращает e^x. Использование: `exp(x)`
`log(x)`	Натуральный логарифм. При вызове в виде `log(x, b)` возвращает логарифм по основанию `b`.
`log10(x)`	Десятичный логарифм
`e`	Основание натуральных логарифмов $e \approx 2{,}71828\ldots$

Библиотека Math

Тригонометрия	
`sin(x)`	Синус угла, задаваемого в радианах
`cos(x)`	Косинус угла, задаваемого в радианах
`tan(x)`	Тангенс угла, задаваемого в радианах
`asin(x)`	Арксинус, возвращает значение в радианах
`acos(x)`	Арккосинус, возвращает значение в радианах
`atan(x)`	Арктангенс, возвращает значение в радианах
`atan2(y, x)`	Полярный угол (в радианах) точки с координатами (x, y).
`hypot(a, b)`	Длина гипотенузы прямоугольного треугольника с катетами a и b.
`degrees(x)`	Преобразует угол, заданный в радианах, в градусы.
`radians(x)`	Преобразует угол, заданный в градусах, в радианы.
`pi`	Константа π

Циклы

Циклы - это инструкции, выполняющие одну и ту же последовательность действий многократно.

В Python имеются два вида циклов: цикл **ПОКА** (выполняется некоторое условие) и цикл **ДЛЯ** (всех значений последовательности)

1) while (с условием)
2) for (с переменной)

Цикл while

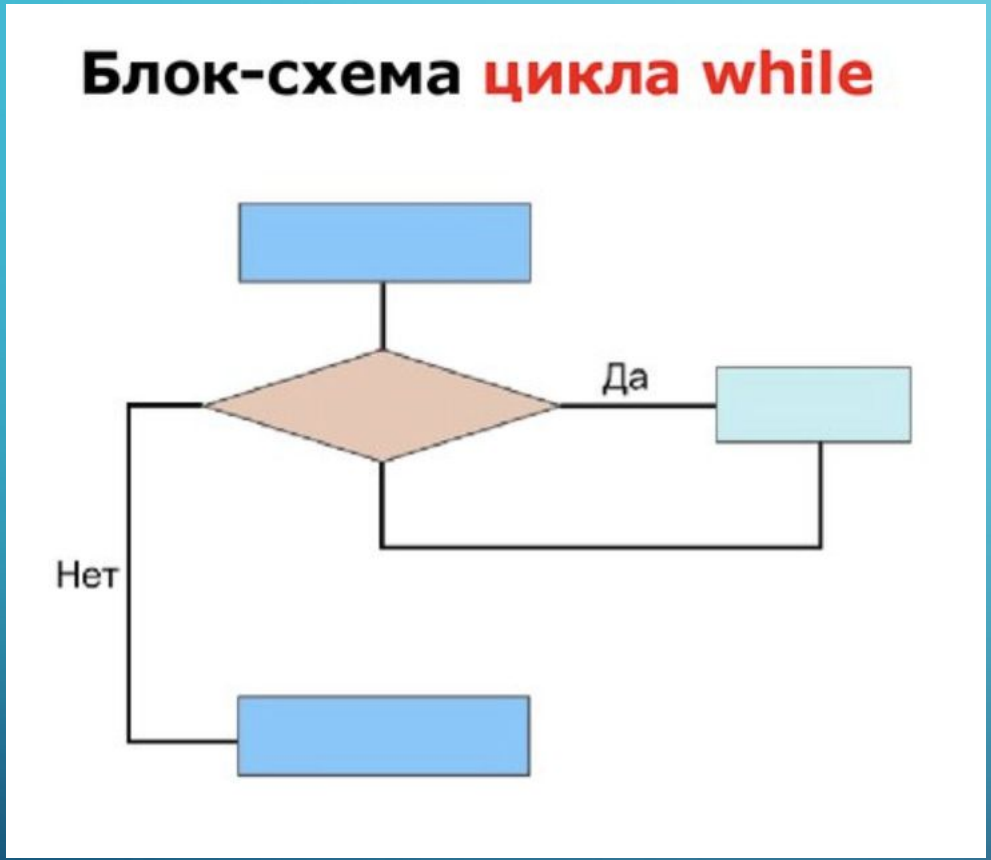
- это инструкции, выполняющие одну и ту же последовательность действий (итерацию), пока действует заданное условие

```
while a логический_оператор b:
    действие(я)
    изменение a
```

Пример:

```
i=1
while i<=10:
    print(i)
    i+=1
```

Цикл while



Пример:

```
a=0
b=1
print(a)
print(b)
n=10
k=0
while k<n:
    sum=a+b
    print(sum)
    a=b
    b=sum
    k+=1
```

Цикл while
Инструкция управления циклом

break – немедленное прекращение цикла
continue – продолжение цикла

Пример:

```
count=0
while True:
    count+=1
    if count>10:
        break
    if count==5:
        continue
    print(count)
```

Цикл for

- используется для повторения какой-либо последовательности действий (итераций) заданное число раз (совместно с функцией range)

- используется для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного

- используется для обхода заданного множества элементов (символов строки или объектов списка)

Цикл for

```
for i  in range(n):    # n != 0, n > 0
    Тело цикла
```

а)
```
for i  in range(a, b):    # переменная i  будет принимать
                          значения от a до b – 1, a <= b
    Тело цикла
```

б)
```
for i  in range(a, b, c):    # переменная i  будет
                             принимать значения от a до
    Тело цикла               b – 1,
                             c – шаг индексной
                             переменной
```

Цикл for
Примеры использования цикла for

1. # Вывод 'Hello' 5 раз и 'There' один раз
```
for i  in range(5):
   print ("Hello")
print ("There")
```

2. # Вывод 'Hello', 'There' 5 раз
```
for i  in range(5):
   print ("Hello")
   print ("There")
```

3. # Два способа вывода цифр от 1 до 10
```
for i  in range(1, 11):
   print (i, end=" ")
for i  in range(10):
   print (i+1)
```

4. # Два способа вывода четных цифр от 2 до 10
```
for i  in range(2, 12, 2):
   print (i)
for i  in range(5):
   print ((i+1)*2)
```

5. # Вывод цифр от 10 до 1
```
for i  in range(10, 0, -1):
   print(i)
```

6. # Что выводится? Почему?
```
for i in range(3):
   print ("a")
   for j in range(3):
         print ("b")
```

Строки (str)

Строки (string) — набор символов, заключенных в кавычки (например, "ball", "What is your name?", 'dkfjUUv', '6589'). Примечание: кавычки в Python могут быть одинарными или двойными.

Строки (str)
Последовательности

Последова-тельность	Описание
\\	Обратный слеш. Выводит один знак обратного слеша
\'	Апостроф, или одиночная кавычка. Выводит один апостроф
\"	Двойные кавычки. Выводит одну такую кавычку
\n	Пустая строка. Перемещает курсор в начало следующей строки
\t	Горизонтальный отступ – символ табуляции. Перемещает курсор вправо на один отступ

Строки (str)
Специальные функции

Функция **len()** определяет длину строки.

Оператор **in** определяет, является ли какой-либо символ элементом строки

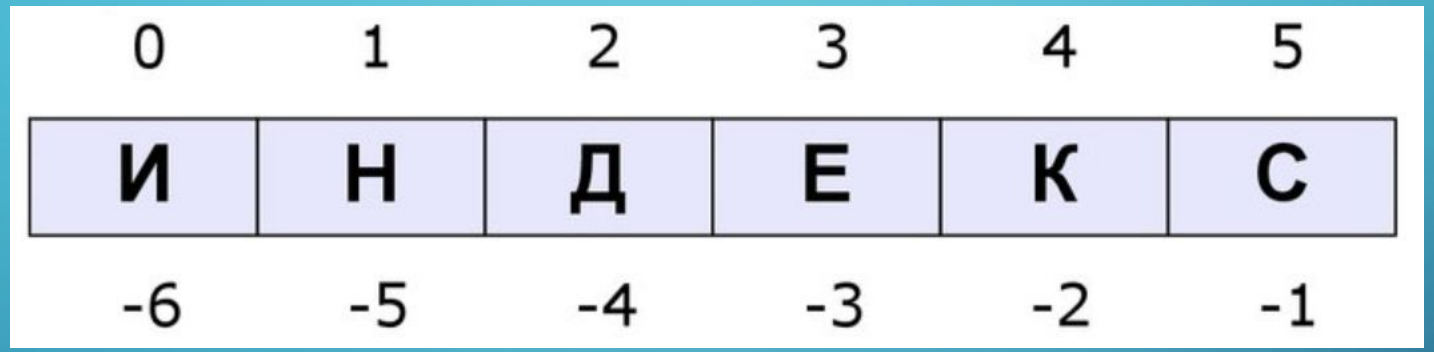
A + B — конкатенация (строка B приписывается к строке A);
A * n — повторение n раз, значение n должно быть целого типа.

Пример:

```
>>> len ('It is a long string')                    определение длины строки
19
>>> '!!!' + ' Hello World ' + '!!!'                конкатенация
'!!! Hello World !!!'
>>> '#' * 20                                       дублирование
'####################'
>>> if  'e'  in  ' Hello World ' :
        print ('встречается в тексте')
else:
        print ('не встречается в тексте')
встречается в тексте
```

Строки (str)
Индексация

Индекс – уникальный порядковый номер символов в строке (а также в других структурных данных: списках, кортежах).

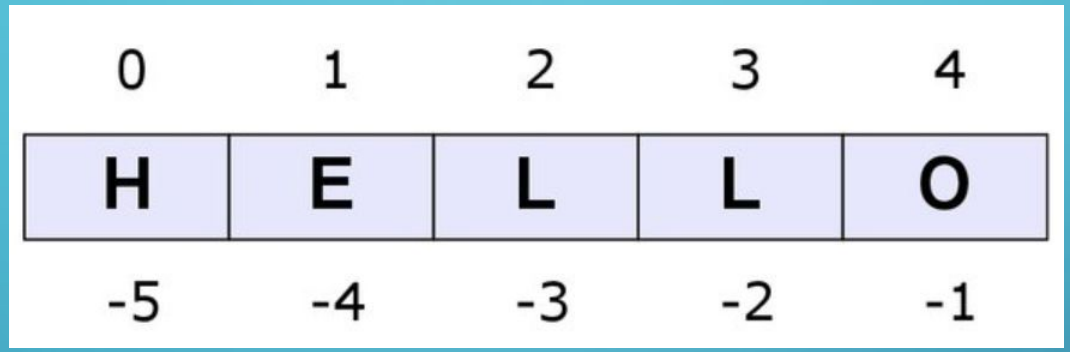


1.
```
>>> 'индекс' [0]
'и'
>>> 'индекс' [-1]
'с'
>>> 'индекс' [-3]
'е'
>>>
```

2.
```
>>> tday = 'morning, afternoon, night'
>>> tday [4]
'i'
>>> a = tday[1]
>>> a
'o'
>>>
```

Строки (str)
Срезы (slices)

Срезы (slices) – извлечение из данной строки одного символа или некоторого фрагмента (подстроки)



Оператор извлечения среза из строки выглядит так [**X:Y**].
X – индекс **начала среза**
Y – индекс **окончания среза** (символ с номером Y в срез не входит).

```
>>> s = 'hello'
>>> s[1:4]
'ell'
>>>
```

```
>>> s = 'hello'
>>> s[-4:-1]
'ell'
```

Строки (str)
Срезы (slices)

Если отсутствует первый индекс, то срез берется от начала до второго индекса

```
>>> a = 'very big string'
>>> a [ : 4]
'very'
```

При отсутствии второго индекса, срез берется от первого индекса до конца строки

```
>>> a = 'very big string'
>>> a [ 9 : ]
'string'
```

Если оператор извлечения среза из строки выглядит так: **[X:Y:Z]**, **Z** – шаг, через который выбирают элементы

```
>>> a = 'very big string'
>>> a [ : : 3]
'vyisi'
```

Строки (str)
Строковые методы

Метод – это функция, применяемая к объекту (в данном случае – к строке)

Вызов метода:

имя_объекта.имя_метода(параметры)

Строковые методы:

1) find
2) rfind
3) replace
4) count

Метод find

Метод find находит в данной строке данную подстроку (которая предлагается в качестве параметра). Функция возвращает индекс первого вхождения искомой подстроки. Если же подстрока не найдена, то метод возвращает значение -1.

Например:
```
>>> s = 'Hello'
>>> print(s.find('e'))
1
>>> print(s.find('ll'))
2
>>> print(s.find('L'))
-1
```

Метод rfind

Метод rfind возвращает индекс последнего вхождения данной строки (“поиск справа”).

Если вызвать метод find с тремя параметрами s.find(T, a, b), то поиск будет осуществляться в срезе s[a:b]. Если указать только два параметра s.find(T, a), то поиск будет осуществляться в срезе S[a:], то есть начиная с символа с индексом a и до конца строки.

1.
```
>>> s = 'Hello'
>>> print(s.find('l'))
2
>>> print(s.rfind('l'))
3
```

2.
```
>>> s='Hello'
>>> print(s.find('l',1,4))
2
>>> print(s.find('H',1))
-1
```

Метод replace

Метод replace - s.replace(old, new) – заменяет в строке s все вхождения подстроки old на подстроку new.

Пример:
```
>>> 'Hello'.replace('l', 'L')
'HeLLo'
```

Формат: s.replace(old, new, count) – заменены будут не все вхождения, а только не больше, чем первые count из них.

Пример:
```
>>> 'Abrakadabra'.replace('a', 'A', 2)
'AbrAkAdabra'
```

Метод count

Метод count s.count(T) возвращает число вхождений строки T внутри строки S.

Пример:
```
>>> 'Abracadabra'.count('a')
4
```

При указании трех параметров s.count(T, a, b), будет выполнен подсчет числа вхождений строки T в срез S[a:b].

Строки (str)
Строковые методы

Метод	Описание
upper()	Возвращает строку, символы которой приведены к верхнему регистру
lower()	Возвращает строку, символы которой приведены к нижнему регистру
swapcase()	Возвращает новую строку, в которой регистр всех символов обращен: верхний становится нижним и наоборот
capitalize()	Возвращает новую строку, в которой первая буква прописная, а остальные – строчные
title()	Возвращает новую строку, в которой первая буква каждого слова прописная, а остальные – строчные
strip()	Возвращает строку, из которой убраны все интервалы(табуляция, пробелы, символы пустых строк) в начале и в конце

План следующего занятия

- Функции
- Локальные и глобальные переменные
- Рекурсия
- Двумерные массивы
- Вложенные списки и массивы

Спасибо за внимание!