Циклы, функции Python и работа с библиотеками

Цикл while

Циклический оператор while используется достаточно редко, но все же бывает полезен. Определенный блок кода выполняется, пока выполняется заданное условие. Пример:

```
#while
n = 0
while n != 10:
    print(n, end=' ')
    n+=1
```

```
0 1 2 3 4 5 6 7 8 9
```

end в выводе позволяет указать, какой знак будет стоять после вывода, по умолчанию – это \n (переход на новую строку). У циклов есть continue и break. Если в цикле используется continue, то цикл будет переходить сразу к следующей итерации, не выполняя код ниже. В случае break – происходит полное прерывание цикла. Пример:

```
t = True
a = 0
while t:
    print(a, end = ' ')
    if a < 5:
        a+=1
        continue
    elif a>=5 and a<=10:
        a=a+2
        continue
    else:break
```

```
0 1 2 3 4 5 7 9 11
```

Цикл for

Цикл с использованием ключевого слова for – это прохождение по элементам некой последовательности. Например, необходимо вывести элементы, хранящиеся в списке:

```
#for
l = [1,2,3,4,5]
for x in l:
    print(x, end = ' ')
```

```
1 2 3 4 5
```

Второй способ – это вывод элементов списка по их индексу:

```
l = [1,2,3,4,5]
for i in range(len(l)):
    print(l[i], end = ' ')
```

```
1 2 3 4 5
```

range – это генератор чисел, в данном случае он генерирует индексы от 0 до длины списка-1, то есть 0,1,2,3,4, затем вывод происходит через обращение к элементу списка по индексу. Примеры использования генератора:

```
print([i for i in range(1,10,2)])
```

```
[1, 3, 5, 7, 9]
```

range(start, stop, step). start – начало, stop – конечное значение (обратите внимание, что stop не входит в интервал генерации чисел), step – шаг.

```
x=range(5)
print(x, type(x))
print(list(x))
```

```
range(0, 5) <class 'range'>
[0, 1, 2, 3, 4]
```

Функции

Функция – блок кода, который выполняется при его вызове в ходе выполнения программы.

Функция в Python начинается с ключевого слова def, далее идет имя функции и (). В () указываются аргументы функции. Сначала идут аргументы, который не даны по умолчанию, затем аргументы, которые даны по умолчанию. Тело функции выделяется отступом. Пример:

```
# Функции
def function(x, y=2):
    if x%y == 0:
        return ('Число нацело делится на %d'%y)
    else:
        return ('Число нацело не делится на %d'%y)

print(function(4))
print(function(4,3))
```

```
Число нацело делится на 2
Число нацело не делится на 3
```

Ключевое слово return – это результат, который возвращает функция.

Библиотека numpy

numpy это библиотека для работы с матрицами и векторами.

Для работы с библиотекой ее необходимо импортировать с помощью команды import.

```
import numpy as np
```

Генерация массива в numpy:

```
a = np.arange(5,10)
print(a, type(a))
```

```
[5 6 7 8 9] <class 'numpy.ndarray'>
```

Можно привести список к numpy массиву:

```
x = np.array([1,2,3,4,5],dtype=float)
print(x, type(x))
```

```
[1. 2. 3. 4. 5.] <class 'numpy.ndarray'>
```

dtype позволяет указать тип хранимых элементов, в данном случае – это вещественные числа.

Двумерные (и не только) массивы:

```
xx = np.array([[1,1,1],[2,2,2],[3,3,3]])
print(xx)
```

```
[[1 1 1]
 [2 2 2]
 [3 3 3]]
```

```python
print(xx.ndim) #кол-во размерностей
print(xx.shape) #кол-во строк и столбцов
```

```
2
(3, 3)
```

```python
#индексация
print(xx[0])
print(xx[1,0])
```

```
[1 1 1]
2
```

```python
#операции с векторами и матрицами
print(xx*xx) # умножение поэлементное
print(xx@xx) # умножение матричное
print(xx.dot(xx))# умножение матричное
print(xx+xx) # сложение
print(xx-xx) # вычитание
print(xx/xx) # деление
```

```
[[1 1 1]
 [4 4 4]
 [9 9 9]]
[[ 6  6  6]
 [12 12 12]
 [18 18 18]]
[[ 6  6  6]
 [12 12 12]
 [18 18 18]]
[[2 2 2]
 [4 4 4]
 [6 6 6]]
[[0 0 0]
 [0 0 0]
 [0 0 0]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

```python
#транспонирование матрицы
print(xx.T)
```

```
[[1 2 3]
 [1 2 3]
 [1 2 3]]
```

np.random.rand() – генерация случайных значений из равномерного распределения от 0 до 1.

```
print(np.random.rand())
```

```
0.4893602888250862
```

```
print(np.random.rand(3,2))
```

```
[[0.76598118 0.67268707]
 [0.61702042 0.60233133]
 [0.9413086  0.07138761]]
```

np.random.rand() генерация случайных значений из нормального распределения с дисперсией, равной 1, и средним, равным 0.

Библиотека pandas

Pandas – это библиотека для обработки и анализа данных. В pandas есть две структуры объекта: DataFrame и Series. Series – это аналог одномерного массива, который имеет индексы, по сути, это ассоциативный массив, как словарь. DataFrame имеет табличную структуру, столбцы у DataFrame – это Series.

```
import pandas as pd
```

```
S = pd.Series(['Мышка','Норушка'])
print(S)
```

```
0      Мышка
1    Норушка
dtype: object
```

```
print(S.index) #индексы
print(list(S.index)) #индексы

print(S.values) #список значений
```

```
RangeIndex(start=0, stop=2, step=1)
[0, 1]
['Мышка' 'Норушка']
```

```
print(S[0], S[1])
```

```
Мышка Норушка
```

```python
#индексы можно задать ручками
new_s = pd.Series(S.values, index=['Лягушка','Квакушка'])
new_s['3']=1 #добавление нового значения
print(new_s)
```

```
Лягушка       Мышка
Квакушка    Норушка
3                 1
dtype: object
```

```python
#групповое присваивание
new_s[['Лягушка','Квакушка']]=3,2
new_s.index=[1,2,3]
print(new_s)
```

```
1    3
2    2
3    1
dtype: object
```

```python
print(new_s[new_s>=2]) #вывод элементов больше или равным 2
```

```
1    3
2    2
dtype: object
```

DataFrame

Загрузка данных из файла .csv в DataFrame.

```python
data = pd.read_csv('Данные.csv', sep = ';',dtype=float)
```

```python
data
```

]:

	A	B	C
0	1.0	2.0	6.0
1	2.0	4.0	12.0
2	3.0	6.0	18.0
3	4.0	8.0	24.0
4	5.0	10.0	30.0
5	6.0	12.0	36.0
6	7.0	14.0	42.0
7	8.0	16.0	48.0
8	9.0	18.0	54.0

Аргумент sep позволяет указать разделитель. Также доступно чтение из txt (read_txt) файла и xlsx файла (read_excel).

Задать столбцы и значения можно вручную:

```
frame = pd.DataFrame({'A':range(1,11),'B':range(100,1001,100)})
```

```
print(frame['A'], type(frame['A']))
```

```
0     1
1     2
2     3
3     4
4     5
5     6
6     7
7     8
8     9
9    10
Name: A, dtype: int64 <class 'pandas.core.series.Series'>
```

Полезным является метод describe(), который позволяет узнать основные статистические показатели по столбцам датафрейма.

```
print(data.columns)
print(data.index)
print(data.values)
```

```
Index(['A', 'B', 'C'], dtype='object')
RangeIndex(start=0, stop=9, step=1)
[[ 1.  2.  6.]
 [ 2.  4. 12.]
 [ 3.  6. 18.]
 [ 4.  8. 24.]
 [ 5. 10. 30.]
 [ 6. 12. 36.]
 [ 7. 14. 42.]
 [ 8. 16. 48.]
 [ 9. 18. 54.]]
```

```
print(data.describe())
```

```
              A          B          C
count  9.000000   9.000000   9.000000
mean   5.000000  10.000000  30.000000
std    2.738613   5.477226  16.431677
min    1.000000   2.000000   6.000000
25%    3.000000   6.000000  18.000000
50%    5.000000  10.000000  30.000000
75%    7.000000  14.000000  42.000000
max    9.000000  18.000000  54.000000
```

Можно создать новый столбец, используя известные данные:

```
data['D']=data['A']+data['B']+data['C']
print(data)
```

```
     A     B     C     D
0  1.0   2.0   6.0   9.0
1  2.0   4.0  12.0  18.0
2  3.0   6.0  18.0  27.0
3  4.0   8.0  24.0  36.0
4  5.0  10.0  30.0  45.0
5  6.0  12.0  36.0  54.0
6  7.0  14.0  42.0  63.0
7  8.0  16.0  48.0  72.0
8  9.0  18.0  54.0  81.0
```

Значения в DataFrame или Series можно отфильтровать:

```
data[(data['B']>=8) & (data['C']<48)]
```

	A	B	C	D
3	4.0	8.0	24.0	36.0
4	5.0	10.0	30.0	45.0
5	6.0	12.0	36.0	54.0
6	7.0	14.0	42.0	63.0

Обращение к датафрейму, метод iloc – индексация по позиции:

```
data.iloc[[0,1,4],[0,1]]
```

	A	B
0	1.0	2.0
1	2.0	4.0
4	5.0	10.0

Метод loc – индексация на основе имени:

```
data.loc[[0,1,4],['B','C']]
```

	B	C
0	2.0	6.0
1	4.0	12.0
4	10.0	30.0

Так же с помощью loc можно фильтровать данные, например, выведем данные, где значение признака A > 3 и значение признака B < 16:

```
1 data.loc[(data.A > 3) & (data.B < 16)]
```

	A	B	C	D
3	4.0	8.0	24.0	36.0
4	5.0	10.0	30.0	45.0
5	6.0	12.0	36.0	54.0
6	7.0	14.0	42.0	63.0

Метод apply() позволяет применить к серии или датафрейму некоторую функцию, например, создание нового признака, если значение в столбце A меньше 5 или больше:

```
1 def function(x):
2     if x['A']<5:
3         x['new']=x['A']
4     else:
5         x['new']=x['A']**2
6     return x
7
8 a=data.apply(function,axis = 1)
9 print(a)
```

```
     A     B     C     D   new
0  1.0   2.0   6.0   9.0   1.0
1  2.0   4.0  12.0  18.0   2.0
2  3.0   6.0  18.0  27.0   3.0
3  4.0   8.0  24.0  36.0   4.0
4  5.0  10.0  30.0  45.0  25.0
5  6.0  12.0  36.0  54.0  36.0
6  7.0  14.0  42.0  63.0  49.0
7  8.0  16.0  48.0  72.0  64.0
8  9.0  18.0  54.0  81.0  81.0
```

axis = 1 – это проход по строкам, axis = 0 – это проход по столбцам, например:

```python
def f(x):
    mean = x.mean()
    print(x.name, mean)
    return x - mean
```

```python
print(data.apply(f,axis = 0))
```

```
A 5.0
B 10.0
C 30.0
D 45.0
     A    B     C     D
0 -4.0 -8.0 -24.0 -36.0
1 -3.0 -6.0 -18.0 -27.0
2 -2.0 -4.0 -12.0 -18.0
3 -1.0 -2.0  -6.0  -9.0
4  0.0  0.0   0.0   0.0
5  1.0  2.0   6.0   9.0
6  2.0  4.0  12.0  18.0
7  3.0  6.0  18.0  27.0
8  4.0  8.0  24.0  36.0
```

Библиотека scikit-learn.

scikit-learn – это библиотека для эффективного предиктивного анализа данных. Она содержит в себе различные алгоритмы машинного обучения, инструменты для предобработки данных, готовые наборы данных для исследований и тд.

Готовые датасеты можно взять в модуле dataset. У каждого датасета есть функция для его загрузки.

Рассмотрим пример загрузки датасета Ames housing:

```python
from sklearn.datasets import fetch_openml
```

```python
df = fetch_openml(name ="house_prices", as_frame=True)
print(type(df))
```

```
<class 'sklearn.utils._bunch.Bunch'>
```

В результате df – это объект Bunch, который представляет собой аналог словаря в библиотеке Scikit-Learn. Соответственно, у него есть ключи.

```python
df.keys()
```

```
dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_names', 'DESCR', 'details', 'url'])
```

data – это таблица признаков. target – это значение целевой переменной. Этот набор данных используется для прогнозирования стоимости дома.

```
1 df.data
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence
0	1.0	60.0	RL	65.0	8450.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	None
1	2.0	20.0	RL	80.0	9600.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	None
2	3.0	60.0	RL	68.0	11250.0	Pave	None	IR1	Lvl	AllPub	...	0.0	0.0	None	None
3	4.0	70.0	RL	60.0	9550.0	Pave	None	IR1	Lvl	AllPub	...	0.0	0.0	None	None
4	5.0	60.0	RL	84.0	14260.0	Pave	None	IR1	Lvl	AllPub	...	0.0	0.0	None	None
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456.0	60.0	RL	62.0	7917.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	None
1456	1457.0	20.0	RL	85.0	13175.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	MnPrv
1457	1458.0	70.0	RL	66.0	9042.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	GdPrv
1458	1459.0	20.0	RL	68.0	9717.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	None
1459	1460.0	20.0	RL	75.0	9937.0	Pave	None	Reg	Lvl	AllPub	...	0.0	0.0	None	None

1460 rows × 80 columns

Подробнее данную библиотеку рассмотрим в следующих практических работах.

Практическая работа

1. Напишите программу, которая считывает с консоли числа (по одному в строке) до тех пор, пока сумма введённых чисел не будет равна 0 и после этого выводит сумму квадратов всех считанных чисел.
2. Напишите программу, которая выводит последовательность чисел, длинною N, где каждое число повторяется столько раз, чему оно равно. На вход программе передаётся неотрицательное целое число N. Например, если N = 7, то программа должна вывести 1 2 2 3 3 3 4. Вывод элементов списка через пробел – print(*list).
3. Матрицу произвольного размера вытянуть в один вектор, не применяя встроенные методы Python. Пример:

```
Исходная матрица
 [[1 1 1]
 [1 1 1]
 [1 1 1]]
Результат  [1, 1, 1, 1, 1, 1, 1, 1, 1]
```

Для создания матрицы можно использовать np.random.rand(кол-во строк, кол-во столбцов).

4. Даны два списка:

   A = [1, 2, 3, 4, 2, 1, 3, 4, 5, 6, 5, 4, 3, 2]

   B = [‘a’, ’b’, ’c’, ’c’, ’c’, ’b’, ’a’, ’c’, ’a’, ’a’, ’b’, ’c’, ’b’, ’a’]

   Создать словарь, в котором ключи – это содержимое списка B, а значения для ключей словаря – это сумма всех элементов списка A в соответствии с буквой, содержащийся на той же позиции в списке B.

   Пример результата программы: {‘a’ : 10, ‘b’ : 15, ‘c’ : 6}.
5. Скачать и загрузить данные о стоимости домов в калифорнии, используя библиотеку sklearn.

```
from sklearn.datasets import fetch_california_housing
data = fetch_california_housing(as_frame=True)
```

6. Используя метод pd.concat([датафрейм1, датафрейм2], axis = 1), добавить к данным столбец, содержащий информацию о медианной стоимости дома (.target и .target_names).
7. Использовать метод info().
8. Узнать, есть ли пропущенные значения, используя isna().sum().
9. Вывести записи, где средний возраст домов в районе более 50 лет и население более 2500 человек, используя метод loc().
10. Узнать максимальное и минимальное значения медианной стоимости дома (max(), min()).
11. Используя метод apply(), вывести на экран название признака и его среднее значение.
12. Составить отчет о проделанной работе. В отчете должен быть представлен код и результаты его выполнения с выводами.