Ədəbiyyat

1.Эрик Вестра «Разработка геоприложений на языке Python» ДМК Пресс Москва. 2017
2.Любанович Билл İntroducing Python:Modern Computing in Simple Packages Питер 2019
3. Персиваль Гарри Test-Driven Development with Python ДМК Пресс 2018
4.Wes McKinney Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter 3rd Edition O'Reilly Media 2022
5. AMZ Press Python programming for Beginners Independently published 2022
6.Майк МакГрат Python программирование для начинающих Москва 2015
7.Kərim Tahiroğlu Python ilə proqramlaşdırma Bakı, "Şərq-Qərb" nəşriyyatı 2016
8. A. Qəhrəmanov, İlahə Cəfərova Python proqramlaşdırma dili. "Müəllim" nəşriyyatı 2015

ОБРАБОТКА ГЕОПРИЛОЖЕНИЙ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ PYTHON

Аскерова Тамилла Ахмед кызы , Канбарова Сахиба Ибрагим кызы

Резюме

В статье был рассмотрен процесс обработки геоданных на языке программирования python. Рассмотрены эффективные средства использования этого языка в картографии. Использовались библиотеки, созданные для этой цели. В качестве примера линия границы между Республикой Грузия и Азербайджанской Республикой показана с помощью графических объектов на языке программирования Python.

PROCESSING GEOAPPLICATIONS IN THE PYTHON PROGRAMMING LANGUAGE

Asgarova Tamilla Ahmed, Qanbarova Sahiba İbrahim

Summary

In the article, the process of geodata processing in the python programming language was considered. Effective means of using this language in mapping are considered. Libraries created for this purpose were used. As an example, the border line between the Republic of Georgia and the Republic of Azerbaijan is shown with the help of graphic objects in the Python programming language.

Rəyçi: İnformatika və cəbr kafedrasının dosenti Verdiyeva Gülmira Zahid qızı
İnformatika və cəbr kafedrasının 26 yanvar 2023 cü il tarixli iclasın 4 saylı protokolu
Daxil olma tarixi 27 yanvar 2023-cü il

UOT:372.0:002

PYHON DİLİNİN NUMPY KİTABXANASI

Babayeva Şəkər Maarif qızı, Əsgərova Tamilla Əhməd qızı
Gəncə Dövlət Universiteti
shakarbabayeva@mail.ru, asgerovatamilla998@mail.ru

Xülasə: *Python-da matrislərlə işləmək üçün bütün lazımi funksiyaları proqramlaşdırmaq olar. Proqramçıların işini asanlaşdırmaq üçün NumPy kitabxanası yaradılmışdır. O, mürəkkəb riyazi hesablamaları asanlıqla və səhvsiz yerinə yetirməyə imkan verir, proqramçını hər dəfə eyni kodu yazmaq məcburiyyətindən xilas edir.*

Cari məqalə Pyhon dilinin NumPy kitabxanasında massivlərin yaradılması üsulları, massivlərlə üzərində əməliyyatların aparılmasına həsr edilmişdir.

Açar sözlər: NumPy**,** çoxölçülü massivlər**,** matrislər**,** matrislərin yaradılması, matrislərlə işləmək üçün funksiyalar.

Ключевые слова: NumPy**,** многомерные массивы, матрицы**,** создание матриц, функции для работы с матрицами.

Key words: NumPy, multidimensional arrays, matrices**,** creating matrices**,** matrix functions.

Matris M sətir və N sütundan ibarət iki ölçülü massivdir. Matrislər tez-tez riyazi hesablamalarda istifadə olunur. Proqramçılar matrislərlə ilk növbədə elmi sahədə işləyirlər, lakin onlardan başqa məqsədlər, məsələn, video oyununda səviyyələrin sürətlə yaradılması üçün də istifadə oluna bilər.

Proqramçı matrislərlə işləmək üçün bütün funksiyaları (vurma, toplama, köçürmə və s.) sərbəst proqramlaşdıra bilər. Bunu Python-da etmək C kimi aşağı səviyyəli dillərə nisbətən daha asandır.

Amma hər dəfə eyni alqoritmləri yazmağın mənası yoxdur, ona görə də NumPy kitabxanası hazırlanmışdır. O, mürəkkəb elmi hesablamalar üçün istifadə olunur və proqramçıya ikiölçülü massivlərlə işləmək üçün hazır funksiyalar verir.

Sadə matris əməliyyatlarını yerinə yetirmək üçün onlarla kod sətirləri yazmaq əvəzinə, proqramçı NumPy-dan bir funksiyadan istifadə edə bilər. Kitabxana Python, C və Fortran dillərində yazılmışdır, buna görə də funksiyalar təmiz Python-dan daha sürətli işləyir.

NumPy (Numerical (rəqəmsal) Python) Python proqramlaşdırma dili üçün açıq mənbəli kitabxanadır. NumPy-yın İmkanları:

çoxölçülü massivlərə dəstək (matrislər daxil olmaqla);

çoxölçülü massivlərlə işləmək üçün nəzərdə tutulmuş yüksək səviyyəli riyazi funksiyalara dəstək.

NumPy MATLAB-a (texniki hesablamalarla bağlı məsələlərin həlli üçün nəzərdə tutulmuş tətbiqi proqramlar paketi) pulsuz alternativ kimi görünə bilər. MATLAB proqramlaşdırma dili səthi olaraq NumPy-ə bənzəyir: hər ikisi interpretasiya olunur, hər ikisi massivlər (matrislər) üzərində əməliyyatlar aparmağa imkan verir.

Həm MATLAB, həm də NumPy əsas xətti cəbr məsələlərini həll etmək üçün LAPACK kitabxana koduna əsaslanan koddan istifadə edir.

NumPy əvvəlcə SciPy kitabxanasının bir hissəsi idi. SciPy elmi və mühəndis hesablamalarını yerinə yetirmək üçün nəzərdə tutulmuş Python proqramlaşdırma dili üçün açıq mənbəli kitabxanadır. Digər layihələrin NumPy kitabxanasından istifadə etməsinə icazə vermək üçün onun kodu ayrıca paketə yerləşdirilib.

NumPy kitabxanasının qoşulması

NumPy Python interpretatoruna daxil edilməyib, ona görə də idxal etməzdən əvvəl quraşdırılmalıdır. Bunu etmək üçün pip yardım proqramından istifadə edə bilərsiniz. Konsolda əmri daxil edin: pip install numpy

Kitabxana quraşdırılandan sonra onu ***import*** əmrindən istifadə etməklə idxal etmək olar. Rahatlıq üçün modulu idxal edərkən ***numpy*** adını ***np***-yə aşağıdakı kimi dəyişdirəcəyik:

```
import numpy as np
```

NumPy-da matrislərin yaradılmasının bir neçə üsulu var. Ən asan üsul ***array()*** funksiyasının istifadəsilə yaradılmasıdır. Funksiyaya birinci arqument kimi iterasiya oluna bilən obyekt (siyahı, kortej) ötürülməlidir. Bu obyektin elementləri eyni uzunluqda olan və eyni tipli verilənləri ehtiva edən digər iterasiya oluna bilən obyekt olmalıdır. Məsələn, funksiyaya arqument kimi ikiölçülü siyahı verilir:

```
a = np.array([[3, 3, 3], [2, 5, 5]])
```

İkinci parametr matris elementlərinin növünü təyin etmək üçün istifadə edilə bilər:

```
a = np.array([[3, 3, 3],[2, 5, 5]], int)
print(a)
```

Nəticə kimi ekrana çıxacaq:

```
[[3 3 3]
```

[2 5 5]]

Nəzərə alın ki, ***int***-i ***str***-ə dəyişsəniz, element növü sətirə dəyişir. Həmçinin, ekrana xaric edərkən, NumPy avtomatik olaraq çıxışı elə formatlaşdırır ki, o, elementləri bir-birinin altına yığılmış matris kimi görünür.

Element tipləri kimi ***int, float, bool, complex, bytes, str, buffers*** istifadə etmək olar. Digər bəzi aşağıda verilən NumPy tiplərindən də istifadə etmək olar:

np.bool8 – 1 bayt yaddaş tutan məntiqi dəyişəndir.

np.int64 – 8 bayt tutan tam ədəddir.

np.uint16 – yaddaşda 2 bayt tutan işarəsiz tam ədəddir.

np.float32 – yaddaşda 4 bayt tutan həqiqi ədəddir.

np.complex64 – 4 baytlıq real hissədən və 4 baytlıq xəyali hissədən ibarət kompleks ədəddir.

NumPy-yın bütün tiplərini ***np.sctypeDict*** lüğətinin tərkibini ekrana çıxarmaqla görmək olar.

Ndarray obyektlərinin ən mühüm atributları bunlardır:

ndarray.ndim – massivin ölçülərinin sayıdır (daha çox "oxlar" adlanır).

ndarray.shape – massivin ölçüləri, onun formasıdır. Bu, hər bir ox boyunca massivin uzunluğunu göstərən natural ədədlər kortejidir, n sətir və m sütunlu matris üçün forma (n,m) olacaqdır. ***shape*** kortejinin elementlərinin sayı ***ndim*** -dir.

ndarray.size – massivin elementlərinin sayı. Aydındır ki, ***shape*** atributunun bütün elementlərinin hasilinə bərabərdir.

ndarray.dtype - massiv elementlərinin tipini təsvir edən obyektdir. Pythonun standart verilənlər tiplərindən istifadə edərək dtype təyin etmək mümkündür. NumPy burada daxili funksiyalarını, məsələn: bool_, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object_, istifadə etməyə və istifadəçiyə öz tiplərini təyin etməyə imkan verir.

ndarray.itemsize – hər bir massiv elementinin baytlarla ölçüsüdür.

shape atributundan istifadə edərək matrisin ölçüsünü (kortej şəklində) öyrənmək olar:

```
size = a.shape
print(size) # ekrana xaric ediləcək:  (2, 3)
```

Birinci rəqəm –sətirlərin sayı (2), ikinci rəqəm –sütunların sayıdır (3).

İkiölçülü matrisin sətrini əldə etmək üçün ona indekslə aşağıdakı kimi müraciət etmək kifayətdir:

```
temp = a[0]
print(temp)  # ekrana xaric ediləcək: [3 3 3]
```

Sütun əldə etmək artıq o qədər də asan deyil. Kəsiklərdən istifadə edərək, kəsiyin birinci elementi kimi heç nə göstərilmir, ikinci element kimi lazım olan sütunun nömrəsi göstərilir. Məsələn:

```
arr = np.array([[3,3,3],[2,5,5]], str)
temp = arr[:,2]
print(temp) # ekrana xaric ediləcək: ['3' '5']
```

Elementi əldə etmək üçün onun yerləşdiyi sütun və sətrin nömrələri göstərilir. Məsələn, 2-ci sətir və 3-cü sütundakı element 5-dir, yoxlayırıq (nömrələmənin 0-dan başladığını unutmuruq):

```
arr = np.array([[3,3,3],[2,5,5]], str)
element = arr[1][2]
print(element) # ekrana xaric ediləcək: 5
```

NumPy-da matrislərin yaradılmasının ikinci üsulu – ***eye(), zeros() və ones()*** daxili funksiyaların istifadəsilə yaradılmasıdır. Adətən massivin elementləri ilkin olaraq naməlum olur və onların saxlanacağı massiv artıq tələb olunur. Buna görə də bəzi ilkin məzmunlu

massivlər yaratmaq üçün bir neçə funksiya mövcuddur (susmaqla yaradılmış massivin növü ***float64*** -dür).

Birinci funksiya N×M vahid matrisi (diaqonal elementləri 1-ə, qalanları isə 0-a bərabər olan) yaradır (M verilmirsə, M = N):

```
b = np.eye(3)
print (b)
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

İkinci və üçüncü funksiyalar müvafiq olaraq yalnız sıfırlardan və ya birlərdən ibarət matrislər yaradır. Hər iki funksiyaya arqument kimi kortej şəklində matrisin ölçüləri və elementərinin tipi (***dtype)*** ötürülür (susmaqla ***float64*** -dür). Məsələn:

```
a_of_zeros = np.zeros((2,2))
print(a_of_zeros)
[[0. 0.]
  [0. 0.]]
a_of_ones = np.ones((2, 5))
print (a_of_ones )
[[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]
```

Üçüncü üsul – ***numpy.arange***([start,]stop, [addım,], ...) funksiyasından istifadə edir ki, bu da verilmiş step addımı ilə [start, stop) intervalından ardıcıl ədədlərin birölçülü massivini yaradır və massiv.reshape(shape) metodunun köməyilə bu birölçülü massivdən shape ölçülü matris yaradır.

shape parametri matrisin ölçüsünü (rəqəmlər korteji şəklində) təyin edir. Metodun işinin məntiqi aşağıdakı misaldan aydındır:

```
v = np.arange(0, 24, 2)
print ("Vektor-sütun:\n", v)
Vektor-sütun:
 [ 0  2  4  6  8 10 12 14 16 18 20 22]
d = v.reshape((3, 4))
print ("Matris:\n", d)
Matris:
 [[ 0  2  4  6]
 [ 8 10 12 14]
 [16 18 20 22]]
```

Ədədlərin ardıcıllığını yaratmaq üçün NumPy Python-un daxili ***range()*** funksiyasına oxşar ***arange()*** funksiyasına malikdir, ancaq siyahılar əvəzinə massivləri qaytarır və yalnız tam ədədləri qəbul etmir:

```
 np.arange(10, 30, 5)
 array([10, 15, 20, 25])
 np.arange(0, 1, 0.1)
 array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

Ümumiyyətlə, ***arange()*** funksiyasından ***float*** tipli arqumentləri ilə istifadə edərkən neçə elementin qaytarılacağına əmin olmaq çətindir (sürüşkən nöqtəli ədəglərin dəqiqliyi məhdudiyyətlərinə görə). Buna görə də, belə hallarda adətən addım əvəzinə arqumentlərdən biri kimi lazım olan elementlərin sayına bərabər olan ədəd qəbul edən ***linspace()*** funksiyasından istifadə etmək daha yaxşıdır:

```
 np.linspace(0, 2, 9)  # 0-dan 2-yə qədər 9 sayda ədəd (2 daxil olmaqla)
array([  0. ,  0.25,  0.5 ,  0.75, 1. , 1.25, 1.5 , 1.75, 2. ])
```

fromfunction() funksiyası müəyyən funksiyanı bütün indekslər kombinasiyasına tətbiq edərək matris yaradır. Məsələn:

```
def f1(i, j):
    return 3 * i + j
np.fromfunction(f1, (3, 4))
array([[ 0.,  1.,  2.,  3.],
       [ 3.,  4.,  5.,  6.],
       [ 6.,  7.,  8.,  9.]])
```

full() funksiyası bütün elementləri verilmiş bir qiymətə bərabər olan lazımi ölçülü matris yaradır. Məsələn:

```
d = np.full((2,2), 7)   # verilmiş qiymətlə (5) doldurulmuş (2, 2) ölçülü matris yaradır.
print(d)
[[ 7.  7.]
 [ 7.  7.]]
```

Matrisləri toplamaq üçün onların bütün uyğun elementləri toplanır. Python matrisləri toplamaq üçün adi "+" operatorundan istifadə edir. Məsələn:

```
arr1 = np.array([[3,3,3],[2,5,5]])
arr2 = np.array([[2,4,2],[1,3,8]])
cem = arr1 + arr2
print(cem)
```

Nəticədə aşağıdakı matris alınacaq:

```
[[ 5  7  5]
 [ 3  8 13]]
```

Yadda saxlamaq vacibdir ki, yalnız eyni sayda sətir və sütunlu matrislər toplanılır, əks halda Python proqramı ValueError istisnası ilə çıxacaq.

Matrisin vurulması toplamadan çox fərqlidir. Sadəcə olaraq, iki matrisin vurulması onların müvafiq elementlərini vurmaqla aparılmır. Birincisi, matrislərin birinin sütunlarının sayı digərinin sətirlərinin sayına bərabər və əksinə olmalıdır, əks halda proqram xətaya səbəb olacaqdır.

NumPy-da vurma ***dot()*** metodundan istifadə etməklə həyata keçirilir. Məsələn:

```
arr1 = np.array([[3,3],[2,5]])
arr2 = np.array([[2,4],[1,3]])
hasil = arr1.dot(arr2)
print(hasil)
```

Bu kodun icrası nəticəsi aşağıdakı kimi olacaq:

```
[[ 9 21]
 [ 9 23]]
```

Bu matris matrislərin vurması tərifinə uyğun alındı. Məsələn, 1 sətrin 2-ci sütunda yerləşən ədədi (21) almaq üçün 1-ci matrisin (arr1) 1-ci sətrini 2-ci matrisin (arr2) 2-ci sütununa vurulur (elementlər tutdiğu mövqeyə uyğun vurulur, yəni 1-ci 1-yə və 2-ci 2-yə və nəticələr toplanır: [3,3] * [4,3] = 3 * 4 + 3 * 3 = 21.

Transponirə edilmiş və tərs matrislər.

Transponirə edilmiş matris – sətirləri və sütunları yerlərilə dəyişdirilmiş matrisdir. NumPy kitabxanası ikiölçülü matrisləri transponirə etmək üçün ***transpose***() metodundan istifadə edir. Məsələn:

```
arr1 = np.array([[3,3],[2,5]])
transp = arr1.transpose()
print(transp)
```

Nəticədə aşağıdakı matris alınır:

```
[[3 2]
 [3 5]]
```

Tərs matrisi almaq üçün ***linalg*** (xətti cəbr) modulundan istifadə etmək lazımdır. Bu modulun ***inv()*** funksiyasından istifadə edilir:

```
arr1 = np.array([[3,3],[2,5]])
eks = np.linalg.inv(arr1)
print(eks)
```

Nəticədə aşağıdakı matris alınır:

```
[[ 0.55555556 -0.33333333]
 [-0.22222222  0.33333333]]
```

Matrisin maksimal və minimal elementlərinin alınması.

Maksimal və ya minimal elementi əldə etmək üçün iki ***for*** dövrlərdən istifadə edərək matrisin bütün elementləri nəzərdən keçirilir. Bu, demək olar ki, hər bir proqramçıya məlum olan alqoritmdir:

```
arr = np.array([[3,3],[2,5]])
min = arr[0][0]
for i in range(arr.shape[0]):
    for j in range(arr.shape[1]):
        if min > arr[i][j]:
            min = arr[i][j]
print("Minimal element:", min)  #  Minimal element: 2
```

NumPy ***amax()*** və ***amin()*** funksiyalarından istifadə edərək maksimal və minimal elementləri tapmağa imkan verir. Matrisin özü funksiyaya arqument kimi verilməlidir. Məsələn:

```
arr1 = np.array([[3,3],[2,5]])
min = np.amin(arr1)
max = np.amax(arr1)
print("Minimal element:", min)   #  Minimal element: 2
print("Maksimal element:", max)   #  Maksimal element: 5
```

Gördüyünüz kimi, Python-da tərtib edilmiş proqramın və NumPy kitabxanasının funksiyasından istifadə edilməsinin nəticələri eynidir.

Deməli, Python-da matrislərlə işləmək üçün bütün lazımi funksiyaları proqramlaşdırmaq olar. Proqramçıların işini asanlaşdırmaq üçün NumPy kitabxanası yaradılmışdır. O, mürəkkəb riyazi hesablamaları asanlıqla və səhvsiz yerinə yetirməyə imkan verir, proqramçını hər dəfə eyni kodu yazmaq məcburiyyətindən xilas edir.

NumPy-yın iştifadəsinə aid tam məlumatı almaq üçün aşağıdakı elektron ünvanda yerləşən zəsmi sənədlərə baxmaq olar:

aidhttps://numpy.org/doc/stable/reference/

Ədəbiyyat

1. S.T.Hüseynov “Python proqramlaşdırma dili elmi hesablamalarda”, Gəncə-2021.
2. Уэс МакКинни (пер. А.Слинкин) «Python и анализ данных», Изд: ДМК-Пресс, 2015.
3. https://pythonworld.ru/numpy/2.html
4. http://cs.mipt.ru/advanced_python/lessons/lab16.html
5. aidhttps://numpy.org/doc/stable/reference/
6. https://python-school.ru/blog/python-numpy-building/
7. https://www.internet-technologies.ru/articles/matricy-i-massivy-numpy-v-python.html
8. https://pythonist.ru/uchebnik-po-biblioteke-numpy-uchites-na-primerah/

БИБЛИОТЕКА NUMPY ЯЗЫКА PYTHON

Бабаева Шакяр Маарифовна, Аскерова Тамилла Ахмедовна

Резюме

На Python можно реализовать все необходимые функции для работы с матрицами. Чтобы упростить работу программистов, была создана библиотека NumPy. Она позволяет производить сложные математические вычисления легко и без ошибок, избавляя программиста от необходимости каждый раз писать один и тот же код.

PYTHON NUMPY LIBRARY

Babayeva Shakar Maarif, Asgerova Tamilla Ahmed

Summary

In Python, you can implement all the necessary functions for working with matrices. To simplify the work of programmers, the NumPy library was created. It allows you to perform complex mathematical calculations easily and without errors, saving the programmer from having to write the same code every time.

Rəyçi: dosent Aslanov Ə.Ə.
İnformatika və cəbr kafedrasının 26 yanvar 2023 cü il tarixli iclasın 4 saylı protokolu
Daxil olma tarixi 27 yanvar 2023-cü il

UOT:531

RESPUBLİKANIN ƏRZAQ PROQRAMININ HƏLLİNDƏ ÖRTÜLÜ ƏKİN SAHƏLƏRİNİN (İSTİXANA - PARNİK) MÜHÜM ƏHƏMİYYƏTİ

Dosent Yusifov Mustafa Qasım oğlu, dosent Cavadov Elçin Məhəmməd oğlu, t.ü.f.d.,Verdiyeva Təranə Zahid qızı, Xəlilova Aynur Mirzəağa qızı
Gəncə Dövlət Universiteti
elcin.55@bk.ru

Xülasə: *Məqalədə respublikanın ərazisində mövcud olan, örtülü torpaq sahələrində (daimi 5÷6 illik və müvəqqəti mövsümü bir illik) bostan-tərəvəz məhsularının yetişdirilməsində erkən alınan məhsulların iqtisadi əhəmiyyətindən bəhs olunur. İstehsal olunan tərəvəz məhsullarının respublikanın ərzaq proqramının həllində mühüm rol oynamaqla, həmin sahənin k/t-da bağçılıq və heyvandarlığında inkişafında mühüm əhəmiyyətindən bəhs olunur. Respublikada istehsal olunan qiymətli tərəvəz məhsulu istehsalının yarıdan çox hissəsi örtülü torpaq sahəlikdən alınmaqla onun ümumi kütləsi 1,2 mlrd tondan çoxdur. Tərəvələr yetişmə-istifadə müddətinə görə 1 illik, 2 illik və çoxillik uzunmüddətli olmaqla, onlar təzə, konservləşdirilmiş-duza qoyulmuş və qurudulmuş halda istifadə olunur.*

Açar sözlər:bostan-tərəvəz bitkilərinin bütün il boyu istehsalında əldə olunan məhsulun yarıdan çox hissəsi örtülü torpaq sahələrində yetişdirilirməklə yetişmə vaxtına görə 1-illik, 2 –illik və çoxillik olmaqla təbabətdə, qida məhsulu kimi, təzə, konservləşdirilmiş və qurudulmış halda istifadə edilir.

Ключевые слова: в республике больщепосовшины производимые овощные продукты вырфщиваются в закрытых почвенных условиях (теплицах – постоянных действующих в течении 5÷6 лет и временных – и сезонных временных) в зависимости от срока вегетации растений – 1-летных, 2-х летных и многолетных овощных культур. Производимые овощные культуры используются в медицине, пищеводстве и консервние производстве в свежем, консервированном (соленном) и сущенном виде.