|  |
| --- |
| **Sahifa 1** |

|  |
| --- |
| **2. sahifa** |

**UDC 004.6**

**BBC: 32.973**

Devpractice jamoasi. Pandalar. Ma'lumotlar bilan ishlash. 2-nashr. - devpractice.ru. 2020 yil. - 170 p.:

loy

Kitob pandalar ma'lumotlari bilan ishlash uchun kutubxonaga bag'ishlangan. Asosiy bilimlardan tashqari

pandalar tuzilmalari haqida siz vaqtinchalik bilan ishlash haqida ma'lumot olasiz

qatorlar, statistikalarni hisoblash, ma'lumotlarni tasavvur qilish va boshqalar

amaliyotga berilgan, ko'rib chiqilayotgan kutubxonaning barcha imkoniyatlari bilan birga

batafsil misollar.

UDC 004.6

BBC: 32.973

*Materiallar quyidagilar tomonidan tuzilgan va tayyorlangan:*

Abdraxmanov M.I.

Barcha huquqlar himoyalangan. Ushbu kitobning biron bir qismini ko'paytirish mumkin emas

qanday shaklda va qanday vositalar bilan yozilmasdan

mualliflik huquqi egalarining ruxsatnomalari.

Kitobda keltirilgan materiallar ko'p marta tekshirilgan. Ammo beri

inson va texnik xatolar hali ham mumkin, muallif va loyiha jamoasi

mumkin bo'lgan xatolar va oqibatlar uchun devpractice.ru javobgar emas,

ushbu kitobdan materiallardan foydalanish bilan bog'liq.

© devpractice.ru, 2020 yil

© Abdraxmanov M.I., 2020 yil

1

|  |
| --- |
| **3. sahifa** |

**Mundarija**

[Kirish ................................................. .................................................. ............................ besh](https://translate.googleusercontent.com/translate_f#6)

Nima [*pandalar*](https://translate.googleusercontent.com/translate_f#6)? ................................................. .................................................. ......... besh

*Pandalarni*[o'rnatish.................................................. .................................................. ......... 6](https://translate.googleusercontent.com/translate_f#7)

[1-bob.](https://translate.googleusercontent.com/translate_f#9)*Pandalardagi*[ma'lumotlar tuzilmalari:](https://translate.googleusercontent.com/translate_f#9)*Seriya*[va](https://translate.googleusercontent.com/translate_f#9)*DataFrame*[................................................. 8](https://translate.googleusercontent.com/translate_f#9)

[1.1 Ma'lumotlar](https://translate.googleusercontent.com/translate_f#9)*turkumi*[.................................................. ......................................... 8](https://translate.googleusercontent.com/translate_f#9)

[1.1.1 Yaratilish](https://translate.googleusercontent.com/translate_f#11)*Python*[ro'yxatidagi](https://translate.googleusercontent.com/translate_f#11)*seriyalar*[.................................................. ................. o'n](https://translate.googleusercontent.com/translate_f#11)

[1.1.2 Yaratish](https://translate.googleusercontent.com/translate_f#12)*Series* numpy dan ndarray qatori .......... [.................................. ... o'n bir](https://translate.googleusercontent.com/translate_f#12)

[1.1.3 Yaratish](https://translate.googleusercontent.com/translate_f#13)Lug'atdan *turkum* (diktat) ...... [...................................... ....................... 12](https://translate.googleusercontent.com/translate_f#13)

[1.1.4 Yaratilish Doimiy .............................................. dan foydalangan holda](https://translate.googleusercontent.com/translate_f#14)*ketma-ketlik.*[13](https://translate.googleusercontent.com/translate_f#14)

[1.1.5](https://translate.googleusercontent.com/translate_f#14)*Seriya*[elementlari bilan ishlash.................................................. .......................... 13](https://translate.googleusercontent.com/translate_f#14)

[1.2](https://translate.googleusercontent.com/translate_f#16)*DataFrame*[ma'lumotlar](https://translate.googleusercontent.com/translate_f#16)*tuzilishi*[.................................................. ............................... 15](https://translate.googleusercontent.com/translate_f#16)

[1.2.1 Yaratilish](https://translate.googleusercontent.com/translate_f#17)*Lug'atdagi DataFrame*[............................................... ...................... o'n oltita](https://translate.googleusercontent.com/translate_f#17)

[1.2.2 Yaratish](https://translate.googleusercontent.com/translate_f#18)*DataFrame*[lug'atlar ro'yxatidan .............................................. ........ 17](https://translate.googleusercontent.com/translate_f#18)

[1.2.3 Yaratish](https://translate.googleusercontent.com/translate_f#19)*DataFrame*[ikki o'lchovli qatordan .............................................. .18](https://translate.googleusercontent.com/translate_f#19)

[1.2.4](https://translate.googleusercontent.com/translate_f#20)*DataFrame elementlari*[bilan ishlash.................................................. .................. o'n to'qqiz](https://translate.googleusercontent.com/translate_f#20)

[2-bob.](https://translate.googleusercontent.com/translate_f#22)*Pandalar*[tarkibidagi ma'lumotlarga kirish.................................................. ............. 21](https://translate.googleusercontent.com/translate_f#22)

[2.1 Ma'lumotlardan foydalanishning ikkita yondashuvi](https://translate.googleusercontent.com/translate_f#22)*pandalar*[......................................... 21](https://translate.googleusercontent.com/translate_f#22)

[2.2 *Seriya* tuzilmasi ma'lumotlariga kirish ............................................ ............................. 23](https://translate.googleusercontent.com/translate_f#24)

[2.2.1 Teglar yordamida kirish ........................................... ........................... 23](https://translate.googleusercontent.com/translate_f#24)

[2.2.2 Butun sonli indekslar yordamida kirish .................................... 24](https://translate.googleusercontent.com/translate_f#25)

[2.2.3](https://translate.googleusercontent.com/translate_f#25)*Qo'ng'iroq qilish*[orqali kirishfunktsiyalari ................................................. ..24](https://translate.googleusercontent.com/translate_f#25)

[2.2.4 Mantiqiy ifoda yordamida kirish ... 25](https://translate.googleusercontent.com/translate_f#26)

[2.3 *DataFrame* ma'lumotlariga kirish ............................................ ..................... 25](https://translate.googleusercontent.com/translate_f#26)

[2.3.1 Teglar yordamida kirish ....................................... ........................... 25](https://translate.googleusercontent.com/translate_f#26)

[2.3.2](https://translate.googleusercontent.com/translate_f#27)*Callable*[yordamida kirishfunktsiyalari ................................................. ..26](https://translate.googleusercontent.com/translate_f#27)

[2.3.3 Mantiqiy ifoda yordamida kirish ... 26](https://translate.googleusercontent.com/translate_f#27)

[2.4 Ma'lumotlarga kirish uchun atributlardan foydalanish ........................................... ..... 27](https://translate.googleusercontent.com/translate_f#28)

[2.5 Tasodifiy tuzilmalar to'plamini olish](https://translate.googleusercontent.com/translate_f#29)*pandalar*[.............................................. 28](https://translate.googleusercontent.com/translate_f#29)

[2.6 Mantiqiy iboralar yordamida indeksatsiya qilish ....................................... 30](https://translate.googleusercontent.com/translate_f#31)

2.7 *Pandalardagi*[ma'lumotlar bilan ishlashda](https://translate.googleusercontent.com/translate_f#33) isin-dan foydalanish[......................................... 32](https://translate.googleusercontent.com/translate_f#33)

[3-bob. Ma'lumot turlari](https://translate.googleusercontent.com/translate_f#35)*pandalar*[................................................. ....................................... 34](https://translate.googleusercontent.com/translate_f#35)

[3.1 Ma'lumot turlari ............................................... .................................................. ............ 35](https://translate.googleusercontent.com/translate_f#36)

[3.2 Turlar bilan ishlash vositalari ............................................ ............................ 37](https://translate.googleusercontent.com/translate_f#38)

[3.2.1 astype () ............................................ .................................................. ................. 37](https://translate.googleusercontent.com/translate_f#38)

[3.2.2 Ma'lumotlarni tayyorlash funktsiyalari ............................................ ................................ 39](https://translate.googleusercontent.com/translate_f#40)

[3.2.3 Yordamchi funktsiyalar ............................................. ................................. 40](https://translate.googleusercontent.com/translate_f#41)

[3.2.4 Ma'lumotlarni namuna olish turlari bo'yicha ........................................... ....................................... 41](https://translate.googleusercontent.com/translate_f#42)

3.3 Kategorik turlari ............. [.................................. ............................................... 42](https://translate.googleusercontent.com/translate_f#43)

[3.3.1 Kategorik ma'lumotlar to'plami bilan tuzilmani yaratish ............................. 42](https://translate.googleusercontent.com/translate_f#43)

[3.3.2 Buyurtma toifalari ............................................. ........................................ 46](https://translate.googleusercontent.com/translate_f#47)

[4-bob. Ma'lumotlardagi bo'shliqlar bilan ishlash .......................................... ................................ 48](https://translate.googleusercontent.com/translate_f#49)

4.1 *Pandalar*[va etishmayotgan ma'lumotlar ............................................... ............................ 48](https://translate.googleusercontent.com/translate_f#49)

2018-04-02 121 2

|  |
| --- |
| **4. sahifa** |

4.2 [Yo'qotilgan ma'lumotlarni almashtirish .............................................. ................................ 50](https://translate.googleusercontent.com/translate_f#51)

4.3 [Yo'qotilgan ma'lumotlar bilan ob'ektlarni / ustunlarni o'chirish ................................. 52](https://translate.googleusercontent.com/translate_f#53)

[5-bob.](https://translate.googleusercontent.com/translate_f#55)*Pandalardagi*[ma'lumotlar tuzilmalari bilan ishlash: o'chirish, birlashtirish,](https://translate.googleusercontent.com/translate_f#55)

[kengaytirish, guruhlash ............................................... .................................................. .54](https://translate.googleusercontent.com/translate_f#55)

[5.1](https://translate.googleusercontent.com/translate_f#55)*Pandalar*[tuzilishiga elementlar qo'shish.................................................. ........ 54](https://translate.googleusercontent.com/translate_f#55)

[5.1.1](https://translate.googleusercontent.com/translate_f#56)*Seriyalarga*[qo'shish.................................................. ....................................... 55](https://translate.googleusercontent.com/translate_f#56)

[5.1.2](https://translate.googleusercontent.com/translate_f#56)*DataFrame-ga*[qo'shilish.................................................. ............................... 55](https://translate.googleusercontent.com/translate_f#56)

[5.2](https://translate.googleusercontent.com/translate_f#58)*Pandalardagi*[strukturadan elementlarni olib tashlash.................................................. ....... 57](https://translate.googleusercontent.com/translate_f#58)

[5.2.1](https://translate.googleusercontent.com/translate_f#58)*Seriyadan*[olib tashlash.................................................. ......................................... 57](https://translate.googleusercontent.com/translate_f#58)

[5.2.2](https://translate.googleusercontent.com/translate_f#59)*DataFrame-*[dan o'chirish.................................................. .................................. 58](https://translate.googleusercontent.com/translate_f#59)

[5.3 Ma'lumotlarning birlashishi ............................................... ............................................... 60](https://translate.googleusercontent.com/translate_f#61)

[5.3.1](https://translate.googleusercontent.com/translate_f#61)*Konkak*[usulidan foydalanish.................................................. ....................... 60](https://translate.googleusercontent.com/translate_f#61)

[5.3.2](https://translate.googleusercontent.com/translate_f#65)*Ma'lumotlar bazasi uslubidan*[foydalanishyondashuv ................................................. ......... 64](https://translate.googleusercontent.com/translate_f#65)

[6-bob. Ma'lumotlarning tashqi manbalari bilan ishlash .......................................... ............... 69](https://translate.googleusercontent.com/translate_f#70)

[6.1](https://translate.googleusercontent.com/translate_f#70)*CSV*[formatidagi ma'lumotlar bilan ishlash.................................................. ...................... 69](https://translate.googleusercontent.com/translate_f#70)

[6.1.1 Ma'lumotlarni o'qish ............................................. .................................................. ... 69](https://translate.googleusercontent.com/translate_f#70)

[6.1.2 Ma'lumotlarni yozib olish ............................................. .................................................. ... 72](https://translate.googleusercontent.com/translate_f#73)

[6.2](https://translate.googleusercontent.com/translate_f#74)*JSON*[ma'lumotlari bilan ishlash.................................................. .................... 73](https://translate.googleusercontent.com/translate_f#74)

[6.2.1 Ma'lumotlarni o'qish ............................................. .................................................. ... 73](https://translate.googleusercontent.com/translate_f#74)

[6.2.2 Ma'lumotlarni ro'yxatga olish ............................................. .................................................. ... 77](https://translate.googleusercontent.com/translate_f#78)

[6.3 bilan ishlash](https://translate.googleusercontent.com/translate_f#80)*Excel*[fayllari ................................................ .......................................... 79](https://translate.googleusercontent.com/translate_f#80)

[6.3.1 Ma'lumotlarni o'qish ............................................. .................................................. ... 79](https://translate.googleusercontent.com/translate_f#80)

[6.3.2 Ma'lumotlarni ro'yxatga olish ............................................. .................................................. ... 82](https://translate.googleusercontent.com/translate_f#83)

[7-bob. Ma'lumotlar bilan ishlash ............................................ .......................................... 84](https://translate.googleusercontent.com/translate_f#85)

[7.1 Arifmetik amallar ............................................... ...................................... 84](https://translate.googleusercontent.com/translate_f#85)

[7.2 Mantiqiy amallar ............................................... ................................................ 86](https://translate.googleusercontent.com/translate_f#87)

[7.3 Statistika ................................................ .................................................. .............. 89](https://translate.googleusercontent.com/translate_f#90)

[7.4 Funktsional kengayish ............................................... .................................. 93](https://translate.googleusercontent.com/translate_f#94)

[7.4.1 Ma'lumotlarni qayta ishlash oqimi ............................................ .............................. 93](https://translate.googleusercontent.com/translate_f#94)

[7.4.2 Funktsiyani satr yoki ustun elementlariga qo'llash ................................. 94](https://translate.googleusercontent.com/translate_f#95)

[7.4.3 Birlashtirish (](https://translate.googleusercontent.com/translate_f#97)*API*[) ................................................. ................................................ 96](https://translate.googleusercontent.com/translate_f#97)

[7.4.4 Ma'lumotlarni o'zgartirish ............................................. .............................. 98](https://translate.googleusercontent.com/translate_f#99)

[7.5](https://translate.googleusercontent.com/translate_f#100)*str*[tipidagi usullardan foydalanishmatnli ma'lumotlar bilan ishlash uchun ................ 99](https://translate.googleusercontent.com/translate_f#100)

[8-bob.](https://translate.googleusercontent.com/translate_f#103)*Pandalarni*[sozlash.................................................. ............................................ 102](https://translate.googleusercontent.com/translate_f#103)

8. [1](https://translate.googleusercontent.com/translate_f#103)*APIpandalar*[sozlamalari bilan ishlash.................................................. .............. 102](https://translate.googleusercontent.com/translate_f#103)

8.2 [Kutubxona sozlamalari](https://translate.googleusercontent.com/translate_f#107)*pandalar*[................................................. ............................. 106](https://translate.googleusercontent.com/translate_f#107)

[9-bob. Ma'lumotlar bilan ishlash vositalari .......................................... .................... 109](https://translate.googleusercontent.com/translate_f#110)

[9.1 Sürgülü oyna. Statistika ................................................. .............................. 109](https://translate.googleusercontent.com/translate_f#110)

[9.2 Oynani kengaytirish. Statistika ................................................. ...................... 116](https://translate.googleusercontent.com/translate_f#117)

[9.3 Vaqtga yo'naltirilgan slayd ............................................. ..................... 120](https://translate.googleusercontent.com/translate_f#121)

[9.4 Ma'lumotlarni birlashtirish ............................................... .................................................. .122](https://translate.googleusercontent.com/translate_f#123)

[10-bob. Vaqt qatorlari ............................................. ................................................ 124](https://translate.googleusercontent.com/translate_f#125)

[10.1 Vaqt shtamplari bilan ishlash ............................................. ............................. 125](https://translate.googleusercontent.com/translate_f#126)

[10.1.1 Vaqt shtampini yaratish ............................................ ............................. 125](https://translate.googleusercontent.com/translate_f#126)

3

|  |
| --- |
| **Sahifa 5** |

[10.1.2 Vaqt markalari seriyasini yaratish ........................................... ..................... 127](https://translate.googleusercontent.com/translate_f#128)

[10.2 Vaqt intervallari bilan ishlash ............................................. ..................... 132](https://translate.googleusercontent.com/translate_f#133)

[10.2.1 Vaqt oralig'ini yaratish ............................................ .................... 132](https://translate.googleusercontent.com/translate_f#133)

[10.2.2 Vaqt oralig'ini yaratish ........................................... ........... 134](https://translate.googleusercontent.com/translate_f#135)

[10.3 Vaqt qatorlarini indeks sifatida ishlatish ................................................................................. 136](https://translate.googleusercontent.com/translate_f#137)

[11-bob. Ma'lumotlarni vizuallashtirish ............................................. ........................................ 139](https://translate.googleusercontent.com/translate_f#140)

[11.1 Grafika qurilishi ............................................... ......................................... 139](https://translate.googleusercontent.com/translate_f#140)

[11.1.1 Chiziqli diagrammalar ............................................. ......................................... 141](https://translate.googleusercontent.com/translate_f#142)

[11.1.2 Shtrixli jadvallar ............................................. .................................. 143](https://translate.googleusercontent.com/translate_f#144)

[11.1.3 Gistogrammalar .............................................. .................................................. .146](https://translate.googleusercontent.com/translate_f#147)

[11.1.4 To'ldirilgan grafik ............................................ ............................................. 147](https://translate.googleusercontent.com/translate_f#148)

[11.1.5 nuqta uchastkasi ............................................. .............................................. 149](https://translate.googleusercontent.com/translate_f#150)

[11.1.6 Dairesel diagrammasi ............................................. ........................................ 150](https://translate.googleusercontent.com/translate_f#151)

[11.1.7 olti burchakli diagramma ............................................ ...................... 152](https://translate.googleusercontent.com/translate_f#153)

[11.2 Diagrammaning ko'rinishini sozlash ............................................. .................. 153](https://translate.googleusercontent.com/translate_f#154)

[11.2.1 Chiziqli diagrammaning ko'rinishini sozlash .......................................... 153](https://translate.googleusercontent.com/translate_f#154)

[11.2.2 Grafiklarni turli tekisliklarda aks ettirish .......................................... ............ 157](https://translate.googleusercontent.com/translate_f#158)

[12-bob. Jadvallarning ko'rinishini sozlash ........................................... ....................... 159](https://translate.googleusercontent.com/translate_f#160)

[12.1 Ma'lumotlarni taqdim etish formatini o'zgartirish ............................................. ....... 161](https://translate.googleusercontent.com/translate_f#162)

[12.2 O'z uslublaringizni yaratish .............................................. ............................. 163](https://translate.googleusercontent.com/translate_f#164)

[12.2.1 Ma'lumot elementlari uchun yorliq rangini o'rnatish ........................................ 163](https://translate.googleusercontent.com/translate_f#164)

[12.2.2 Jadval katakchasining rangini belgilash ........................................... ........................ 164](https://translate.googleusercontent.com/translate_f#165)

[12.2.3 Jadval qatorining rangini belgilash ........................................... ........................ 165](https://translate.googleusercontent.com/translate_f#166)

12.3 Ichki uslublar vositalari ............... [.............................. ........... 165](https://translate.googleusercontent.com/translate_f#166)

[12.3.1 Minimal va maksimal qiymatlarni yoritish .............................. 166](https://translate.googleusercontent.com/translate_f#167)

[12.3.2 Nol elementlarni ajratib ko'rsatish ............................................ ................................ 167](https://translate.googleusercontent.com/translate_f#168)

[12.3.3 Issiqlik xaritasini o'rnatish ............................................ .................................. 168](https://translate.googleusercontent.com/translate_f#169)

[12.3.4 Ustunlar jadvalini qoplash ... ................ 168](https://translate.googleusercontent.com/translate_f#169)

[12.3.5](https://translate.googleusercontent.com/translate_f#170) Zanjir hisob-kitoblari ( *Chaining Method*) [ko'rinishini sozlash uchun](https://translate.googleusercontent.com/translate_f#170)

[jadvallar ................................................. .................................................. ................. 169](https://translate.googleusercontent.com/translate_f#170)

[Xulosa ................................................. .................................................. .................... 170](https://translate.googleusercontent.com/translate_f#171)

4

|  |
| --- |
| **6-bet** |

**Kirish**

**Nima *pandas* ?**

*Pandas* - bu juda qulay bo'lgan, kutubxonadir

foydalanish nuqtai nazaridan, ma'lumotlarni saqlash va ular bilan ishlash vositalari.

Agar siz ma'lumotlarni tahlil qilish yoki mashinani o'rganish bilan shug'ullansangiz va

agar siz *Pythondan* foydalanayotgan bo'lsangiz , unda *pandalarni* bilish juda soddalashtiriladi

siz uchun ishlang.

*Pandalar numfocus* tomonidan homiylik qilingan loyihalar guruhining bir qismidir

([*https://www.numfocus.org/*](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=https://www.numfocus.org/) ). *Numfocus*bu tashkilot

turli xil loyihalar va ilmiy dasturlarni qo'llab-quvvatlaydi

mavzular.

Rasmiy *pandas* sayti bu erda [*http://pandas.pydata.org/*](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=http://pandas.pydata.org/) .

Shuni ta'kidlash kerakki, ushbu mahsulot yaxshi hujjatlarga ega, siz buni qila olasiz

havolani ([*http://pandas.pydata.org/pandas-docs/stable/*](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=http://pandas.pydata.org/pandas-docs/stable/)) ga

u bilan tanishib chiqing.

*Pandalarning* o'ziga xos xususiyati shundaki, bu kutubxona juda tez va

moslashuvchan va shu sababli u *Python* bilan ishlatiladi , bu farq qilmaydi

yuqori ishlash, tezlik masalasi, ishlashda

katta miqdordagi ma'lumotlar kalitlardan biriga aylanadi. *Pandalar*

1D va 2D bilan ishlash uchun juda yaxshi

tashqi jadval bilan yaxshi birlashtirilgan ma'lumotlar jadvallari: ha

*CSV* fayllari , *Excel* jadvallari bilan ishlash qobiliyati , mumkin

*R* tili bilan bog'lang .

besh

|  |
| --- |
| **Sahifa 7** |

***Pandalarni*o'rnatish**

Ilmiy hisob-kitoblar, ma'lumotlarni tahlil qilish va qurish uchun

*Python tili* uchun mashinani o'rganish doirasidagi modellar mavjud

mukammal echim *Anaconda* . *Anakonda* - bu o'z ichiga olgan paket

o'zida turli xil kutubxonalarning katta to'plami, til tarjimoni

*Python* va bir nechta rivojlanish muhiti.

*Pandalar Anaconda* bilan standart *keladi* . Agar u erda bo'lsa

yo'q, uni alohida o'rnatishingiz mumkin. Buning uchun siz foydalanishingiz kerak

*Anaconda* bilan birga keladigan paket menejeri , bu

*konda* deb nomlangan . Uni ishga tushirish uchun katalogga o'tishingiz kerak

*\ \ Scripts [Anaconda yo'l o'rnatish]* ustiga *Windows* . *Linux* operatsion tizimida ,

*Anaconda-* ni o'rnatgandan so'ng *conda* menejeri bo'lishi kerak

hamma joyda.

Buyruqning satrini kiriting:

conda o'rnatish pandalari

Agar *pandalarning* ma'lum bir versiyasi kerak bo'lsa , u holda uni ko'rsatish mumkin

o'rnatish paytida:

conda install pandas = 0.13.1

Agar kerak bo'lsa, siz *pip* to'plami menejeridan foydalanishingiz mumkin ,

*Python* tarqatilishiga kiritilgan :

pip o'rnatish pandalari

6

|  |
| --- |
| **8-sahifa** |

Agar siz *Linuxdan* foydalanayotgan bo'lsangiz , unda *pandalarni* o'rnatishning yana bir usuli bor -

bu operatsiya xonasining paket menejeridan o'zi foydalanish

tizimlar. Uchun *Ubuntu,* bu kabi ko'rinadi:

sudo apt-get install python-pandas

O'rnatishdan so'ng, *pandalar* o'rnatilganligini tekshirishingiz kerak

to'g'ri ishlaydi. Buning uchun *Python* tarjimonini ishga tushiring va yozing

unda quyidagi buyruqlar mavjud (>>> belgilar majmui kiritilishi shart emas, bu shunday

*Python* tarjimoni so'rovi ):

>>> pandalarni pd **sifatida import qilish**

>>> pd.test ()

Natijada terminal oynasida quyidagi matn paydo bo'lishi kerak:

Pandalar uchun ishlaydigan birlik sinovlari

pandalar 0.18.1 versiyasi

numpy versiyasi 1.11.1

pandalar c: \ Anaconda3 \ lib \ site-packages \ pandas-ga o'rnatilgan

Python versiyasi 3.5.2 | Anaconda 4.1.1 (64-bit) | (sukut bo'yicha, 2016 yil 5-iyul,

11:41:13) [MSC v.1900 64 bit (AMD64)]

burun versiyasi 1.3.7

..........

-------------------------------------------------- -----------

0.422-larda 11 ta sinov

OK

Bu shuni anglatadiki, *pandalar* o'rnatilgan va ulardan foydalanish mumkin.

7

|  |
| --- |
| **9-bet** |

**1-bob. *Pandalardagi* ma'lumotlar tuzilmalari : *ketma-ket* va**

***DataFrame***

*Pandas* kutubxonasi ikkita tuzilmani taqdim etadi: *Series* va *DataFrame*

ma'lumotlar bilan tez va qulay ishlash uchun (aslida ulardan uchtasi bor, bor

yana bir tuzilma *Panel* , ammo hozirgi paytda u maqomga ega

*eskirgan* va kelajakda *pandalar* kutubxonasidan olib tashlanadi ).

*Seriya* - bu bo'lishi mumkin bo'lgan bir o'lchovli ma'lumotlar tuzilishi

bitta qatorli (yoki ustunli) jadval sifatida tasavvur qiling. S *seriyasi*

siz oddiy qator bilan ishlashingiz mumkin (raqamga qarang

indeks) va tegishli qatorda bo'lgani kabi, bu holda ham mumkin

ma'lumotlar elementlariga kirish uchun kalitdan foydalaning.

*DataFrame* - ikki o'lchovli etiketli tuzilma. Ideal holda, u juda yaxshi

odatdagi jadvalga o'xshaydi, u uni yaratish uslubida va

uning elementlari bilan ishlash.

*Panel* - bu uch o'lchovli ma'lumotlar tuzilishi. *Panel* haqida biz

biz ko'proq aytmaymiz. Ushbu qismda biz to'xtalamiz

tuzilmalarning ma'lumotlar elementlarini yaratish va ulardan foydalanish masalalari

*Series* va *DataFrame* .

**1.1 Ma'lumotlar *turkumi***

*Pandalar* ma'lumotlar tuzilmalari bilan ishlashni boshlash uchun

avval tegishli modulni import qilishingiz kerak.

*Pandalar* kutubxonasi sizning kompyuteringizda o'rnatilganligiga ishonch hosil qiling , oh

buni qanday qilishni kitobning kirish qismida o'qish mumkin. Biz ham qilamiz

*Python tili* bilan tanish ekanligingizni taxmin qiling .

8

|  |
| --- |
| **10-bet** |

*Pandalarning* o'zi bilan bir qatorda, biz *tinglash* kutubxonasiga muhtojmiz . Uchun

tajribalar, biz interaktiv *IPython* qobig'ini tavsiya qilamiz, mumkin

mustaqil dastur sifatida ishga tushirish yoki ba'zilari ichida ishlatish

ba'zi *IDE* ( *Spyder* , *PyCharm* ) yoki IDLE-dan foydalaning, ammo unday emas

grafikalar bilan ishlash uchun mos.

Agar kod satrlarida >>> prefiksi bo'lsa, unda bu

bu buyruqlarni interaktiv qobiqqa kiritamiz degan ma'noni anglatadi

aks holda, bu kod muharrirda yozilganligini anglatadi.

Amaliyotga kirish vaqti keldi! Birinchidan, biz import qilamiz

kutubxonalar:

>>> pandalarni pd **sifatida import qilish**

>>> numpy- **ni** np **sifatida import qilish**

Siz quyidagi ma'lumotlar turlariga asoslanib *ketma-ket* tuzilmani yaratishingiz mumkin:

- lug'at (diktat) *Python* ;

- ro'yxat (ro'yxat) *Python* ;

- ndarray qatori (numpy kutubxonasidan);

skalar qiymati.

*Seriya* konstruktori quyidagicha ko'rinadi:

Seriya (ma'lumotlar = **Yo'q** , indeks = **Yo'q** , dip = **Yo'q** , ism = **Yo'q** , nusxa = **Yolg'on** ,

fastpath = **False** )

to'qqiz

|  |
| --- |
| **11-bet** |

Keling, ba'zi konstruktor parametrlarini tavsiflaylik:

• ma'lumotlar *:* massiv, skaler qiymat, dikt; standart qiymat:

Yo'q

*Series Seriya* quriladigan inshoot *.*

• indeks: bir o'lchovli massiv; standart qiymat: Yo'q

Access Kirish uchun ishlatiladigan teglar ro'yxati

*Seriya* elementlari . Ro'yxat uzunligi ma'lumotlar uzunligiga teng bo'lishi kerak.

• dtype *:* tovushsiz *.*dip turi; standart qiymat: Yo'q

◦ ma'lumotlar turini belgilaydigan ob'ekt.

• nusxa ko'chirish *:* bool; standart qiymat: noto'g'ri

◦ Agar parametr rost bo'lsa, massivning nusxasi yaratiladi

ma'lumotlar.

Ko'pgina hollarda, Seriyani yaratishda faqat foydalaning

dastlabki ikkita parametr.

**1.1.1 *Python* ro'yxatidan *seriya* yaratish**

*Seriya* yaratishning eng oson usuli bu o'tish

konstruktorning yagona parametri bu *Python* ro'yxati :

>>> s1 = pd seriyalari ([ 1 , 2 , 3 , 4 , 5 ])

>>> s1

0 1

12

2 3

3 4

4 5

dip turi: int64

o'n

|  |
| --- |
| **Sahifa 12** |

Misolda, *ketma-ket* tuzilish tildan olingan ro'yxat asosida yaratilgan

*Python* . *Seriya* elementlariga kirish uchun bu holda siz qilishingiz mumkin

faqat musbat butun sonlardan foydalaning - chap ustun

noldan boshlanadigan raqamlar elementlarning aniq ko'rsatkichlari

o'ng ustunda ko'rsatilgan tuzilmalar.

Iplar ro'yxatini ikkinchi element sifatida o'tkazamiz (bizning holatlarimizda -

bu alohida belgilar). Bu elementlarga kirishga imkon beradi

*Series* strukturasi nafaqat raqamli indeks, balki yorliq bilan ham

lug'at bilan ishlashga o'xshash bunday ob'ekt bilan ishlashni amalga oshiradi:

>>> s2 = pd.Seriyalar ([ 1 , 2 , 3 , 4 , 5 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

>>> chop etish (s2)

a 1

b 2

v 3

d 4

e 5

dip turi: int64

Chap ustunga e'tibor bering, unda yorliqlar mavjud

biz strukturani yaratishda uni indeks *-* parametr sifatida o'tkazdik .

O'ng ustun hali ham tuzilish a'zolari.

**1.1.2 Numpy dan ndarray massividan *seriyalar* yarating**

Ro'yxatiga o'xshash beshta raqamdan iborat ndarray qatorini yarating

oldingi bo'lim. Pandalar va tovushsiz kutubxonalar bo'lishi kerak

oldindan import qilingan:

>>> ndarr = np.array ([ 1 , 2 , 3 , 4 , 5 ])

>>> turi (ndarr)

< **class 'numpy.ndarray'** >

o'n bir

|  |
| --- |
| **13-bet** |

Keling, harf yorliqlari bilan bir *qator* yarataylik :

>>> s3 = pd.Series (ndarr, [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

>>> chop etish (s3)

a 1

b 2

v 3

d 4

e 5

dtype: int32

Ko'rib turganingizdek, natija siz olgan narsaga o'xshaydi

*Python* ro'yxatlaridan foydalanish .

**1.1.3 Lug'atdan *turkum* yaratish (diktat)**

*Series* strukturasini yaratishning yana bir usuli bu lug'atdan foydalanish

yorliqlar va qiymatlarni bir vaqtning o'zida o'rnatish uchun:

>>> d = { **'a'** : 1 , **'b'** : 2 , **'c'** : 3 }

>>> s4 = pd.Seriyalar (d)

>>> chop etish (s4)

a 1

b 2

v 3

dip turi: int64

D lug'atidagi kalitlar s4 yorlig'i va qiymatlari bo'ladi

lug'at (qiymatlar) - qadriyatlar.

12

|  |
| --- |
| **14-bet** |

**1.1.4 Doimiy yordamida *ketma-ketlik* yaratish**

Keling, *Seriya* yaratishning yana bir usulini ko'rib chiqaylik . Bu safargi qiymatlar

tuzilish hujayralari bir xil bo'ladi:

>>> a = 7

>>> s5 = pd.Series (a, [ **'a'** , **'b'** , **'c'** ])

>>> chop etish (s5)

a 7

b 7

v 7

dip turi: int64

Olingan strukturada bir xil bo'lgan uchta element mavjud

tarkib.

**1.1.5 *Seriya*elementlari bilan ishlash**

Indexing va *Series* va *DataFrame* elementlari bilan ishlashga bag'ishlanadi

alohida bob, endi biz asosiy yondashuvlarni ko'rib chiqamiz

*pandalar* beradi .

*Seriyalar* elementlariga qachon raqamli indeks orqali kirish mumkin

ushbu yondashuvda struktura bilan ishlash ishlashdan farq qilmaydi

*Python-* dagi ro'yxatlar *:*

>>> s6 = pd.Seriyalar ([ 1 , 2 , 3 , 4 , 5 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

>>> s6 [ 2 ]

3

Siz yorliqdan foydalanishingiz mumkin, keyin *Series* bilan ishlash ishlashga o'xshaydi

*Pythonda* lug'at (dikt) bilan *:*

>>> s6 [ **'d'** ]

4

13

|  |
| --- |
| **Sahifa 15** |

Dilim bilan ishlash uchun sintaksis mavjud:

>>> s6 [: 2 ]

a 1

b 2

dip turi: int64

Indeks uchun maydonga shartli ifodani qo'yishingiz mumkin:

>>> s6 [s6 <= 3 ]

a 1

b 2

v 3

dip turi: int64

Siz *Series* tuzilmalari bilan vektorlarda bo'lgani kabi ishlashingiz mumkin: add,

vektorni raqamga ko'paytirish va hk.:

>>> s7 = pd.Seriyalar ([ 10 , 20 , 30 , 40 , 50 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

Tuzilmalar qo'shilganda ularning elementlari bir-biriga qo'shiladi:

>>> s6 + s7

a 11

b 22

v 33

g 44

e 55

dip turi: int64

o'n to'rt

|  |
| --- |
| **16-bet** |

Strukturani raqamga ko'paytirganda, strukturaning barcha elementlari

berilgan omilga ko'paytiriladi:

>>> s6 \* 3

a 3

b 6

v 9

d 12

e 15

dip turi: int64

**1.2 *DataFrame*ma'lumotlar *tuzilishi***

Agar *Series* bu bir o'lchovli tuzilma bo'lsa

siz o'zingizni bir qatorli jadval sifatida tasavvur qilishingiz mumkin, keyin *DataFrame* -

bu allaqachon ikki o'lchovli tuzilish - ko'p qatorli to'liq stol

va ustunlar.

*DataFrame* sinfining konstruktori quyidagicha ko'rinadi:

DataFrame (ma'lumotlar = **Hech** , indeks = **hech** , ustunlar = **hech** , dtype = **hech** , nusxa = **Soxta** )

Konstruktor parametrlari:

• ma'lumotlar: ndarray, dict yoki *DataFrame* ; standart qiymat: Yo'q

◦ *DataFrame* yaratiladigan ma'lumotlar .

• indeks *:* bir o'lchovli massiv; standart qiymat: Yo'q

Records yozuvlar yorliqlari ro'yxati (jadval satrlari nomlari).

• ustunlar *:* bir o'lchovli massiv; sukut bo'yicha: yo'q

Of Dala yorliqlari ro'yxati (jadval ustunlari nomlari).

• dtype *:* tovushsiz *.*dip turi; standart qiymat: Yo'q

◦ ma'lumotlar turini belgilaydigan ob'ekt.

15

|  |
| --- |
| **Sahifa 17** |

• nusxa ko'chirish *:* bool; standart qiymat: noto'g'ri

◦ Agar parametr rost bo'lsa, massivning nusxasi yaratiladi

ma'lumotlar.

A *DataFrame* quyidagi ma'lumotlar turlari asosida tashkil etilishi mumkin:

• lug'at (dikt), uning elementlari quyidagilar bo'lishi mumkin:

bir o'lchovli ndarray, ro'yxatlar, boshqa lug'atlar, *ketma-ket* tuzilmalar ;

• ikki o'lchovli ndarray;

• *ketma-ket* tuzilishi ;

• boshqa *DataFrame* .

Keling, amalda *DataFrames* yaratishning turli xil yondashuvlarini ko'rib chiqamiz .

**1.2.1 bir yaratish *DataFrame* bir Lug'atga dan**

*DataFrame* yaratish uchun biz lug'at, elementlardan foydalanamiz

ro'yxatlar, *ketma-ket* tuzilmalar va boshqalar bo'lishi mumkin . Variant bilan boshlaymiz,

elementlar *ketma-ket* tuzilmalar bo'lganda *:*

>>> d = { **'narx'** : pd.Series ([ 1 , 2 , 3 ], index = [ **'v1'** , **'v2'** , **'v3'** ]), **'count'** :

pd.Series ([ 10 , 12 , 7 ], index = [ **'v1'** , **'v2'** , **'v3'** ])}

>>> df1 = pd.DataFrame (d)

>>> chop etish (df1)

narxlar soni

v1 1 10

v2 2 12

v3 3 7

o'n olti

|  |
| --- |
| **18-bet** |

Yaratilgan *DataFrame* indekslari :

>>> chop etish (df1.index)

Indeks ([ **'v1'** , **'v2'** , **'v3'** ], dtype = **'ob'ekt'** )

Yaratilgan *DataFrame* ustunlari :

>>> chop etish (df1. ustunlar)

Indeks ([ **'price'** , **'count'** ], dtype = **'object'** )

Keling, shunga o'xshash lug'at tuzamiz, lekin ndarray elementlari bo'yicha *:*

>>> d2 = { **'narx'** : np.array ([ 1 , 2 , 3 ]), **'count'** : np.array ([ 10 , 12 , 7 ])}

>>> df2 = pd.DataFrame (d2, index = [ **'v1'** , **'v2'** , **'v3'** ])

>>> chop etish (df2)

narxlar soni

v1 1 10

v2 2 12

v3 3 7

>>> chop etish (df2.index)

Indeks ([ **'v1'** , **'v2'** , **'v3'** ], dtype = **'ob'ekt'** )

>>> chop etish (df2. ustunlar)

Indeks ([ **'price'** , **'count'** ], dtype = **'object'** )

Ko'rib turganingizdek, natija oldingisiga o'xshaydi. Ndarray o'rniga, mumkin

oddiy *Python* ro'yxatidan foydalaning .

**1.2.2 Lug'atlar ro'yxatidan *DataFrame* yaratish**

Bungacha biz elementlari bo'lgan lug'atdan *DataFrame* yaratdik

*ketma-ket* tuzilmalar , ro'yxatlar va massivlar mavjud edi, endi biz yaratamiz

*DataFrame* elementlari lug'atlar bo'lgan ro'yxatdan:

17

|  |
| --- |
| **19-bet** |

>>> d3 = [{ **'narx'** : 3 , **'hisoblash'** : 8 }, { **'narx'** : 4 , **'hisoblash'** : 11 }]

>>> df3 = pd.DataFrame (d3)

>>> chop etish (df3)

narxni hisoblash

0 8 3

1 11 4

Yaratilgan *DataFrame-* ning qisqacha mazmuni bilan tanishishingiz mumkin

info () funktsiyasidan foydalaning. Bu strukturaning turi haqida ma'lumot,

yozuvlar soni, ustunlardagi bo'sh bo'lmagan elementlar soni, turlari va

saqlangan narsalar soni va ishlatilgan xotira hajmi:

>>> chop etish (df3.info ())

< **class 'pandas.core.frame.DataFrame'** >

RangeIndex: 2 ta yozuv, 0 dan 1 gacha

Ma'lumot ustunlari (jami 2 ta ustun):

hisoblash 2 nodavlat null Int64

narx 2 null bo'lmagan int64

diplar: int64 ( 2 )

xotiradan foydalanish: 112.0 bayt

**Yo'q**

**1.2.3 bir yaratish *DataFrame* bir 2D qator**

Siz *DataFrame-* ni ikki o'lchovli massivdan yaratishingiz mumkin

masalan, bu numpy kutubxonasidan ndarray bo'lishi mumkin:

>>> nda1 = np.array ([[ 1 , 2 , 3 ], [ 10 , 20 , 30 ]])

>>> df4 = pd.DataFrame (nda1)

>>> chop etish (df4)

0 1 2

0 1 2 3

1 10 20 30

18

|  |
| --- |
| **20-bet** |

**1.2.4 *DataFrame elementlari*bilan ishlash**

*DataFrame* elementlari *bilan* ishlash juda keng mavzudir va u

3-bobda keltirilgan. Endi biz eng ko'p ko'rib chiqamiz

tuzilish elementlariga kirish uchun ishlatilgan yondashuvlar. Asosiy

usullari 1.1-jadvalda keltirilgan.

**1.1-jadval - *DataFrame elementlari* bilan ishlash**

**Ishlash**

**Sintaksis**

**Natija qaytarildi**

Ustun tanlash

df [col]

*Seriya*

Yorliq bo'yicha qatorni tanlash

df.loc [label]

*Seriya*

Bir qatorni indeks bo'yicha tanlash

df.iloc [loc]

*Seriya*

Satr bo'yicha bo'lak

df [0: 4]

*DataFrame*

Mos keladigan chiziqlarni tanlash

holat

df [bool\_vec]

*DataFrame*

Keling, ushbu operatsiyalar bilan ishlashni amalda ko'rib chiqaylik. Boshlanishiga

*DataFrame* yarating *:*

>>> d = { **'narx'** : np.array ([ 1 , 2 , 3 ]), **'hisoblash'** : np.array ([ 10 , 20 , 30 ])}

>>> df = pd.DataFrame (d, indeks = [ **'a'** , **'b'** , **'c'** ])

>>> chop etish (df)

narxlar soni

a 1 10

b 2 20

v 3 30

Amaliyot: ustun tanlash:

>>> df [ **'hisoblash'** ]

a 10

b 20

dan 30

Ism: count, dtype: int32

o'n to'qqiz

|  |
| --- |
| **21-bet** |

Amaliyot: yorliq bo'yicha qatorni tanlang:

>>> df.loc [ **'a'** ]

narx 1

10ni hisoblash

Ism: a, dtype: int32

Amaliyot: indeks bo'yicha qatorni tanlang:

>>> df.iloc [ 1 ]

narx 2

hisoblash 20

Ism: b, dtype: int32

Amaliyot: satrlar bo'yicha tilim:

>>> df [ 0 : 2 ]

narxlar soni

a 1 10

b 2 20

Amaliyot: shartga mos keladigan qatorlarni tanlang:

>>> df [df [ **'hisoblash'** ]> = 20 ]

narxlar soni

b 2 20

v 3 30

20

|  |
| --- |
| **Sahifa 22** |

**2-bob. *Pandalar*tarkibidagi ma'lumotlarga kirish**

**2.1 *Pandalarda* ma'lumotlarga kirishning ikkita yondashuvi**

*Series* va *DataFrame tuzilmalari* bilan ishlashda , qoida tariqasida, foydalaning

element qiymatlarini olishning ikkita asosiy usuli mavjud.

**Birinchi usul** yorliqlardan foydalanishga asoslangan, bu holda ish

.loc usuli orqali boshqariladi. Agar siz yo'qolgan yorliqni nazarda tutsangiz,

keyin KeyError istisnosi tashlanadi. Ushbu yondashuv imkon beradi

foydalanish:

• individual belgilar [ **'** a **'** ] yoki raqamlar [5], raqamlar ko'rinishidagi yorliqlar

agar tuzilish bo'lmasa yorliq sifatida ishlatiladi

teglar bilan ro'yxat ko'rsatildi;

• yorliqlar ro'yxati [ **'** a **'** , **'** b **'** , **'** c **'** ];

• tilim yorliqlari [ **'** a **'** : **'** c **'** ];

• mantiqiy o'zgaruvchilar massivi;

• *callable -* bitta argumentli funktsiya.

**Ikkinchi yo'l,** kirish uchun butun sonlardan foydalanishga asoslangan

ma'lumotlar .iloc usuli orqali aniqlanadi. Agar siz murojaat qilsangiz

mavjud bo'lmagan element, istisno tashlanadi

IndexError. .Iloc dan foydalanish mantig'i ishlashga juda o'xshaydi

.lok. Ushbu yondashuv yordamida siz quyidagilarni ishlatishingiz mumkin:

• struktura elementlariga kirish uchun alohida butun sonlar;

• butun sonli massivlar [0, 1, 2];

• butun sonlarning tilimlari [1: 4];

• mantiqiy o'zgaruvchilar massivlari;

• *callable -* bitta argumentli funktsiya.

21

|  |
| --- |
| **23-bet** |

.Loc shakli tuzilish turiga qarab o'zgaradi:

• *Series* uchun quyidagicha ko'rinadi: s.loc [indexer];

• uchun *DataFrame* bu kabi: df.loc [row\_indexer, column\_indexer] *.*

Kelajakda *Series* va *DataFrame turidagi* moslamalarni yarataylik

biz tomonidan tajribalar uchun foydalaniladi.

*Seriya* tarkibi :

>>> s = pd.Series ([ 10 , 20 , 30 , 40 , 50 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

>>> s [ **'a'** ]

o'n

>>> lar

a 10

b 20

dan 30

g 40

e 50

dip turi: int64

*DataFrame* tuzilishi :

>>> d = { **'narx'** : [ 1 , 2 , 3 ], **'hisoblash'** : [ 10 , 20 , 30 ], **'foiz'** : [ 24 , 51 ,

71 ]}

>>> df = pd.DataFrame (d, indeks = [ **'a'** , **'b'** , **'c'** ])

>>> df

narxlar soni

a 1 10 24

b 2 20 51

v 3 30 71

22

|  |
| --- |
| **24-bet** |

**2.2 Bir *qator* strukturaning ma'lumotlariga kirish**

**2.2.1 Yorliqlar yordamida kirish**

Ma'lumotlarga kirish uchun yorliqlardan foydalanganda, bittasi

quyidagi yondashuvlardan:

• birinchi navbatda, *Series* va in turidagi o'zgaruvchining nomini yozganda

kvadrat qavslar kerakli yorliqni bildiradi

aloqa (misol: s [ **'** a **'** ]);

• ikkinchidan, o'zgaruvchidan keyin .loc va undan keyin yozing

yorliqni to'rtburchak qavsda ko'rsating (masalan: s.loc [ **'** a **'** ]).

Alohida yorliq bilan davolash.

**"** A **"** deb nomlangan element :

>>> s [ **'a'** ]

o'n

Bir qator yorliqlarga kirish.

**'** A **'** , **'** c **'** va **'** e **'** yorliqli elementlar :

>>> s [[ **'a'** , **'c'** , **'e'** ]]

a 10

dan 30

e 50

dip turi: int64

Yorliqlar bo'lagi bo'ylab teskari yo'nalish.

**"** A **"** dan **"** e **"** gacha belgilangan tarkibiy qismlar :

>>> s [ **'a'** : **'c'** ]

a 10

b 20

dan 30

dip turi: int64

23

|  |
| --- |
| **25-bet** |

**2.2.2 Butun sonli indekslar yordamida kirish**

Butun sonli indekslar bilan ishlashda indeks darhol o'rnatilishi mumkin

o'zgaruvchan nomdan keyin to'rtburchak qavsda (masalan: s [1]) yoki

.iloc dan foydalaning (masalan: s.iloc [1]).

Alohida indeksga murojaat qilish.

Indeks 1 bo'lgan buyum:

>>> lar [ 1 ]

20

Indekslar ro'yxati yordamida kirish.

1, 2 va 3 indekslari bo'lgan narsalar:

>>> lar [[ 1 , 2 , 3 ]]

b 20

dan 30

g 40

Dilimga kirish uchun indeks.

Tuzilishning dastlabki uchta elementini olish:

>>> lar [: 3 ]

a 10

b 20

dan 30

dip turi: int64

**2.2.3 *Qo'ng'iroq qilinadigan* funktsiya yordamida kirish**

Ushbu yondashuv bilan u indeks yoki emas

yorliq va funktsiya (odatda *lambda* funktsiyasi)

tuzilish elementlarini tanlash uchun ishlatiladi.

24

|  |
| --- |
| **Sahifa 26** |

Qiymati 30 dan katta yoki unga teng bo'lgan narsalar:

>>> s [ **lambda** x: x> = 30 ]

dan 30

g 40

e 50

dip turi: int64

**2.2.4 Boolean Expression yordamida kirish**

Ushbu yondashuv *chaqiriladigan* funktsiya bilan ishlashga o'xshaydi : kvadrat ichida

mantiqiy ifoda qavs ichida yozilgan, unga muvofiq bo'ladi

tanlov amalga oshirildi.

Qiymati 30 dan katta bo'lgan narsalar:

>>> lar [s> 30 ]

g 40

e 50

dip turi: int64

**2.3 *DataFrame* ma'lumotlariga kirish**

**2.3.1 Yorliqlar yordamida kirish**

Yorliqlar uchun turli xil holatlarni ko'rib chiqamiz

ikkala jadval ustuni nomlari va qator nomlari bo'ling.

Muayyan ustunga murojaat qilish.

**"** Hisoblash **"** ustunining elementlari :

>>> df [ **'hisoblash'** ]

a 10

b 20

dan 30

Ism: count, dtype: int64

25

|  |
| --- |
| **Sahifa 27** |

Bir qator ustunlar yordamida kirish.

**"** Hisoblash **"** va **"** narx **"** ustunlari elementlari :

>>> df [[ **'hisoblash'** , **'narx'** ]]

narxni hisoblash

a 10 1

b 20 2

v 30 3

Yorliqlar bo'lagi bo'ylab teskari yo'nalish.

**"** A **"** dan **"** b **"** gacha bo'lgan elementlar :

>>> df [ **'a'** : **'b'** ]

narxlar soni

a 1 10 24

b 2 20 51

**2.3.2 *Qo'ng'iroq qilinadigan* funktsiya yordamida kirish**

*DataFrame* uchun *chaqiriladigan* funktsiya bilan ishlashga yondashuv bir xil

faqat shartlarni shakllantirishda *Series* uchun ishlatiladigan narsa

ustun nomini qo'shimcha ravishda ko'rsatishingiz kerak.

**"** Hisoblash **"** ustunidan katta qiymatga ega bo'lgan barcha narsalarni oling

15:

>>> df [ **lambda** x: x [ **'hisoblash'** ]> 15 ]

narxlar soni

b 2 20 51

v 3 30 71

**2.3.3 Boolean Expression yordamida kirish**

Mantiqiy ifodani shakllantirishda siz ko'rsatishingiz kerak

ustun nomlari, shuningdek *chaqiriladigan* funktsiyalar bilan ishlashda , tomonidan

namuna olinadi.

26

|  |
| --- |
| **Sahifa 28** |

**"** Narxi **"** 2 dan katta yoki teng bo'lgan barcha elementlarni olaylik :

>>> df [df [ **'narx'** ]> = 2 ]

narxlar soni

b 2 20 51

v 3 30 71

**2.4 Ma'lumotlarga kirish uchun atributlardan foydalanish**

Ma'lumotlarga kirish uchun tuzilmalar atributlaridan foydalanishingiz mumkin

bu yorliqlar. *Seriya* tuzilishidan boshlaymiz .

Keling, bizga tanish bo'lgan tuzilishni ishlatamiz:

>>> s = pd.Series ([ 10 , 20 , 30 , 40 , 50 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ])

Atribut orqali elementga kirish uchun uni belgilashingiz kerak

o'zgaruvchining nomidan keyin nuqta:

>>> sa

o'n

>>> sc

o'ttiz

Chunki *s* tuzilmasi **'***a***'***,***'** b **'***,***'** c **'***,***'** d **'***,***'** e **'** yorliqlariga ega , so'ngra kirish uchun

**'** a **'** deb nomlangan element sa sintaksisidan foydalanadi

Xuddi shu yondashuv *DataFrame* turidagi o'zgaruvchiga nisbatan qo'llanilishi *mumkin* :

>>> d = { **'narx'** : [ 1 , 2 , 3 ], **'hisoblash'** : [ 10 , 20 , 30 ], **'foiz'** : [ 24 , 51 ,

71 ]}

>>> df = pd.DataFrame (d, indeks = [ **'a'** , **'b'** , **'c'** ])

27

|  |
| --- |
| **Sahifa 29** |

Keling, **"** narx **"** ustuniga kiraylik :

>>> narx

a 1

b 2

v 3

Ism: narx, tur: int64

**2.5 *Pandalar konstruktsiyalaridan* tasodifiy to'plamni olish**

*Pandalar* kutubxonasi tasodifiy olish imkoniyatini beradi

mavjud tuzilmaning ma'lumotlar to'plami. Bunday funktsionallik

ikkala beradi *Series* va *DataFrame* . Tasodifiy namuna,

sample () usuli yordamida olingan.

Dastlab, strukturani misol qilib ushbu usul bilan ishlashni ko'rib chiqamiz.

*Seriya* .

*Seriyadan* tasodifiy elementni tanlash uchun quyidagilar ishlatiladi

sintaksis:

>>> s.sample ()

a 10

dip turi: int64

Siz bir nechta elementlardan birini tanlashingiz mumkin, buning uchun sizga kerak

elementlar sonini n parametridan o'tkazing:

>>> namuna ( n = 3 )

dan 30

a 10

g 40

dip turi: int64

28

|  |
| --- |
| **Sahifa 30** |

Tuzilmadagi ob'ektlarning umumiy sonining ulushini ko'rsatish mumkin,

frac parametri buning uchun javobgardir:

>>> s.sample ( frac = 0.3 )

g 40

e 50

dip turi: int64

Bundan tashqari, argument sifatida biz og'irlik vektorini,

uning uzunligi strukturadagi elementlar soniga teng bo'lishi kerak va

vektor elementlarining yig'indisi bitta. Og'irligi, bu holda

namunadagi narsaning paydo bo'lishi ehtimoli.

Sinov strukturamizda beshta element mavjud, biz og'irliklar vektorini hosil qilamiz

buning uchun va uchta elementni tanlang:

>>> w = [ 0,1 , 0,2 , 0,5 , 0,1 , 0,1 ]

>>> namuna ( n = 3 , vazn = w)

dan 30

b 20

a 10

dip turi: int64

*DataFrame* tuzilmasi uchun sample () usuli ham mavjud :

>>> d = { **'narx'** : [ 1 , 2 , 3 , 5 , 6 ], **'hisoblash'** : [ 10 , 20 , 30 , 40 , 50 ],

**'foiz'**: [ 24 , 51 , 71 , 25 , 42 ]}

>>> df = pd.DataFrame (d)

>>> df.sample ()

narxlar soni

2 3 30 71

29

|  |
| --- |
| **Sahifa 31** |

*DataFrame* bilan ishlashda siz o'qni belgilashingiz mumkin. Ustunni tanlash uchun

tasodifiy o'qni 1 ga qo'ying:

>>> df.sample ( o'q = 1 )

hisoblash

0 10

1 20

2 30

3 40

4 50

Ikkita ustunni tasodifiy tanlash:

>>> df.sample ( n = 2 , o'q = 1 )

foizlar soni

0 24 10

1 51 20

2 71 30

3 25 40

4 42 50

Ikki qatorni tasodifiy tanlash:

>>> df.sample ( n = 2 )

narxlar soni

0 1 10 24

1 2 20 51

**2.6 Mantiqiy iboralar yordamida indeksatsiya qilish**

Amalda, ko'pincha mavjud bo'lganlardan pastki namunalarni olish kerak

ma'lumotlar to'plami. Masalan: barcha tovarlarni oling, unda chegirma

besh foizdan ko'p yoki ma'lumotlar bazasidan ma'lumotlarni tanlang

30 yoshdan oshgan erkak xodimlar. Bu jarayonga juda o'xshaydi

jadvallar bilan ishlashda yoki ma'lumotlar bazasidan tanlovni olishda filtrlash

o'ttiz

|  |
| --- |
| **Sahifa 32** |

ma'lumotlar. Shunga o'xshash funktsiyalar *pandalarda* ham amalga oshiriladi va biz allaqachon tegdik

Indekslashga turli xil yondashuvlarni ko'rib chiqishda ushbu savolga qarang:

indeks o'rniga shartli ifoda yozilishi kerak

tuzilish elementlariga murojaat qilganda kvadrat qavslar.

*Series* bilan ishlashda quyidagi foydalanish holatlari mumkin:

>>> s = pd.Seriyalar ([ 10 , 20 , 30 , 40 , 50 , 10 , 10 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ,

**'f'**, **'g'**])

>>> lar [s> 30 ]

g 40

e 50

dip turi: int64

Biz strukturaning barcha elementlarini olamiz, ularning qiymati 10 ga teng:

>>> s [s == 10 ]

a 10

f 10

g 10

dip turi: int64

Qiymatlari [30, 50) oralig'ida bo'lgan *s* tuzilish elementlari :

>>> s [(s> = 30 ) & (s < 50 )]

dan 30

g 40

dip turi: int64

*DataFrame* bilan ishlashda uning ustunini belgilashingiz kerak

filtrlash (tanlov) amalga oshiriladi:

>>> d = { **'narx'** : [ 1 , 2 , 3 , 5 , 6 ], **'hisoblash'** : [ 10 , 20 , 30 , 40 , 50 ],

**'foiz'**: [ 24 , 51 , 71 , 25 , 42 ], **'mushuk'**: [ **'A'**, **'B'**, **'A'**, **'A'**, **'C'**]}

>>> df = pd.DataFrame (d)

31

|  |
| --- |
| **Sahifa 33** |

>>> df

mushuklarning narxlari soni

0 1 10 24 A

1 2 20 51 B

2 3 30 71 A

3 5 40 25 A

4 6 50 42 S

**"** Narx **"** maydonida qiymati bo'lgan jadval qatorlari ro'yxatini tanlang

3 dan ortiq:

>>> df [df [ **'narx'** ]> 3 ]

mushuklarning narxlari soni

3 5 40 25 A

4 6 50 42 S

Mantiqiy ifoda sifatida siz juda ko'p foydalanishingiz mumkin

xarita, filtr, *lambda* funktsiyalari yordamida murakkab inshootlar -

iboralar va boshqalar.

**"** Mushuk **"** maydon qiymatiga ega bo'lgan df strukturasining indekslari ro'yxatini olaylik.

"A" ga teng:

>>> fn = df [ **'mushuk'** ] .map ( **lambda** x: x == **'A'** )

>>> df [fn]

mushuklarning narxlari soni

0 1 10 24 A

2 3 30 71 A

3 5 40 25 A

**2.7 *Pandalardagi* ma'lumotlar bilan ishlashda isin-dan foydalanish**

Elementlar bilan *massivlarni pandalar* ma'lumotlar tuzilmalarida qurish mumkin

bool turi, bu bilan siz uning mavjudligini yoki yo'qligini tekshirishingiz mumkin

yoki asl tuzilishdagi boshqa element.

32

|  |
| --- |
| **Sahifa 34** |

Biz quyidagi tuzilma bilan ishlaymiz:

>>> s = pd.Seriyalar ([ 10 , 20 , 30 , 40 , 50 , 10 , 10 ], [ **'a'** , **'b'** , **'c'** , **'d'** , **'e'** ,

**'f'**, **'g'**])

Bool tipidagi elementlar bilan yangi tuzilmani quraylik, agar shunday bo'lsa

asl tuzilish elementining qiymati ro'yxatda [10, 20], keyin

element qiymati True, aks holda False:

>>> s.isin ([ 10 , 20 ])

a **True**

b **To'g'ri**

v **noto'g'ri**

d **soxta**

e **Soxta**

f **To'g'ri**

g **To'g'ri**

dip: bool

A bilan ishlash *DataFrame bo'lgan* bir bilan ishlash o'xshash *Series* tuzilishi :

>>> df = pd.DataFrame ({ **'narx'** : [ 1 , 2 , 3 , 5 , 6 ], **'hisoblash'** : [ 10 , 20 , 30 , 40 ,

50 ], **'foiz'** : [ 24 , 51 , 71 , 25 , 42 ]})

Agar asl tuzilish elementining qiymati ro'yxatda bo'lsa [1, 3,

25, 30, 10], keyin yangi strukturadagi elementning qiymati To'g'ri, aks holda -

Yolg'on:

>>> df.isin ([ 1 , 3 , 25 , 30 , 10 ])

narxlar soni

0 **Haqiqiy yolg'on**

1 **yolg'on yolg'on**

2 **Haqiqiy yolg'on**

3 **yolg'on yolg'on**

4 **yolg'on yolg'on**

33

|  |
| --- |
| **Sahifa 35** |

**3-bob. *Pandalardagi* ma'lumotlar turlari**

Har qanday dasturlash tilini o'rganishda eng muhim mavzulardan biri

master tizimining turi. Masalan, bilan ishlashda

yilda *Python,* siz bunday uchun int va float sifatida turdagi duch

raqamlar bilan ishlash, str - satrlar bilan ishlash uchun; ko'proq turlari bor

ro'yxatlar, lug'atlar, to'plamlar kabi murakkab. *Pandalar* kutubxonasi

samarali saqlash uchun o'ziga xos ma'lumotlar turlarini o'z ichiga oladi va

ma'lumotlar manipulyatsiyasi. Oldingi boblarda siz allaqachon duch kelgansiz

masalan, dtype kabi tushunchalar bilan: masalan, bilan bir *qator* tarkibini namoyish qilishda

saqlangan qiymatlar turi ko'rsatilganda bosib chiqarish funktsiyasidan foydalanish

oxirgi satrda:

>>> s = pd.Seriyalar ([ 1 , 2 , 3 ])

>>> bosma (lar)

0 1

12

2 3

dip turi: int64

Yuqoridagi misolda s-da saqlanadigan qiymatlarning turi int64.

Yoki haqida batafsil ma'lumot aks *DataFrame* tuzilishi bilan

info () usuli yordamida: ustun nomlaridan tashqari, haqida ma'lumot

elementlarning soni, ishg'ol qilingan xotira va boshqalar, turi chiqarib tashlanadi

har bir ustun uchun saqlangan ma'lumotlar:

>>> d = [{ **'name'** : **'ruchka'** , **'price'** : 3.9 , **'count'** : 8 }, { **'name'** : **'book'** ,

**'narx'**: 4,5 , **'hisoblash'**: 11 }]

>>> df = pd.DataFrame (d)

34

|  |
| --- |
| **Sahifa 36** |

>>> df.info ()

< **class 'pandas.core.frame.DataFrame'** >

RangeIndex: 2 ta yozuv, 0 dan 1 gacha

Ma'lumot ustunlari (jami 3 ta ustun):

hisoblash 2 nodavlat null Int64

nom 2 nodavlat null ob'ekt

narx 2 bo'sh bo'lmagan float64

diplar: float64 ( 1 ), int64 ( 1 ), ob'ekt ( 1 )

xotiradan foydalanish: 128.0 + bayt

Yuqoridagi ma'lumotlardan siz hisoblash maydonchasi saqlanishini ko'rishingiz mumkin

int64 tipidagi elementlar, nom maydonida - ob'ekt turi bilan, narx maydonida - bilan

float64 turi. Ushbu bobda biz turlarni batafsilroq ko'rib chiqamiz

*pandas* kutubxonalari va ular bilan ishlashni o'rganing.

**3.1 Ma'lumot turlari**

*Pandalarda* ishlatiladigan asosiy ma'lumotlar turlari berilgan

3.1-jadval.

**3.1-jadval - *Pandalardagi* ma'lumotlar turlari**

**Ma'lumot turi**

**Tavsif**

int64

Ga bog'liq bo'lmagan 64 bitli tamsayı qiymati

platforma

float64

64 bitli suzuvchi nuqta raqami

platforma

ob'ekt

Matn yoki boshqa ma'no

bool

Mantiqiy: To'g'ri / noto'g'ri

toifasi

Matn elementlarining cheklangan to'plami

datetime64 [ns]

Sana vaqti

timedelta64 [ns]

Ikki datetime elementlari orasidagi farq

35

|  |
| --- |
| **37-bet** |

Odatda, sizga *pandalar* mumkin bo'lgan qiymatlar turini aniq belgilashingiz shart emas

turini o'zingiz chiqaring, lekin agar siz uni belgilashingiz kerak bo'lsa yoki

turini o'zgartirishni amalga oshiring, keyin siz imkoniyatlardan foydalanishingiz kerak

kutubxona beradi.

Tuzilishda ishlatiladigan turlari haqida ma'lumot olish mumkin

diplar xususiyati. Df tuzilishi uchun biz yuqoridagi misoldan olamiz

quyidagi natija:

>>> df.dtypes

hisoblash int64

ob'ekt nomi

float64

turi: ob'ekt

Dtype xususiyati uchun mavjud *DataFrame* va *Series* tuzilmalar .

*Pandalar* ichida turlarni yaratish va ulardan foydalanish ham mumkin

kengaytmalar. Kutubxona kengaytma turlari bilan ta'minlangan,

3.2-jadvalda keltirilgan.

**Jadval 3.2 - *Pandalarni* kengaytirish turlari**

**Turi**

**Tavsif**

DatetimeTZDtype

vaqt mintaqasini qo'llab-quvvatlash bilan datetime

Kategoriya turi

To'liq ma'lumotlar uchun yozing (cheklangan to'plam)

matn elementlari)

Davr turi

Davriy ma'lumotlar turi

Siyrak

Siyrak ma'lumotlar bilan ishlash turi

IntervalDtype

Ma'lumotlarning intervalli turi

36

|  |
| --- |
| **38-bet** |

**3.2 Turlar bilan ishlash vositalari**

*Pandalar* , turlari bilan ishlash uchun bir qator vositalarni taqdim etadi

"Turlar bilan ishlash" biz kastingning turli xil operatsiyalarini tushunamiz

turlari.

***3.2.1 astype ()***

Biz ko'rib chiqadigan birinchi vosita bu astype () usuli. u

uchun ham mavjud *Series* va *DataFrame tuzilmalar* :

astype (self, dtype, copy = **True** , xatolar = **'ko'tarish'** , \*\* kwargs)

• dip turi: ma'lumotlar turi yoki ustun nomlari bilan lug'at

Cast Translatsiya uchun ma'lumotlar turi bo'lishi mumkin

numpy.dtype, *pandas* turi yoki lug'atda ishlatilgan

format {col: dtype, ...}, bu erda col - ustun nomi, dtype

kerakli ma'lumotlar turi.

• nusxa ko'chirish: bool; standart qiymat: To'g'ri

◦ astype, agar parametr bo'lsa, strukturaning nusxasini qaytaradi

to'g'ri, aks holda oqim

tuzilishi.

• xatolar: { **'** ko'tarish **'** , **'** e'tiborsiz qoldirish **'** }; sukut bo'yicha: "ko'tarish"

Exc Istisnolarni tashlash jarayonini boshqaradi:

▪ ko'tarish: istisnolardan foydalanishni yoqish;

▪ e'tiborsiz qoldiring: istisnolarni e'tiborsiz qoldiring.

• \*\* kvarg

◦ Konstruktorga o'tish uchun argumentlar.

***Seriyali* tuzilish uchun turdagi konvertatsiya**

Keling, butun sonlar bilan tuzilishni yarataylik:

>>> s = pd.Seriyalar ([ 1 , 2 , 3 ])

>>> s.dype

dtype ( **'int64'** )

37

|  |
| --- |
| **39-bet** |

Keling, ushbu ma'lumotlarni float64 turiga o'tkazaylik:

>>> s.astype ( **'float64'** )

0 1.0

1 2.0

2 3.0

dip: float64

***DataFrame* struktura turlarini o'zgartirish**

Keling, tuzilmani yarataylik:

>>> d = [{ **'name'** : **'ruchka'** , **'price'** : 3.9 , **'count'** : 8 }, { **'name'** : **'book'** ,

**'narx'**: 4,5 , **'hisoblash'**: 11 }]

>>> df = pd.DataFrame (d)

>>> df.dtypes

hisoblash int64

ob'ekt nomi

float64

turi: ob'ekt

Bo'lsin »tashladilar s maydon turi **"** soni **"** uchun int32:

>>> df [ **'hisoblash'** ] = df [ **'hisoblash'** ] .astype ( **'int32'** )

>>> df.dtypes

hisoblash int32

ob'ekt nomi

float64

turi: ob'ekt

Keling, avvalgi turini **"** hisoblash **"** maydoniga qaytaramiz , boshqacha usulni qo'llaymiz

funktsiyasini chaqirish:

>>> df = df.astype ({ **'count'** : **'int64'** })

>>> df.dtypes

hisoblash int64

ob'ekt nomi

float64

turi: ob'ekt

38

|  |
| --- |
| **40-bet** |

**3.2.2 Ma'lumotlarni tayyorlash funktsiyalari**

Ko'pincha oldin ma'lumotlarni oldindan tayyorlash vazifasi mavjud

turdagi konversion operatsiyani bajarish. Ma'lumotlar to'plami

tajribalar 3.3-jadvalda keltirilgan.

**3.3-jadval - tajribalar uchun ma'lumotlar to'plami**

**Harorat**

**Bosim**

**Yog'ingarchilik**

**sana**

- 8 ° S

96292 Pa

Ha

2019-11-20

- 10,3 ° S

97292 Pa

Ha

2019-11-21

- 9,1 ° S

96325 Pa

Yo'q

2019-11-22

Keling, ushbu jadvalni *DataFrame-ga yuklaymiz* :

>>> df = pd.read\_csv ( **"c: /data.csv"** , Sep = **','** )

>>> df

Harorat bosimi Yog'ingarchilik sanasi

0 - 8 ° C 96292 Pa Ha 2019 - 11 - 20

1 - 10,3 ° C 97292 Pa Ha 2019 - 11 - 21

2 - 9,1 ° C 96325 Pa yo'q 2019 - 11 - 22

Keling, turlarini ko'rib chiqaylik:

>>> df.dtypes

Ob'ekt harorati

Ob'ekt bosimi

Yog'ingarchilik ob'ekti

sana

ob'ekt

turi: ob'ekt

Ko'rib turganingizdek, barcha maydonlarning turlari ob'ektivdir va biz bunga muhtojmiz

birinchi ustundagi ma'lumotlar (Harorat) float64 formatida edi

ikkinchisi - int64, uchinchisi - bool. Agar biz to'g'ridan-to'g'ri harakat qilsak

astype () funktsiyasidan foydalangan holda quying, keyin ushbu operatsiyani bajaring

muvaffaqiyatsizlikka uchraydi, chunki masalan, harorat uchun

39

|  |
| --- |
| **41-bet** |

bizni qiziqtiradigan raqamli qiymat birlikdir

o'lchovlar. Ushbu muammoni hal qilish uchun biz bir nechta funktsiyalarni yozamiz

ma'lumotlarni kerakli shaklga keltirishga yordam beradigan o'zgarishlar:

>>> temper\_convertor = **lambda** x: x.replace ( **'° C'** , **''** ) .strip ()

>>> pressure\_convertor = **lambda** x: x.replace ( **'Pa'** , **''** ) .strip ()

>>> prec\_convertor = **lambda** x: **True bo'lsa** x == **'Ha' else False**

Yaratilgan funktsiyalarni tuzilish elementlariga qo'llaymiz:

>>> df [ **'Harorat'** ] =

df [ **'Temperature'** ] .apply (temper\_convertor) .astype ( **'float64'** )

>>> df [ **'Bosim'** ] =

df [ **'Bosim'** ] .apply ( **press\_convertor** ) .astype ( **'int64'** )

>>> df [ **'Yog'ingarchilik'** ] = df [ **'Yog'ingarchilik'** ] .apply (prec\_convertor)

>>> df.dtypes

Harorat

float64

Bosim

int64

Yog'ingarchilik

bool

sana

ob'ekt

turi: ob'ekt

**3.2.3 Yordamchi funktsiyalari**

*Pandalar* konversion yordamchi funktsiyalarini ta'minlaydi

rivojlanish zarurligini bartaraf etadigan ma'lumotlar

tegishli echimlarning o'zi, uchta funktsiya mavjud:

to\_numeric () - ma'lumotlarni raqamli turga o'zgartiradi;

to\_datetime () - ma'lumotlarni datetime turiga o'zgartiradi;

to\_timedelta () - Ma'lumotlarni timedelta turiga o'zgartiradi.

40

|  |
| --- |
| **42-bet** |

Yangi tayyorlangan ma'lumotlar to'plamini yuklaylik:

>>> df = pd.read\_csv ( **'c: /data.csv'** , **sep** = **','** )

>>> df.dtypes

Harorat

ob'ekt

Bosim

ob'ekt

Yog'ingarchilik

ob'ekt

sana

ob'ekt

turi: ob'ekt

Transformatsiya qilish uchun yuqoridagi funktsiyalardan foydalanamiz

ma'lumotlar:

>>> df [ **'Harorat'** ] = pd.to\_numeric (df [ **'Harorat'** ],

xatolar = **'majburlash'** )

>>> df [ **'Bosim'** ] = pd.to\_numeric (df [ **'Bosim'** ],

xatolar = **'majburlash'** )

>>> df [ **'Sana'** ] = pd.to\_datetime (df [ **'Sana'** ], xatolar = **'majburlash'** )

>>> df.dtypes

Harorat

float64

Bosim

int64

Yog'ingarchilik

ob'ekt

sana

datetime64 [ns]

dip turi:

ob'ekt

**3.2.4 Ma'lumotlarni turlari bo'yicha olish**

Ma'lumotlarni turiga qarab tanlash uchun select\_dtypes () funktsiyasidan foydalaning,

u asl *DataFrame '* a, in- dan o'rnatilgan *DataFrame-ni* qaytaradi

argumentda ko'rsatilgan turlarga ustunlar kiradi

o'z ichiga oladi va turlari ko'rsatilgan ustunlarni o'z ichiga olmaydi

argumentni chiqarib tashlash:

>>> df.select\_dtypes (o'z ichiga = [ **'float64'** , **'int64'** ])

Harorat

Bosim

0

- 8.0

96292

1

- 10.3

97292

2018-04-02 121 2

- 9.1

96325

41

|  |
| --- |
| **43-bet** |

>>> df.select\_dtypes (chiqarib tashlang = **'datetime64 [ns]'** )

Harorat

Yog'ingarchilik bosimi

0

- 8.0

96292

Ha

1

- 10.3

97292

Ha

2018-04-02 121 2

- 9.1

96325

Yo'q

**3.3 Kategorik turlari**

*Pandalardagi* kategorik ma'lumotlar turlari tabiatan o'xshashdir

statistikadagi sifat belgilari. Ular final sifatida ko'rsatilgan

qator o'zgaruvchilari to'plami. Bunga misollar kiradi

quyidagi:

• rang: *qizil* , *yashil* , *ko'k* ;

• shrift uslubi: *Normal* , *qalin* , *kursiv* ;

• matnni tekislash: *o'ng* , *markaz* , *chap* .

**3.3.1 Kategorik ma'lumotlar to'plami bilan tuzilmani yaratish**

Elementlar bilan tuzilishni yaratish variantlarini ko'rib chiqing

turkum turi.

***Seriya* tuzilishi bilan ishlash**

Agar biz shunchaki *ketma-ket* strukturani yaratadigan bo'lsak , turini ko'rsatmasdan, keyin

biz ob'ekt turi elementlari to'plamini olamiz:

>>> s = pd.Series ([ **'r'** , **'r'** , **'g'** , **'b'** ])

>>> lar

0

r

1

r

2018-04-02 121 2

g

3

b

turi: ob'ekt

42

|  |
| --- |
| **Sahifa 44** |

Agar elementlarga tegishli ekanligini aniq ko'rsatish kerak bo'lsa

kategorik turi, bu dtype argumenti orqali amalga oshirilishi kerak:

>>> s = pd.Series ([ **'r'** , **'r'** , **'g'** , **'b'** ], dtype = **'category'** )

>>> lar

0

r

1

r

2018-04-02 121 2

g

3

b

dip turi: turkum

Kategoriyalar ( 3 , ob'ekt): [b, g, r]

***Kategorik* tip bilan ishlash**

Agar toifaning tuzilishi oldindan ma'lum bo'lsa: uning elementlari to'plami va

buyurtma, keyin siz *Kategorik* sinf ob'ektini yaratishingiz mumkin :

Kategorik (qiymatlar, toifalar = **Yo'q** , buyurtma qilingan = **Yo'q** , dtype = **Yo'q** ,

fastpath = **False** )

• qiymatlar: ro'yxat

◦ buyumlar. Agar toifa qo'shimcha ravishda ko'rsatilgan bo'lsa

kategoriyalar parametri orqali, keyin qiymatlar toifadan emas

NaN bilan almashtiriladi.

• toifalar: noyob elementlar to'plami, Yo'q; qiymati

sukut bo'yicha: yo'q

◦ Element qabul qilishi mumkin bo'lgan qiymatlar to'plamini belgilaydi

toifalar. Agar Hech kimga teng bo'lsa, unda toifadagi to'plam tomonidan quriladi

qiymatlar parametridan noyob elementlar.

• buyurtma qilingan: bool; standart qiymat: noto'g'ri

◦ toifaning tartibli yoki yo'qligini aniqlaydi. Agar a

qiymati True, keyin kategoriya tartibli bo'ladi.

• dtype: categoricalDtype

◦ Kategoriya uchun foydalaniladigan kategoriya turi.

43

|  |
| --- |
| **45-bet** |

Keling, toifani yarataylik:

>>> ranglar = pd.Tategorik ([ **'r'** , **'g'** , **'g'** , **'b'** , **'r'** ])

>>> ranglar

[r, g, g, b, r]

Kategoriyalar ( 3 , ob'ekt): [b, g, r]

Ushbu misolda kategoriya o'tgan narsalar asosida qurilgan

ma'lumotlar, xuddi *Series* bilan misolda bo'lgani kabi . Keling, toifani aniq ko'rsatib o'tamiz:

>>> ranglar = pd.Tategorik ([ **'r'** , **'g'** , **'g'** , **'b'** , **'r'** ], kategoriyalar = [ **'r'** ,

**'g'**, **'b'**])

>>> ranglar

[r, g, g, b, r]

Kategoriyalar ( 3 , ob'ekt): [r, g, b]

Agar ma'lumotlar to'plamida mavjud bo'lmagan elementlar bo'lsa

toifaga kirsangiz, ularga NaN qiymatlari beriladi:

>>> ranglar = pd.Tategorik ([ **'r'** , **'g'** , **'g'** , **'b'** , **'r'** , **'y'** , **'o'** ],

toifalar = [ **'r'** , **'g'** , **'b'** ])

>>> ranglar

[r, g, g, b, r, NaN, NaN]

Kategoriyalar ( 3 , ob'ekt ): [r, g, b]

*Kategorik* ob'ektdan siz *ketma-ket* tuzilmani qurishingiz mumkin :

>>> colors\_s = pd.Seriyalar (ranglar)

>>> ranglar\_lari

0

r

1

g

2018-04-02 121 2

g

3

b

4

r

besh

NaN

6

NaN

dip turi: turkum

Kategoriyalar ( 3 , ob'ekt ): [r, g, b]

44

|  |
| --- |
| **46-bet** |

Dropna () funktsiyasi ma'lumotni tozalash uchun ishlatiladi:

>>> colors\_s = pd.Series (ranglar) .dropna ()

>>> ranglar\_lari

0

r

1

g

2018-04-02 121 2

g

3

b

4

r

dip turi: turkum

Kategoriyalar ( 3 , ob'ekt ): [r, g, b]

***DataFrame* tuzilishi bilan ishlash**

*Series-ga* o'xshashlik bilan siz *DataFrame-* ni kategorik bilan yaratishingiz mumkin

ma'lumotlar:

>>> df = pd.DataFrame ({ **'C1'** : list ( **'rrg'** ), **'C2'** : list ( **'rgb'** )},

dtype = **'toifa'** )

>>> df

C1 C2

0

r

r

1

r

g

2018-04-02 121 2

g

b

>>> df.dtypes

C1

toifasi

C2

toifasi

turi: ob'ekt

>>> df [ **'C1'** ]

0

r

1

r

2018-04-02 121 2

g

Nomi: C1, dip turi: toifasi

Kategoriyalar ( 2 , ob'ekt ): [g, r]

45

|  |
| --- |
| **Sahifa 47** |

**3.3.2 Tartib toifalari**

Yuqorida muhokama qilingan toifadagi ma'lumotlar o'z ichiga olmaydi

tartib munosabati, ya'ni qiymatlarni oladigan o'zgaruvchilar uchun

bunday toifalarni haddan ziyod bilan taqqoslash mumkin emas.

Buyurtma munosabati ko'rsatilgan toifalar mavjud, ular

ehtimol filmdagi rol, ta'lim va h.k.

Buyurtmaning ta'rifi bilan ma'lumotlar bazasini yarataylik:

>>> darajasi = pd.Kategorik ([ **'h'** , **'h'** , **'m'** , **'l'** ], kategoriyalar = [ **'l'** ,

**'m'**, **'h'**], buyurilgan = **To'g'ri** )

>>> darajasi

[h, h, m, l]

Kategoriyalar ( 3 , ob'ekt): [l <m <h]

So'nggi qatorga e'tibor bering, deyiladi

toifadagi elementlarning qiymatlari

"ko'proq kamroq". Agar biz buyurtma qilingan = Haqiqiy parametrni ko'rsatmasak, unda uchun

bunday ma'lumotlar to'plamini minimal qiymatdan qidirib bo'lmaydi

maksimal element:

>>> c\_var = pd.Series (pd.Tategorik ([ **'r'** , **'g'** , **'g'** , **'b'** , **'r'** ],

kategoriyalar = [ **'r'** , **'g'** , **'b'** ], buyurilgan = **Yolg'on** ))

>>> c\_var

0

r

1

g

2018-04-02 121 2

g

3

b

4

r

dip turi: turkum

Kategoriyalar ( 3 , ob'ekt): [r, g, b]

46

|  |
| --- |
| **Sahifa 48** |

>>> c\_var.min ()

Traceback (so'nggi qo'ng'iroq oxirgi): ...

**'Buyurtma bo'yicha**toifali **\ n '**.format (op = op))

TypeError: kategoriyalari **emas** buyurdi **uchun** operatsiya daqiqa

.as\_ordered () dan foydalanishingiz mumkin

bitta

Yaratilgan belgilangan daraja uchun bunday operatsiyani bajarish mumkin:

>>> lev\_var = pd.Series (daraja)

>>> chop etish ( **'min: {}, max: {}'** .format (lev\_var.min (), lev\_var.max ()))

**'min: l, max: h'**

Tartibsiz toifalar uchun ob'ektlar darajasida taqqoslashga yo'l qo'yilmaydi,

masalan:

>>> c1 = pd.Series (pd.Tategorik ([ **'r'** , **'g'** , **'b'** , **'r'** ], kategoriyalar = [ **'r'** ,

**'g'**, **'b'**], buyurilgan = **Yolg'on** ))

>>> c2 = pd.Series (pd.Tategorik ([ **'b'** , **'g'** , **'g'** , **'r'** ], kategoriyalar = [ **'r'** ,

**'g'**, **'b'**], buyurilgan = **Yolg'on** ))

>>> c1> c2

Traceback (so'nggi qo'ng'iroq oxirgi):

...

**oshirish**TypeError ( **'Tartibsiz kategoriyalar faqat taqqoslanishi mumkin'**

TypeError: tartibsiz kategoriyalar faqat tenglikni taqqoslay oladi **yoki** taqqoslamaydi

Agar ob'ektlarda *tartib* elementlari bo'lgan ma'lumotlar mavjud bo'lsa

toifasi, keyin hamma narsa to'g'ri bajariladi:

>>> v1 = pd.Series (pd.Tategorik ([ **'h'** , **'m'** , **'l'** , **'h'** ], kategoriyalar = [ **'l'** ,

**'m'**, **'h'**], buyurilgan = **To'g'ri** ))

>>> v2 = pd.Series (pd.Tategorik ([ **'m'** , **'m'** , **'h'** , **'l'** ], kategoriyalar = [ **'l'** ,

**'m'**, **'h'**], buyurilgan = **To'g'ri** ))

>>> v1> v2

0

**To'g'ri**

1

**Yolg'on**

2018-04-02 121 2

**Yolg'on**

3

**To'g'ri**

dip: bool

47

|  |
| --- |
| **49-bet** |

**4-bob. Ma'lumotlardagi bo'shliqlar bilan ishlash**

Ko'pincha, katta miqdordagi ma'lumotlarga tayyorlanadi

tahlil, bo'shliqlar mavjud. Foydalanish imkoniyatiga ega bo'lish uchun

ushbu ma'lumotlardan modellarni yaratadigan mashinani o'rganish algoritmlari,

bu bo'shliqlarni to'ldirish kerak. "Nima to'ldirish kerak?" Degan savolga. Biz emas

chunki biz ushbu kitob doirasida javob beramiz bu ko'proq mavzu bilan bog'liq

mashinada o'rganish va ma'lumotlarni tahlil qilish. Ammo savol "qanday to'ldirish kerak?"

biz javob beramiz va biz *pandas* kutubxonasi yordamida bu muammoni hal qilamiz ,

buning uchun vositalarni taqdim etadi.

**4.1 *Pandalar* va etishmayotgan ma'lumotlar**

Bizning tajribalar uchun, biz bir yaratadi *DataFrame* tuzilishini , deb bo'ladi

bo'shliqlarni o'z ichiga oladi. Buning uchun biz import qilamiz

kutubxonalar:

>>> pandalarni pd **sifatida import qilish**

>>> **dan** io **import** StringIO

Shundan so'ng, *CSV* formatida ob'ekt yarataylik . *CSV* bu eng yuqori ko'rsatkichlardan biridir

oddiy va keng tarqalgan ma'lumotlarni saqlash formatlari

elementlar bir-biridan vergul bilan ajratilgan:

>>> ma'lumotlar = **'narx, son, foiz \ n 1.10, \ n 2.20.51 \ n 3.30,'**

>>> df = pd.read\_csv (StringIO (ma'lumotlar))

Natijada paydo bo'lgan df ob'ekti *bo'shliqlarga* ega bo'lgan *DataFrame* :

>>> df

narxlar soni

0 1 10 NaN

1 2 20 51.0

2 3 30 NaN

48

|  |
| --- |
| **Sahifa 50** |

Bizning misolimizda 0 va 2 indekslari bo'lgan ob'ektlarda ma'lumotlar mavjud emas

**"** foiz **"** maydoni . Yo'qolgan ma'lumotlar NaN sifatida belgilanadi.

Mavjud tuzilishga yana bir ob'ekt (yozuv) qo'shaylik,

**"** hisoblash **"** maydonida qiymat yo'qoladi :

>>> df.loc [ 3 ] = { **'narx'** : 4 , **'hisoblash'** : **yo'q** , **'foiz'** : 26.3 }

>>> df

narxlar soni

0 1.0 10.0 NaN

1 2.0 20.0 51.0

2 3.0 30.0 NaN

3 4.0 NaN 26.3

Birinchidan, dan usullari navbati qilaylik *pandas* kutubxona , qaysi

tuzilmalarda NaN elementlari borligini tezda aniqlashga imkon beradi.

Agar jadval kichik bo'lsa, unda siz kutubxona usulidan foydalanishingiz mumkin

isnull ():

>>> pd.isnull (df)

narxlar soni

0 **Yolg'on Yolg'on To'g'ri**

1 **yolg'on yolg'on**

2 **Yolg'on Yolg'on To'g'ri**

3 **yolg'on haqiqiy yolg'on**

Natijada, biz bir xil o'lchamdagi jadvalni olamiz, lekin joyida

haqiqiy ma'lumotlar, unda teng bool tipidagi elementlar mavjud

Agar maydon qiymati NaN ga teng bo'lmasa, noto'g'ri yoki aks holda True.

49

|  |
| --- |
| **Sahifa 51** |

Bunga qo'shimcha ravishda siz haqida batafsil ma'lumotlarni ko'rishingiz mumkin

info () usuli yordamida ob'ekt:

>>> df.info ()

< **class 'pandas.core.frame.DataFrame'** >

Int64Index: 4 ta yozuv, 0 dan 3 gacha

Ma'lumot ustunlari (jami 3 ta ustun):

narx 4 bo'sh bo'lmagan float64

hisoblash 3 nodavlat null float64

foiz 2 bo'sh bo'lmagan float64

diplar: float64 ( 3 )

xotiradan foydalanish: 128.0 bayt

Masalan, df ob'ekti uchta ustunga ega ekanligini ko'rsatadi (hisoblash, foiz va

narx), barcha ustunlar ustunida esa - NaN emas, in

hisoblash ustunida bitta NaN ob'ekti, foiz maydonida ikkita NaN ob'ekt mavjud.

Aniqlash uchun quyidagi yondashuvdan foydalanishingiz mumkin

yozuvlardagi NaN elementlari soni:

>>> df.isnull (). sum ()

narx 0

hisoblash 1

foiz 2

dip turi: int64

**4.2 Yo'qotilgan ma'lumotlarni almashtirish**

Yo'qolgan ob'ekt ma'lumotlarini aniq bilan almashtirish mumkin

fillna () usuli yordamida raqamli qiymatlar.

50

|  |
| --- |
| **Sahifa 52** |

Tajribalar uchun biz yaratilgan df tuzilmasidan foydalanamiz

oldingi bo'lim:

>>> df

narxlar soni

0 1.0 10.0 NaN

1 2.0 20.0 51.0

2 3.0 30.0 NaN

3 4.0 NaN 26.3

>>> df.fillna ( 0 )

narxlar soni

0 1.0 10.0 0.0

1 2.0 20.0 51.0

2 3.0 30.0 0.0

3 4.0 0.0 26.3

Ushbu usul hozirgi tuzilmani o'zgartirmaydi, strukturani qaytaradi

*Ma'lumot* bazasi NaN qiymatlarini almashtirish bilan mavjud bo'lgan asosida yaratilgan

argument sifatida usulga o'tganlarga. Ma'lumotlar mumkin

o'rtacha ustun bilan to'ldiring:

>>> df.fillna (df.mean ())

narxlar soni

0 1.0 10.0 38.65

1 2.0 20.0 51.00

2 3.0 30.0 38.65

3 4.0 20.0 26.30

Vazifaga qarab, u yoki bu to'ldirish usuli qo'llaniladi

etishmayotgan elementlar, u nolga teng bo'lishi mumkin,

matematik kutish, o'rtacha va boshqalar.

51

|  |
| --- |
| **Sahifa 53** |

NaN elementlarini ma'lum qiymatlar bilan almashtirish uchun siz qilishingiz mumkin

usulda amalga oshiriladigan interpolatsiyadan foydalaning

interpolate (), interpolatsiya algoritmi argument orqali aniqlanadi

usul.

**4.3 Ob'ektlarni / ustunlarni etishmayotgan holda olib tashlash**

**ma'lumotlar**

G'oyib bo'lganlar bilan ishlashda juda keng tarqalgan yondashuv

ma'lumotlar bu yozuvlar (qatorlar) yoki maydonlarni (ustunlarni) o'chirish

bo'shliqlar mavjud.

NaN qiymatlarini o'z ichiga olgan barcha moslamalarni olib tashlash uchun

argumentlarsiz dropna () usulidan foydalaning:

>>> df.dropna ()

narxlar soni

1 2.0 20.0 51.0

Yozuvlar o'rniga siz maydonlarni o'chirib tashlashingiz mumkin, buning uchun siz usulni chaqirishingiz kerak

dropna () o'qi = 1 argument bilan:

>>> df.dropna ( o'q = 1 )

narx

0 1.0

1 2.0

2 3.0

3 4.0

52

|  |
| --- |
| **Sahifa 54** |

*Pandalar* sizga NaN bo'lmagan elementlar soni uchun chegara o'rnatishga imkon beradi. IN

Quyidagi misol barcha ustunlarni o'chirib tashlaydi

NaN bo'lmagan elementlarning soni uchdan kam:

>>> df.dropna ( o'q = 1 , urish = 3 )

narxlar soni

0 1.0 10.0

1 2.0 20.0

2 3.0 30.0

3 4.0 NaN

53

|  |
| --- |
| **Sahifa 55** |

**5-bob. *Pandalardagi* ma'lumotlar tuzilmalari bilan ishlash :**

**yo'q qilish, birlashtirish, kengaytirish, guruhlash**

**5.1 *Pandalar*tuzilishiga elementlar qo'shish**

Umumiy ko'rinishni tuzilishga elementlar qo'shilishi bilan boshlaymiz. Hammasi

harakatlar ikkita tuzilma uchun ko'rib chiqiladi: *Series* va *DataFrame* . Uchun

Keling, biz ishlaydigan tuzilmalarni yaratishdan boshlaymiz.

*Seriya* tarkibi :

>>> s = pd.Seriyalar ([ 1 , 2 , 3 , 4 , 5 ], [ **'A'** , **'B'** , **'C'** , **'D'** , **'E'** ])

*DataFrame* uchun lug'at :

>>> d = { **'rang'** : [ **'qizil'** , **'yashil'** , **'ko'k'** ], **'tezlik'** : [ 56 , 24 , 65 ],

**'jild'**: [ 80 , 65 , 50 ]}

*DataFrame* tuzilishi :

>>> df = pd.DataFrame (d)

>>> df

rang tezligi hajmi

0 qizil 56 80

1 yashil 24 65

2 ko'k 65 50

54

|  |
| --- |
| **Sahifa 56** |

**5.1.1 *Seriyalarga*qo'shish**

Faqatgina *Series* tarkibiga yangi element qo'shish juda oson

ob'ekt uchun yangi indeksni belgilang va elementga qiymat o'rnating.

**"** F **"** indeksi va qiymati 6 bo'lgan s tuzilishga yangi element qo'shaylik :

>>> s [ **'F'** ] = 6

>>> lar

A 1

B 2

C 3

D 4

E 5

F 6

dip turi: int64

**5.1.2 *DataFrame-ga*qo'shilish**

*DataFrame* bilan ishlashda jadvalga qo'shimcha sifatida qo'shishingiz mumkin

ustun yoki butun yozuv. Ustundan boshlaymiz, bu operatsiya juda yaxshi

uchun elementi qo'shib o'xshaydi *Series* : biz maydonda ko'rsatadi

ustun nomini qavs ichiga oladi va har biri uchun qiymatlar ro'yxatini belgilaydi

yozuvlar:

>>> df

rang tezligi hajmi

0 qizil 56 80

1 yashil 24 65

2 ko'k 65 50

55

|  |
| --- |
| **57-bet** |

Har bir yozuv (satr) uchun belgilanadigan **"** tur **"** yangi ustunini qo'shing

ushbu maydonning qiymati:

>>> df [ **'type'** ] = [ **'aylana'** , **'kvadrat'** , **'uchburchak'** ]

>>> df

rang tezligi hajmi turi

0 qizil 56 80 doira

1 yashil 24 65 kvadrat

2 ko'k 65 50 uchburchak

Siz doimiy qiymat ustunini qo'shishingiz mumkin:

>>> df [ **'qiymat'** ] = 7

>>> df

rang tezligi hajmi turi qiymati

0 qizil 56 80 doira 7

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

In *DataFrame* bir ob'ekt qo'shishingiz mumkin *Series* bilan mag'lubiyatga sifatida. Ushbu harakat

ko'proq "ma'lumotlar sintezi" mavzusiga murojaat qiladi, ammo bu erda

bu o'rinli:

>>> new\_row = pd.Series ([ **'sariq'** , 34 , 10 , **'to'rtburchak'** , 7 ], [ **'rang'** ,

**'tezlik'**, **'tovush'**, **'tur'**, **'qiymat'**])

>>> df.append (new\_row, ignore\_index = **True** )

rang tezligi hajmi turi qiymati

0 qizil 56 80 doira 7

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

3 sariq 34 10 to'rtburchak 7

56

|  |
| --- |
| **Sahifa 58** |

**5.2 *Pandalardagi*strukturadan elementlarni olib tashlash**

**5.2.1 *Seriyadan*olib tashlash**

*Series* strukturasidan elementlarni olib tashlash uchun usuldan foydalaning

drop (), bu argument sifatida yorliqlar ro'yxatiga uzatiladi

o'chirish. Shu bilan birga, foydalanishda esda tutish kerak

ushbu usulning sukut bo'yicha joriy tuzilishi o'zgarmaydi, lekin

a *Series* ob'ekt bilan qaytariladi hech saylovlari

yorliqlar.

Tuzilishning dastlabki holati:

>>> lar

A 1

B 2

C 3

D 4

E 5

F 6

dip turi: int64

>>> s\_new = s.drop ([ **'A'** , **'B'** ])

Drop () operatsiyasidan keyin s tuzilishi bir xil bo'lib qoladi:

>>> lar

A 1

B 2

C 3

D 4

E 5

F 6

dip turi: int64

57

|  |
| --- |
| **Sahifa 59** |

S\_new tuzilmasi s elementlaridan tashqari barcha elementlarni o'z ichiga oladi

uning indekslari drop () usuliga o'tgan:

>>> s\_new

C 3

D 4

E 5

F 6

dip turi: int64

Misoldan ko'rinib turibdiki, s ning tuzilishi o'zgarmagan. Drop () usulini chaqirish

s\_new without deb nomlangan yana bir tuzilishga olib keldi

belgilangan narsalar. Agar siz to'g'ridan-to'g'ri o'zini o'zgartirishingiz kerak bo'lsa

tuzilish, keyin qo'shimcha ravishda usulning ichki argumenti

drop () True ga o'rnatilgan:

>>> s.drop ([ **'A'** , **'B'** ], inplace = **True** )

>>> lar

C 3

D 4

E 5

F 6

dip turi: int64

**5.2.2 *DataFrame-*dan o'chirish**

*DataFrame* tuzilmasidan elementlarni olib tashlash uchun ham amal qilinadi

drop () usuli. Namoyish uchun biz df ob'ektidan foydalanamiz,

oldingi bo'limda yaratilgan:

58

|  |
| --- |
| **Sahifa 60** |

>>> df

rang tezligi hajmi turi qiymati

0 qizil 56 80 doira 7

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

>>> df\_new = df.drop ([ 0 ])

>>> df\_new

rang tezligi hajmi turi qiymati

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

Agar siz df tuzilmasini o'zi o'zgartirishingiz kerak bo'lsa, aniqlang

qo'shimcha parametr inplace = True *:*

>>> df.drop ([ 0 ], inplace = **True** )

>>> df

rang tezligi hajmi turi qiymati

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

*DataFrame* - bu ikki o'lchovli jadval bo'lib, undan ko'proq narsani o'chirishingiz mumkin

qatorlar, shuningdek ustunlar. Buning uchun biz o'qni ko'rsatishingiz kerak

biz ishlaymiz, u eksa parametri orqali o'rnatiladi, sukut bo'yicha

o'q = 0, bu satrlar bilan ishlashni anglatadi. Agar siz eksa = 1 ni belgilasangiz, unda bu

keraksiz ustunlarni olib tashlaydi:

>>> df

rang tezligi hajmi turi qiymati

1 yashil 24 65 kvadrat 7

2 ko'k 65 50 uchburchak 7

>>> df.drop ([ **"rang"** , **"qiymati"** ], o'qi = 1 , inplace = **True** )

59

|  |
| --- |
| **Sahifa 61** |

>>> df

tezlik balandligi turi

1 24 65 kvadrat

2 65 50 uchburchak

**5.3 Ma'lumotlarni birlashtirish**

*Pandalar* muammoni hal qilish uchun vositalar to'plamini taqdim etadi

ma'lumotlarni birlashtirish. Bizda shunday imkoniyat bor

oddiygina tuzilmalarni qo'shimcha ishlov bermasdan biriga birlashtirish,

yoki yordamida bu jarayonni yanada aqlli qilish

*pandalar* tomonidan taqdim etilgan funksionallik.

Biz ushbu funksiyani *Series* va uchun alohida ko'rib *chiqmaymiz*

*DataFrame* , barcha imkoniyatlar faqat *DataFrame* uchun ko'rsatiladi .

**5.3.1 *Konkak*usulidan foydalanish**

*Shiqillagan* usul *pandalardagi* tuzilmalarni birlashtirish muammosini hal qiladi *:*

pandas.concat (objs, o'q = 0 , qo'shilish = **'tashqi'** , join\_axes = **yo'q** ,

ignore\_index = **False** , tugmalar = **Yo'q** , darajalar = **Yo'q** , ismlar = **Yo'q** ,

verify\_integrity = **False** , copy = **True** )

Keling, eng muhim dalillarni ko'rib chiqaylik:

• objs: bir array yoki lug'at *Series* , *DataFrame,* yoki *Panel* tuzilmalar .

Combine Birlashtirish uchun tuzilmalar.

• o'qi: 0 - qatorlar, 1 - ustunlar; standart qiymat: 0

◦ Birlashma bajariladigan o'q.

• qo'shilish: {'ichki', 'tashqi'}; standart: 'tashqi'

◦ Birlashtirish operatsiyasining turi, "tashqi" - natijada tuzilish bo'ladi

uzatilgan birlashma natijasi (mantiqiy YOKI)

60

|  |
| --- |
| **62-bet** |

tuzilmalar, "ichki" - natijada paydo bo'lgan tuzilish natijasi bo'ladi

uzatilgan inshootlarning kesishishi (mantiqiy VA).

• ignore\_index: bool; standart: noto'g'ri

◦ To'g'ri - jarayonda indeks qiymati ishlatilmaydi

uyushmalari, Soxta *bo'lgan* ishlatiladi.

Birinchidan, ba'zi *DataFrame* tuzilmalarini yarataylik *:*

dfr1 = pd.DataFrame ({ **'a\_type'** : [ **'a1'** , **'a2'** , **'a3'** ], **'b\_type'** : [ **'b1'** , **'b2'** ,

**'b3'**], **'c\_type'**: [ **'c1'**, **'c2'**, **'c3'**]}, indeks = [ 0 , 1 , 2 ])

>>> dfr1

a\_type b\_type c\_type

0 a1 b1 c1

1 a2 b2 c2

2 a3 b3 c3

>>> dfr2 = pd.DataFrame ({ **'a\_type'** : [ **'a4'** , **'a5'** , **'a6'** ], **'b\_type'** : [ **'b4'** ,

**'b5'**, **'b6'**], **'c\_type'**: [ **'c4'**, **'c5'**, **'c6'**]}, index = [ 3 , 4 , 5 ])

>>> dfr2

a\_type b\_type c\_type

3 a4 b4 c4

4 a5 b5 c5

5 a6 b6 c6

Keling, ushbu ikkita tuzilmani biriga birlashtiramiz:

>>> df1 = pd.concat ([dfr1, dfr2])

>>> df1

a\_type b\_type c\_type

0 a1 b1 c1

1 a2 b2 c2

2 a3 b3 c3

3 a4 b4 c4

4 a5 b5 c5

5 a6 b6 c6

61

|  |
| --- |
| **63-bet** |

Keling, yana bir tuzilma yarataylik va o'qni o'zgartirib, uni birinchisi bilan birlashtiramiz:

>>> dfr3 = pd.DataFrame ({ **'d\_type'** : [ **'d1'** , **'d2'** , **'d3'** ], **'e\_type'** : [ **'e1'** ,

**'e2'**, **'e3'**]})

>>> dfr3

d\_type e\_type

0 d1 e1

1 d2 e2

2 d3 e3

>>> df2 = pd.concat ([dfr1, dfr3], o'q = 1 )

>>> df2

a\_type b\_type c\_type d\_type e\_type

0 a1 b1 c1 d1 e1

1 a2 b2 c2 d2 e2

2 a3 b3 c3 d3 e3

Yakuniy tarkibdagi tarkibiy qismlarni ajratib ko'rsatish uchun,

birlashganda kalitlar parametridan foydalaning:

>>> df3 = pd.concat ([dfr1, dfr2], tugmalar = [ **'dfr1'** , **'dfr2'** ])

>>> df3

a\_type b\_type c\_type

dfr1 0 a1 b1 c1

1 a2 b2 c2

2 a3 b3 c3

dfr2 3 a4 b4 c4

4 a5 b5 c5

5 a6 b6 c6

62

|  |
| --- |
| **64-bet** |

>>> df3.loc [ **'dfr2'** ]

a\_type b\_type c\_type

3 a4 b4 c4

4 a5 b5 c5

5 a6 b6 c6

Olingan tuzilish birlashish natijasida bo'lishi kerak bo'lsa

(mantiqiy YOKI), keyin qo'shilish parametriga qiymat berilishi kerak

"tashqi".

>>> dfr4 = pd.DataFrame ({ **'d\_type'** : [ **'d2'** , **'d3'** , **'d4'** ], **'e\_type'** : [ **'e2'** ,

**'e3'**, **'e4'**]}, indeks = [ 1 , 2 , 3 ])

>>> dfr4

d\_type e\_type

1 d2 e2

2 d3 e3

3 d4 e4

>>> df4 = pd.concat ([dfr1, dfr4], o'q = 1 , qo'shilish = **'tashqi'** )

>>> df4

a\_type b\_type c\_type d\_type e\_type

0 a1 b1 c1 NaN NaN

1 a2 b2 c2 d2 e2

2 a3 b3 c3 d3 e3

3 NaN NaN NaN d4 e4

63

|  |
| --- |
| **Sahifa 65** |

Olingan struktura kesishishning natijasi bo'lishi kerak bo'lsa

(mantiqiy VA), keyin qo'shilish parametriga qiymat berilishi kerak

"ichki" *:*

>>> df5 = pd.concat ([dfr1, dfr4], o'q = 1 , qo'shilish = **'ichki'** )

>>> df5

a\_type b\_type c\_type d\_type e\_type

1 a2 b2 c2 d2 e2

2 a3 b3 c3 d3 e3

**5.3.2 *Ma'lumotlar bazasi uslubidagi* yondashuvdan foydalanish**

Ushbu yondashuvning mohiyati shundaki, juda tezkor usul qo'llaniladi

mafkuraviy jihatdan o'xshash bo'lgan ma'lumotlar tuzilmalarini birlashtirish

relyatsion ma'lumotlar bazalari bilan operatsiyalar. *Pandalar* tarkibi kiritilgan

ushbu funktsiyani ifodalovchi birlashma funktsiyasi:

pandas.merge (chapda, o'ngda, qanday qilib = **"ichki"** , ustida = **Yo'q** , chapda\_on = **Yo'q** ,

right\_on = **Yo'q** , chap\_index = **False** , right\_index = **False** , sort = **True** ,

qo'shimchalar = ( **'\_x'** , **'\_y'** ), copy = **True** , ko'rsatkich = **False** , validate = **Yo'q** )

Keling, ushbu funktsiya argumentlarini tahlil qilaylik (quyidagi ro'yxatda)

eng ko'p ishlatiladigan argumentlar ko'rsatilgan):

• chapda: *DataFrame*

*Data DataFrame* tuzilmasi "chapda" .

• o'ngda: *DataFrame*

Data "to'g'ri" *DataFrame* tuzilishi.

• qanday: {'chap', 'o'ng', 'tashqi', 'ichki'}; qiymati

sukut: "ichki"

The Birlashtirish usullaridan biri: 'chap' *,* 'o'ng' *,* 'tashqi' *,*

"ichki" *:*

64

|  |
| --- |
| **Sahifa 66** |

▪ "chap" - bu qachon "LEFT OUTER JOIN" *SQL* operatsiyasining analogidir

bu faqat chapdagi tugmachalardan foydalanadi

*DataFrame.*

▪ "o'ng" - "RIGHT OUTER JOIN" *SQL* operatsiyasining analogi -

o'ng *DataFrame-dan* kalitlardan *foydalaniladi* .

▪ *"* tashqi" - "FULL OUTER JOIN" *SQL* operatsiyasining analogi -

o'ng va chap tugmalar birikmasi ishlatiladi

*DataFrame* .

▪ "ichki" - "INNER JOIN" *SQL* operatsiyasining analogi -

o'ng va chap tugmachalarning kesishishi ishlatiladi

*DataFrame* .

• ro'yxat

Combine Birlashtiriladigan ustunlar nomlari ro'yxati, ustunlar shart

ikkala chap va o'ng *DataFrame-ni kiriting* .

• chap\_on: ro'yxat

Chap ning ustunlar ◦ ro'yxati *DataFrame* bo'lish

kalit sifatida ishlatilishi mumkin.

• o'ng\_on: ro'yxat

Data o'ng DataFrame-dan ustunlar ro'yxati

kalit sifatida ishlatilishi mumkin.

• chap\_index: bool; standart qiymat: noto'g'ri

◦ Agar parametr rost bo'lsa, u holda indeks (yorliqlar)

satrlar) chapdagi *DataFrame-* dan kalit (lar) sifatida

uyushmalar.

• right\_index: bool; standart qiymat: noto'g'ri

◦ Agar parametr rost bo'lsa, u holda indeks (yorliqlar)

satrlar) o'ng *DataFrame-* dan kalit (lar) sifatida

uyushmalar.

65

|  |
| --- |
| **67-bet** |

• sort: bool; standart: noto'g'ri

◦ Agar parametr rost bo'lsa, u holda olingan *DataFrame-* dagi ma'lumotlar

leksikografik tartibda saralanadi.

Funktsiyadan qanday foydalanishingiz haqida bir nechta misollarni ko'rib chiqamiz

birlashtirish ().

Uchta *DataFrame* yarataylik :

dfr1 = pd.DataFrame ({ **'k'** : [ **'k1'** , **'k2'** , **'k3'** ], **'a\_type'** : [ **'a1'** , **'a2'** , **'a3'** ],

**'b\_type'**: [ **'b1'**, **'b2'**, **'b3'**]})

>>> dfr1

k a\_type b\_type

0 k1 a1 b1

1 k2 a2 b2

2 k3 a3 b3

>>> dfr2 = pd.DataFrame ({ **'k'** : [ **'k1'** , **'k2'** , **'k3'** ], **'c\_type'** : [ **'c1'** , **'c2'** ,

**'c3'**]})

>>> dfr2

k c\_type

0 k1 c1

1 k2 c2

2 k3 c3

>>> dfr3 = pd.DataFrame ({ **'k'** : [ **'k0'** , **'k1'** , **'k2'** ], **'c\_type'** : [ **'c1'** , **'c2'** ,

**'c3'**]})

>>> dfr3

k c\_type

0 k0 c1

1 k1 c2

2 k2 c3

66

|  |
| --- |
| **Sahifa 68** |

Keling, ularni birlashtirish () orqali birlashtiramiz, biz kalit sifatida foydalanamiz

k ustun *:*

>>> dfm1 = pd.merge (dfr1, dfr2, ustiga = **"K"** )

>>> dfm1

k a\_type b\_type c\_type

0 k1 a1 b1 c1

1 k2 a2 b2 c2

2 k3 a3 b3 c3

Qanday qilib = 'chap' dan foydalanib misol:

>>> dfm2 = pd.merge (dfr1, dfr3, qanday = **"left"** , haqida = **"K"** )

>>> dfm2

k a\_type b\_type c\_type

0 k1 a1 b1 c2

1 k2 a2 b2 c3

2 k3 a3 b3 NaN

= "Right" dan foydalangan holda misol:

>>> dfm3 = pd.merge (dfr1, dfr3, qanday = **"o'ng"** , haqida = **"K"** )

>>> dfm3

k a\_type b\_type c\_type

0 k1 a1 b1 c2

1 k2 a2 b2 c3

2 k0 NaN NaN c1

= "Tashqi" dan foydalangan holda misol:

>>> dfm4 = pd.merge (dfr1, dfr3, qanday = **'tashqi "** , haqida = **" K "** )

>>> dfm4

k a\_type b\_type c\_type

0 k1 a1 b1 c2

1 k2 a2 b2 c3

2 k3 a3 b3 NaN

3 k0 NaN NaN c1

67

|  |
| --- |
| **69-bet** |

Qanday qilib = "ichki" dan foydalanilgan misol:

>>> dfm5 = pd.merge (dfr1, dfr3, qanday = **'ichki "** , haqida = **" K "** )

>>> dfm5

k a\_type b\_type c\_type

0 k1 a1 b1 c2

1 k2 a2 b2 c3

68

|  |
| --- |
| **70-bet** |

**6-bob. Ma'lumotlarning tashqi manbalari bilan ishlash**

**6.1 *CSV*formatidagi ma'lumotlar bilan ishlash**

*CSV* ( *vergul bilan ajratilgan qiymatlar* - *vergul bilan ajratilgan* qiymatlar)

saqlashning eng mashhur formatlaridan biridir

jadval ma'lumotlari - bu matnli hujjat

ma'lumotlar elementlari vergul bilan ajratilgan va fayl satrlari ajratilgan

jadvaldagi yozuvlar.

**6.1.1 Ma'lumotlarni o'qish**

*CSV* fayllaridan ma'lumotlarni *pandalarga* yuklash uchun ushbu usuldan foydalaning

read\_csv (). Undagi argumentlar soni etarlicha katta, dan

eng ko'p ishlatiladigan quyidagilar:

• filepath\_or\_buffer: str[1](https://translate.googleusercontent.com/translate_f#70) .

Data Ma'lumotlarni o'z ichiga olgan faylga yoki buferga yo'l

*CSV* formati .

• sep: str; standart qiymat: ','

◦ Ajratuvchi, sukut bo'yicha ',', chunki uchun *CSV* ma'lumotlari

vergul bilan ajratilgan. Ko'pincha imkoniyat bor

ma'lumotlar yorliqlar bilan ajratilganda, *TSV-*

fayllar, agar siz ushbu formatdan foydalansangiz, unda sizga kerak

sep parametrini '\ t' ga o'rnating.

• header: int yoki int'ov ro'yxati; standart qiymat: 0

◦ Yuklangan ustunlar nomlarini o'z ichiga olgan qator raqami

jadvallar. Odatiy bo'lib, header = 0. Agar header = None bo'lsa, unda ismlar

ustunlar nomlar parametrida o'tkazilishi mumkin.

1 Bundan tashqari mumkin: *pathlib.Path* , *py.\_path.local.LocalPath* yoki *read ()* usuli bo'lgan har qanday ob'ekt

69

|  |
| --- |
| **71-bet** |

• nomlar: massiv; standart qiymat: Yo'q

◦ Jadval ustunlari nomlari ro'yxati, agar faylda mavjud bo'lmasa ishlatiladi

ustun nomlari bo'lgan satrlar va sarlavha parametri Yo'q.

*CSV* ma'lumotlarini to'g'ridan-to'g'ri *Python-da* yaratish mumkin

dastur yoki fayldan yuklash. Birinchi variantdan boshlaylik. Uchun

bizga kerak bo'lgan kutubxonalarni yuklashni boshlang:

>>> pandalarni pd **sifatida import qilish**

>>> **dan** io **import** StringIO

Endi Keling ma'lumotlarni o'z ichiga olgan mag'lubiyatga yaratish *CSV* formatida va

*Ularning* asosida *DataFrame yarataylik* :

>>> csv\_d1 = **'col\_A, col\_B, col\_C \ n a1, b1, c1 \ n a2, b2, c2'**

>>> df = pd.read\_csv (StringIO (csv\_d1))

>>> df

col\_A col\_B col\_C

0 a1 b1 c1

1 a2 b2 c2

Yuqoridagi misolda ma'lumotlar to'plamining birinchi qatorida

jadval ustun sarlavhalarini o'z ichiga oladi. Ammo bu har doim ham shunday emas. Quyida

ma'lumotlar jadvalida ismlar haqida ma'lumot mavjud bo'lmaganda beriladi

ustunlar, agar bu parametr parametr bo'lsa, tuzatilishi mumkin

None-ni tayinlang va kerakli parametrlarni nomlar parametri orqali o'tkazing:

>>> csv\_d2 = **'a1, b1, c1 \ n a2, b2, c2 \ n a3, b3, c3'**

>>> df1 = pd.read\_csv (StringIO (csv\_d2), sarlavha = **Yo'q** , ismlar = [ **'type\_a'** ,

**'type\_b'**, **'type\_c'**])

>>> df1

type\_a type\_b type\_c

0 a1 b1 c1

1 a2 b2 c2

2 a3 b3 c3

70

|  |
| --- |
| **72-bet** |

Agar *CSV* ma'lumotlar fayli diskda bo'lsa, unda buning uchun

yuklash uchun birinchi parametr sifatida fayl nomini berish kifoya

read\_csv () usuli:

>>> df3 = pd.read\_csv ( **'c: /test.csv'** )

>>> df3

Ism yoshi

Shahar

0 Anna 29 Moskva

1 Yuhanno 21 Nyu-York

2 Ivan 18

Tomsk

3 Mayk 32 Los-Anjeles

Agar fayl hajmi juda katta bo'lsa va imkonsiz bo'lsa

bir vaqtning o'zida xotiraga yuklang ( *DataFrame* ), tomonidan yuklashni *ishlating*

qismlar. Buning uchun qo'shimcha ravishda read\_csv () funktsiyasiga o'tishingiz kerak

satrlar soni ko'rsatilgan chunksize parametri,

bu bir qism ichida hisoblanishi kerak. Ushbu yondashuv imkon beradi

*DataFrames* bilan iteratorlar singari ishlash :

>>> df\_chunks = pd.read\_csv ( **'c: /test.csv'** , chunksize = 2 )

>>> **uchun** I, bo'lak **yilda** Enumerate (df\_chunks):

chop etish ( **f'Chunk # {i} '** )

chop etish (qism)

*# 0* qism

Yoshi shahar

0 Anna 29 Moskva

1 Yuhanno 21 Nyu-York

*№1* qism

Ism yoshi

Shahar

2 Ivan 18

Tomsk

3 Mayk 32 Los-Anjeles

71

|  |
| --- |
| **73-bet** |

>>> df\_chunks = pd.read\_csv ( **'c: /test.csv'** , chunksize = 2 )

>>> chop etish (df\_chunks.get\_chunk ())

Yoshi shahar

0 Anna 29 Moskva

1 Yuhanno 21 Nyu-York

>>> chop etish (df\_chunks.get\_chunk ())

Ism yoshi

Shahar

2 Ivan 18

Tomsk

3 Mayk 32 Los-Anjeles

**6.1.2 Ma'lumotlarni yozib olish**

*CSV* formatida ma'lumotlarni yozish uchun to\_csv () usuli qo'llaniladi.

Keling, foydali bo'lishi mumkin bo'lgan ba'zi dalillarni ko'rib chiqaylik:

• path\_or\_buf: str yoki fayl ushlagichi; standart qiymat: Yo'q

The Ma'lumotlar saqlanadigan fayl yoki bufer nomi

*CSV* formati .

• sep: str; sukut bo'yicha: ','

◦ Mahsulotni ajratuvchi, sukut bo'yicha sep = ','.

• header: bool yoki satrlar ro'yxati; standart qiymat: To'g'ri

◦ Agar rost qiymati sifatida qabul qilingan bo'lsa, unda birinchisi

mag'lubiyat strukturadan olingan ustun nomlarini yozadi

ma'lumotlar. Agar satrlar ro'yxati berilgan bo'lsa, u holda ustunlar nomlari

undan olinadi.

• kodlash: str

◦ kodlash. Agar siz *Python* 3 dan foydalanayotgan bo'lsangiz , u holda sukut bo'yicha

"utf-8" ga teng.

72

|  |
| --- |
| **74-bet** |

Yozish uchun ma'lumotlarni tayyorlaylik:

>>> csv\_data = **'col\_A, col\_B, col\_C \ n a1, b1, c1 \ n a2, b2, c2'**

>>> df = pd.DataFrame (StringIO (csv\_data))

Endi df strukturasining tarkibini tmp.csv deb nomlangan faylga yozamiz:

>>> df.to\_csv ( **'tmp.csv'** )

**6.2 *JSON*ma'lumotlari bilan ishlash**

Dasturlash tilining rivojlanishi va keng tarqalishi bilan

*JavaScript* ma'lumotlar formati sifatida mashhur bo'lib, u organik ravishda o'sdi

ushbu tilga mos keladi - *JSON ( JavaScript ob'ekti belgisi)* . Batafsil

ushbu format haqida Vikipediyada o'qishingiz mumkin

*(*[*https://ru.wikipedia.org/wiki/JSON*](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=https://ru.wikipedia.org/wiki/JSON) ).

**6.2.1 Ma'lumotlarni o'qish**

Read\_json () usuli *JSON* ma'lumotlarini o'qish uchun ishlatiladi.

Keling, ushbu usul uchun eng ko'p ishlatiladigan argumentlarni ko'rib chiqaylik:

• path\_or\_buf: *JSON* qatori yoki fayl; standart qiymat:

Yo'q

◦ *JSON-ga* yo'l (bu diskdagi fayl yoki *URL bo'lishi mumkin* )

fayl yoki mazmuni haqiqiy *JSON* bo'lgan satr .

• orient: str; standart qiymat: Yo'q

◦ Yuklangan *JSON* bo'lishi mumkin bo'lgan yo'nalish

u bo'lishi kerak bo'lgan *pandas* ma'lumotlar tuzilishiga aylantirildi

ma'lum bir tur. Quyidagi mumkin bo'lgan qiymatlar

orient va ularga mos keladigan *JSON:*

▪ 'split': tuzilishga ega lug'at {index -> [index],

ustunlar -> [ustunlar], ma'lumotlar -> [qiymatlar]}

73

|  |
| --- |
| **Sahifa 75** |

▪ "yozuvlar": tuzilishga ega ro'yxat

[{ustun ->

qiymat}, ..., {ustun -> qiymat}]

▪ 'indeks': tuzilishga ega lug'at {indeks -> {ustun ->

qiymat}}

▪ 'ustunlar': tuzilishga ega lug'at {ustun -> {indeks ->

qiymat}}

▪ 'qadriyatlar': qiymatlar massivi.

• typ: yoziladigan ob'ekt turi; standart: 'ramka'

◦ *Pandalar* tuzilish turi , 'ketma-ket' *seriyali* , 'ramka' bu

*DataFrame* . Matn qiymatiga qarab siz foydalanishingiz mumkin

ma'lum yo'nalish qiymatlari. Agar typ = 'series' bo'lsa, u holda yo'naltiriladi

'split', *'* yozuvlar' yoki 'indeks' bo'lishi mumkin, agar typ = 'frame',

keyin yo'nalishni quyidagi ro'yxatdan tanlash kerak: 'split',

'yozuvlar', *'* indeks', 'ustunlar', 'qiymatlar'.

Endi *JSON* fayllari va buferlarini o'qish bilan bir oz mashq qilaylik . IN

ma'lumotlar *JSON-* da qanday formatlanishiga bog'liq

fayl yoki bufer, u yoki bu parametr qiymati ishlatiladi

read\_json () usulining yo'nalishi.

orient = "bo'linish"

Orient = 'split' uchun *JSON* ma'lumotlar *formati* o'xshash bo'lishi kerak

quyida bayon qilinganidek:

{

'ustunlar' : [ 'col\_A' , 'col\_B' , 'col\_C' ],

'indeks' : [ 0 , 1 ],

'ma'lumotlar' : [[ 'a1' , 'b1' , 'c1' ], [ 'a2' , 'b2' , 'c2' ]]

}

74

|  |
| --- |
| **Sahifa 76** |

*JSON* o'qish uchun kod :

>>> json\_buf = **'{' ustunlar ': [' col\_A ',' col\_B ',' col\_C '],' index ':**

**[0,1], 'ma'lumotlar': [['a1', 'b1', 'c1'], ['a2', 'b2', 'c2']]} '**

>>> df1 = pd.read\_json (json\_buf, orient = **'split'** )

>>> df1

col\_A col\_B col\_C

0 a1 b1 c1

1 a2 b2 c2

orient = 'yozuvlar'

*JSON* ma'lumotlari:

[

{

"col\_A" : "a1" ,

'col\_B' : 'b1' ,

'col\_C' : 'c1'

} ,

{

"col\_A" : "a2" ,

'col\_B' : 'b2' ,

'col\_C' : 'c2'

}

]

*JSON* o'qish uchun kod :

>>> json\_buf2 = **'[{' col\_A ':' a1 ',' col\_B ':' b1 ',' col\_C ':' c1 '}, {' col\_A ':**

**'a2', 'col\_B': 'b2', 'col\_C': 'c2'}] '**

>>> df2 = pd.read\_json (json\_buf2, orient = **'yozuv'** )

>>> df2

col\_A col\_B col\_C

0 a1 b1 c1

1 a2 b2 c2

75

|  |
| --- |
| **Sahifa 77** |

orient = 'indeks'

*JSON* ma'lumotlari:

{

0 : { 'col\_A' : 'a1' , 'col\_B' : 'b1' , 'col\_C' : 'c1' }

1 : { 'col\_A' : 'a2' , 'col\_B' : 'b2' , 'col\_C' : 'c2' }

}

*JSON* o'qish uchun kod :

>>> json\_buf3 = **'{' 0 ': {' col\_A ':' a1 ',' col\_B ':' b1 ',' col\_C ':' c1 '},' 1 ':**

**{'col\_A': 'a2', 'col\_B': 'b2', 'col\_C': 'c2'}} '**

>>> df3 = pd.read\_json (json\_buf3, orient = **'indeks'** )

>>> df3

col\_A col\_B col\_C

0 a1 b1 c1

1 a2 b2 c2

orient = 'ustunlar'

*JSON* ma'lumotlari:

{

'col\_A' : { '0' : 'a1' , '1' : 'a2' } ,

'col\_B' : { '0' : 'b1' , '1' : 'b2' } ,

'col\_C' : { '0' : 'c1' , '1' : 'c2' }

}

*JSON* o'qish uchun kod :

>>> json\_buf4 = **'{' col\_A ': {' 0 ':' a1 ',' 1 ':' a2 '},' col\_B ': {' 0 ':' b1 ',' 1 ':**

**'b2'}, 'col\_C': {'0': 'c1', '1': 'c2'}} '**

>>> df4 = pd.read\_json (json\_buf4, orient = **'ustunlar'** )

>>> df4

col\_A col\_B col\_C

0 a1 b1 c1

1 a2 b2 c2

76

|  |
| --- |
| **78-bet** |

orient = 'qiymatlar'

*JSON* ma'lumotlari:

[

[ 'a1' , 'b1' , 'c1' ] ,

[ 'a2' , 'b2' , 'c2' ]

]

*JSON* o'qish uchun kod :

>>> json\_buf5 = **'[[' a1 ',' b1 ',' c1 '], [' a2 ',' b2 ',' c2 ']]'**

>>> df5 = pd.read\_json (json\_buf5, orient = **'qiymatlar'** )

>>> df5

0 1 2

0 a1 b1 c1

1 a2 b2 c2

**6.2.2 Ma'lumotlarni yozib olish**

*JSON* bilan ishlashda ko'pincha konvertatsiya qilish kerak bo'ladi

ushbu formatdagi tayyor ma'lumotlar tuzilmalari. Buning uchun ishlatiladi

to\_json () funktsiyasi.

Ushbu usulning eng muhim ikkita argumenti path\_or\_buf va

orient, ularning maqsadi read\_json () usuli bilan bir xil, faqat hozir

biz ma'lumotlarni yozib olish haqida, ya'ni. biz faylga yoki buferga ishora qilamiz

bu ma'lumotlarni joylashtiradi.

Oldingi qismdan *DataFrame-* ni olamiz :

>>> d = { **'rang'** : [ **'qizil'** , **'yashil'** , **'ko'k'** ], **'tezlik'** : [ 56 , 24 , 65 ],

**'jild'**: [ 80 , 65 , 50 ]}

>>> df = pd.DataFrame (d)

77

|  |
| --- |
| **79-bet** |

>>> df

rang tezligi hajmi

0 qizil 56 80

1 yashil 24 65

2 ko'k 65 50

Qanday tashkil qarab *JSON* fayli (bufer), biz istaymiz

olish, siz orientatsiya parametrini mos ravishda o'rnatishingiz kerak

qiymati, turli xil variantlarni ko'rib chiqing.

orient = "split":

>>> json\_split = df.to\_json ( orient = **'split'** )

>>> json\_split

**'{' ustunlar ': [' rang ',' tezlik ',' tovush '],' indeks ': [0,1,2],' ma'lumotlar ':**

**[['qizil', 56.80], ['yashil', 24.65], ['ko'k', 65.50]]} '**

orient = 'yozuvlar':

>>> json\_records = df.to\_json ( orient = **'yozuvlar'** )

>>> json\_records

**'[{' color ':' red ',' speed ': 56,' volume ': 80},**

**{'rang': 'yashil', 'tezlik': 24, 'tovush': 65},**

**{'color': 'blue', 'speed': 65, 'volume': 50}] '**

orient = 'indeks':

>>> json\_index = df.to\_json ( orient = **'indeks'** )

>>> json\_index

**'{' 0 ': {' rang ':' qizil ',' tezlik ': 56,' tovush ': 80},' 1 ':**

**{'rang': 'yashil', 'tezlik': 24, 'tovush': 65}, '2':**

**{'rang': 'ko'k', 'tezlik': 65, 'tovush': 50}} '**

78

|  |
| --- |
| **80-bet** |

orient = 'ustunlar':

>>> json\_columns = df.to\_json ( orient = **'ustunlar'** )

>>> json\_columns

**'{' color ': {' 0 ':' red ',' 1 ':' green ',' 2 ':' blue '},' speed ':**

**{'0': 56, '1': 24, '2': 65}, 'jild': {'0': 80, '1': 65, '2': 50}} '**

orient = 'qadriyatlar':

>>> json\_values ​​= df.to\_json ( orient = **'qiymatlar'** )

>>> json\_values

**'[[' qizil ', 56.80], [' yashil ', 24.65], [' ko'k ', 65.50]]'**

**6.3 *Excel* fayllari bilan ishlash**

Asosiy xususiyatlari da Keling qarash *pandas* beradi uchun

*Excel* fayllari bilan ishlash .

**6.3.1 Ma'lumotlarni o'qish**

Ma'lumotlarni o'qish uchun read\_excel () usuli qo'llaniladi. U qila oladi

*Excel* 2003 formatidagi fayllar bilan (kengaytma *.xls* ) va bilan ishlash

*Excel* 2007 formatidagi fayllar (kengaytma *.xlsx* ). Tajribalar uchun

*"test.xls"* nomi bilan *Excel* 2003 formatida *Excel* faylini yarating , qo'shing

unda *"Sheet1"* , *"Sheet2"* ikkita varaq *mavjud* (6.1 va 6.2-jadvallarga qarang).

**6.1-jadval - Tarkibni *Sayfa1'e* ning *test.xls***

*qizil*

17

1

*ko'k*

35

2018-04-02 121 2

*oq*

42

3

*sariq*

63

4

*yashil*

53

besh

79

|  |
| --- |
| **81-sahifa** |

**Table 6,2 - Tarkibni *Sayfa2* ning *test.xls***

3

*a*

*a1*

besh

*h*

*b1*

1

*d*

*c1*

3

*s*

*d1*

6

*a*

*e1*

Keling, *"Sheet1"* varag'ini o'qib chiqamiz va natijada olingan ma'lumotlarni *DataFrame-ga joylashtiramiz* :

>>> df\_xls\_sheet1 = pd.read\_excel ( **"c: \\ test.xls"** , sheetname = **'tablo1 »** ,

sarlavha = **yo'q** )

>>> df\_xls\_sheet1

0 1 2

0 qizil 17 1

1 ko'k 35 2

2 oq 42 3

3 sariq 63 4

4 yashil 53 5

Sarlavha parametriga e'tibor bering. Odatiy bo'lib, u nolga teng -

bu ustun satrlari 0 satrda ekanligini anglatadi.

bizning holatimizda jadvallarda sarlavha yo'q, shuning uchun sarlavha parametri kerak

qiymatini belgilang Yo'q.

80

|  |
| --- |
| **Sahifa 82** |

Keling, *"Sheet2"* ning mazmunini o'qing *:*

>>> df\_xls\_sheet2 = pd.read\_excel ( **"c: \\ test.xls"** , sheetname = **'Sayfa2 »** ,

sarlavha = **yo'q** )

>>> df\_xls\_sheet2

0 1 2

0 3 a a1

1 5 soat b1

2 1 d c1

3 3 s d1

4 6 a e1

*Excel* fayli bilan ishlash uchun siz ExcelFile sinfidan foydalanishingiz mumkin,

ob'ektlari diskdagi ma'lum bir fayl bilan bog'liq.

Ushbu sinfning ob'ektlari kontekst menejerlari hisoblanadi

qurilish bilan ishlash mumkin:

>>> **bilan** pd.ExcelFile ( **"C: \\ test.xls»** ) **deb** Excel:

DF1 = pd.read\_excel (Excel, sheetname = **'tablo1'** , header = **Hech** )

chop etish (df1)

0 1 2

0 qizil 17 1

1 ko'k 35 2

2 oq 42 3

3 sariq 63 4

4 yashil 53 5

81

|  |
| --- |
| **83-bet** |

Oddiy holatda, ExcelFile bilan ishlash quyidagicha ko'rinishi mumkin:

>>> excel = pd.ExcelFile ( **'c: \\ test.xls'** )

>>> df2 = pd.read\_excel (Excel, sheetname = **'Sayfa2'** , header = **Hech** )

>>> df2

0 1 2

0 3 a a1

1 5 soat b1

2 1 d c1

3 3 s d1

4 6 a e1

**6.3.2 Ma'lumotlarni yozib olish**

*Excel* fayliga ma'lumotlarni yozish uchun avval tayyorlanamiz

tegishli *DataFrame* . Keling, bo'limdagi misoldan foydalanaylik,

*JSON-ga* bag'ishlangan :

>>> json\_buf = **'[[' a1 ',' b1 ',' c1 '], [' a2 ',' b2 ',' c2 ']]'**

>>> df = pd.read\_json (json\_buf, orient = **'qiymatlar'** )

>>> df

0 1 2

0 a1 b1 c1

1 a2 b2 c2

Yozish to\_excel () usuli yordamida amalga oshiriladi:

>>> df.to\_excel ( **'test\_excel.xlsx'** , sheet\_name = **'Sheet1'** )

Natijada, biz oladi *Excel* nomli fayl *test\_excel.xlsx* ega bo'lgan,

uning *varaqasi* deb nomlangan bitta varaq bo'ladi, uning mazmuni

6.3-jadvalda keltirilgan.

82

|  |
| --- |
| **84-bet** |

**Jadval 6.3 - Tarkibni *Sayfa1'e* ning *test\_excel.xlsx***

0

1

2018-04-02 121 2

0

*a1*

*b1*

*c1*

1

*a2*

*b2*

*c2*

Agar sarlavha parametri Yo'q deb o'rnatilgan bo'lsa, u holda jadval o'z ichiga oladi

etishmayotgan ustun sarlavhalari:

>>> df.to\_excel ( **'test\_excel1.xlsx'** , sheet\_name = **'Sheet1'** , header = **None** )

*Test\_excel1.xlsx* faylining tarkibi 6.4-jadvalda keltirilgan.

**Table 6,4 - Tarkibni *Sayfa1'e* ning *test\_excel1.xlsx***

0

*a1*

*b1*

*c1*

1

*a2*

*b2*

*c2*

Satrlarning indekslarini (nomlarini) olib tashlash uchun sizga qo'shimcha kerak

parametr indeksini qo'shish = False:

>>> df.to\_excel ( **'test\_excel2.xlsx'** , sheet\_name = **'Sheet1'** , header = **None** ,

indeks = **Yolg'on** )

Natijada, *test\_excel2.xlsx* faylini 6.5-jadval ma'lumotlari bilan olamiz *.*

**Table 6,5 - Tarkibni *Sayfa1'e* ning *test\_excel2.xlsx***

*a1*

*b1*

*c1*

*a2*

*b2*

*c2*

83

|  |
| --- |
| **Sahifa 85** |

**7-bob. Ma'lumotlar bilan ishlash**

**7.1 Arifmetik amallar**

*Pandalar* ma'lumotlar tuzilmalari qo'shilishi, chiqarilishi, ko'paytirilishi va

ajratish (element elementga qarab).

Birinchidan, ikkita *DataFrame* tuzilishini yarataylik *:*

>>> json\_buf1 = **'[[' 10 ',' 20 ',' 30 '], [' 40 ',' 50 ',' 60 ']]'**

>>> df1 = pd.read\_json (json\_buf1, orient = **'qiymatlar'** )

>>> json\_buf2 = **'[[' 12 ',' 24 ',' 14 '], [' 16 ',' 54 ',' 25 ']]'**

>>> df2 = pd.read\_json (json\_buf2, orient = **'qiymatlar'** )

>>> df1

0 1 2

0 10 20 30

1 40 50 60

>>> df2

0 1 2

0 12 24 14

1 16 54 25

Quyida ko'rib chiqilgan usullar strukturaning o'zini o'zgartirmaydi; ular

alohida saqlanishi mumkin bo'lgan yangi tuzilmani qaytaring

o'zgaruvchan.

Add () usuli tuzilmalarni qo'shish uchun ishlatiladi:

>>> df1.add (df2)

0 1 2

0 22 44 44

1 56 104 85

84

|  |
| --- |
| **Sahifa 86** |

Siz strukturaning elementlariga doimiy qo'shishingiz mumkin:

>>> df1.add ( 5 )

0 1 2

0 15 25 35

1 45 55 65

Chiqish sub () usuli yordamida amalga oshiriladi:

>>> df1.sub (df2)

0 1 2

0 - 2 - 4 16

1 24 - 4 35

>>> df1.sub ( 7 )

0 1 2

0 3 13 23

1 33 43 53

Mul () usuli tuzilmalarni ko'paytirish uchun ishlatiladi:

>>> df1.mul (df2)

0 1 2

0 120 480 420

1 640 2700 1500

>>> df1.mul ( 2 )

0 1 2

0 20 40 60

1 80 100 120

85

|  |
| --- |
| **Sahifa 87** |

Div () usuli bo'linish uchun ishlatiladi:

>>> df1.div (df2)

0

1

2018-04-02 121 2

0 0.833333 0.833333 2.142857

1 2.500000 0.925926 2.400000

>>> df1.div ( 2 )

0 1 2

0 5.0 10.0 15.0

1 20.0 25.0 30.0

**7.2 Mantiqiy amallar**

Mavjud tuzilmalardan yangilarini qurish uchun foydalanish mumkin, ularning elementlari

mantiqiy o'zgaruvchilar bo'ladi, ularning qiymatlari quyidagicha aniqlanadi

dastlabki tuzilish elementi ma'lum bir shartni qondiradimi

yoki yo'q.

Oldingi qismdan df2 tuzilishini olaylik:

>>> json\_buf2 = **'[[' 12 ',' 24 ',' 14 '], [' 16 ',' 54 ',' 25 ']]'**

>>> df2 = pd.read\_json (json\_buf2, orient = **'qiymatlar'** )

>>> df2

0 1 2

0 12 24 14

1 16 54 25

Df2 strukturasining elementlarini 20 dan katta qiymatlari bilan aniqlaylik:

>>> df2> 20

0 1 2

0 **yolg'on haqiqiy yolg'on**

1 **Yolg'on To'g'ri**

86

|  |
| --- |
| **88-bet** |

*Pandalar* ma'lumotlar tuzilmalarini katlama vositalari bilan ta'minlaydi

qisqacha ma'lumot olish. Operatsiyani bajarish uchun “mantiqiy

YoKI "qatorlarida yoki ustunlarida any () usulidan foydalaning. Tanlash

yo'nalish o'qi parametri bilan belgilanadi.

Ustunlar bo'yicha yig'ish:

>>> (df2> 20 ). har qanday ()

0 **yolg'on**

1 **To'g'ri**

2 **To'g'ri**

dip: bool

Qator bo'yicha aylantirish:

>>> (df2> 20 ). har qanday ( o'q = 1 )

0 **To'g'ri**

1 **To'g'ri**

dip: bool

Qatorlarda yoki ustunlarda mantiqiy VA operatsiyani bajarish uchun

all () usuli ishlatiladi. Yo'nalishni tanlash ham aniqlanadi

parametr o'qi.

Ustunlar bo'yicha yig'ish:

>>> (df2> 20 ). barchasi ()

0 **yolg'on**

1 **To'g'ri**

2 **yolg'on**

dip: bool

87

|  |
| --- |
| **89-bet** |

Qator bo'yicha aylantirish:

>>> (df2> 20 ). barchasi ( o'q = 1 )

0 **yolg'on**

1 **yolg'on**

dip: bool

Tenglik uchun tuzilish elementlarini taqqoslash uchun siz foydalanishingiz mumkin

tildan teng () usuli yoki tenglik operatori (==)

*Python* . Boshqa *DataFrame* yarataylik :

>>> json\_buf3 = **'[[' 12 ',' 17 ',' 18 '], [' 16 ',' 54 ',' 68 ']]'**

>>> df3 = pd.read\_json (json\_buf3, orient = **'qiymatlar'** )

>>> df3

0 1 2

0 12 17 18

1 16 54 68

Df2 strukturasi qanday ko'rinishini eslaylik:

>>> df2

0 1 2

0 12 24 14

1 16 54 25

Agar biz ushbu tuzilmalarni == operatori yordamida taqqoslasak, natijada

biz *DataFrame-ni* olamiz , uning elementlari mantiqiy o'zgaruvchilar bo'ladi.

Agar har ikkala berilgan holatda da elementlar qiymatlari *DataFrame "* s

mos keladigan bo'lsa, u holda o'zgaruvchi True bo'ladi, aks holda False *:*

>>> df2 == df3

0 1 2

0 **Haqiqiy yolg'on**

1 **Haqiqiy yolg'on**

88

|  |
| --- |
| **90-bet** |

Siz strukturaning har bir elementini bir oz doimiy bilan taqqoslashingiz mumkin

qiymati:

>>> df2 == 12

0 1 2

0 **Haqiqiy yolg'on**

1 **yolg'on yolg'on**

Ikki strukturaning tengligini tezda tekshirish uchun usuldan foydalaning

teng (). Agar tuzilmalar teng bo'lsa (xuddi shu pozitsiyalarda bitta mavjud

va bir xil qiymatlar), keyin natija rost bo'ladi, aks holda noto'g'ri:

>>> df2. teng (df3)

**Yolg'on**

>>> df2. teng (df2)

**To'g'ri**

**7.3 Statistika**

*Pandalar* turli xil statistik ma'lumotlarni hisoblash usullarini taqdim etadi (ro'yxat

7.1-jadvalda keltirilgan). Agar siz *DataFrame* tuzilmalari bilan ishlayotgan bo'lsangiz ,

unda siz hisoblash amalga oshiriladigan o'qni belgilashingiz mumkin: o'q = 0

ustunlar uchun eksa = 1 qatorlar uchun, o'q = 0 sukut bo'yicha.

**7.1-jadval - Statistikani hisoblash usullari**

**Usul**

**Tavsif**

hisoblash

*NA bo'lmagan* ob'ektlar soni

sum

Miqdor

anglatadi

O'rtacha qiymati

Telba

O'rtacha mutlaq og'ish

o'rtacha

Median

min

Eng kam

maksimal

Maksimal

rejimi

Moda

abs

Mutlaq qiymat

mahsulot

Tarkibi

89

|  |
| --- |
| **91-bet** |

std

Standart og'ish

var

Xolis farq

sem

O'rtacha standart xato

qiyshiq

Noqulaylik (uchinchi tartib momenti)

kurt

Ortiqcha (4-tartibning momenti)

miqdoriy

Miqdor (%)

jum

Kümülatif sum

zambil

Kümülatif mahsulot

cummax

Kümülatif maksimal

cummin

Kümülatif minimal

Endi qanday qilib foydalanishingiz mumkinligi haqida bir nechta misollarni ko'rib chiqamiz

ushbu usullar. Uchdan to'rtga *DataFrame* yarataylik :

>>> json\_buf = **'[[' 12 ',' 24 ',' 14 ',' 17 '], [' 16 ',' 54 ',' 25 ',' 83 '], [' 65 ',**

**'35', '12', '72']] '**

>>> df = pd.read\_json (json\_buf, orient = **'qiymat'** )

>>> df

0 1 2 3

0 12 24 14 17

1 16 54 25 83

2 65 35 12 72

Quyida statistikani hisoblash funktsiyalari qanday ishlashiga misollar keltirilgan.

Ustun summasi:

>>> df.sum ()

0 93

1 113

2 51

3,172

dip turi: int64

90

|  |
| --- |
| **92-bet** |

Qatorlar bo'yicha yig'indisi:

>>> df.sum ( o'q = 1 )

0 67

1 178

2 184

dip turi: int64

Ustun degani:

>>> df.mean ()

0 31.000000

1 37.666667

2 17.000000

3 57.333333

dip: float64

O'rtacha ustunlar bo'yicha:

>>> df.median ()

0 16.0

1 35.0

2 14.0

3 72.0

dip: float64

Ustunlar bo'yicha yig'indisi:

>>> df.cumsum ()

0 1 2 3

0 12 24 14 17

1 28 78 39 100

2 93 113 51 172

91

|  |
| --- |
| **93-bet** |

Statistik ma'lumotlarning qisqacha mazmuni uchun siz buni qilishingiz mumkin

usulidan foydalaning () *:*

>>> df.describe ()

0

12

3

hisoblash 3.000000 3.000000 3.0 3.000000

o'rtacha 31.000000 37.666667 17.0 57.333333

std 29.512709 15.176737 7.0 35.360053

min 12.000000 24.000000 12.0 17.000000

25 % 14.000000 29.500000 13.0 44.500000

50 % 16.000000 35.000000 14.0 72.000000

75 % 40.500000 44.500000 19.5 77.500000

max 65.000000 54.000000 25.0 83.000000

*DataFrames* uchun sukut bo'yicha statistika hisoblab chiqiladi

ustunlar.

Keling, statistikani funktsiyasi bilan hisoblash uchun *pandalar* statistikasi haqida umumiy ma'lumotni yakunlaymiz

value\_count (). Birinchidan, o'ttizta qiymatlar ro'yxatini tuzaylik,

ularning har biri 0 dan 7 gacha bo'lgan tasodifiy son:

>>> tasodifiy **import**

>>> tasodifiy urug '(123)

>>> rnd\_list = [random.randint ( 0 , 7 ) **uchun** I **-yilda** qator ( 30 )]

Da aniq raqamlar soni to'g'risida ma'lumot olish uchun

natijalar ro'yxati value\_count () funktsiyasidan foydalanadi *:*

>>> s = pd.Series (rnd\_list)

>>> s.value\_count ()

0 9

2 7

4 4

3 4

o'n to'rt

5 2

dip turi: int64

Ko'rib turganingizdek, bizning massivimizdagi hamma narsa nolga teng.

92

|  |
| --- |
| **Sahifa 94** |

**7.4 Funktsional kengayish**

**7.4.1 Ma'lumotlarni uzatish**

Ma'lumotlarni oqimlash - bu sizga qulay imkoniyat yaratadigan yondashuv

*pandalar* tuzilishi shaklidagi ma'lumotlarga ishlov berishni belgilaydigan shakl

natijasi berilgan ba'zi bir funktsiyalarning argumenti

keyingi funktsiyaga va boshqalar. Bu truboprovodga o'xshaydi. Uchun

ushbu yondashuvdan foydalanib, quvur () usuli qo'llaniladi. Ko'rib chiqing

bu misol. Birinchidan, bunday ishlov berishni holda amalga oshiraylik

mo'ri () funktsiyasidan foydalanish.

*Keling* barpo etish, bir *DataFrame* bir dan *JSON* kanop:

>>> json\_buf = **'[[' 12 ',' 24 ',' 14 ',' 17 '], [' 16 ',' 54 ',' 25 ',' 83 '], [' 65 ',**

**'35', '12', '72']] '**

>>> df = pd.read\_json (json\_buf, orient = **'qiymat'** )

Keling, uchta funktsiyani yarataylik: kvadrat, uchinchi ildiz va

10 raqamini chiqaradigan funktsiya:

>>> sqr = **lambda** x: x \*\* 2

>>> root3 = **lambda** x: x \*\* ( 1.0 / 3.0 )

>>> minus10 = **lambda** x: x - 10

93

|  |
| --- |
| **Sahifa 95** |

Agar biz avval df strukturasining har bir elementini o'rnatmoqchi bo'lsak

kvadrat, keyin olingan qiymatdan 10ni chiqarib oling va keyin oling

uchinchi darajaning ildizi, keyin buni amalga oshirishning eng oddiy usuli

vazifalar quyidagicha ko'rinadi:

>>> root3 (minus10 (sqr (df)))

0

1

2018-04-02 121 2

3

0 5.117230 8.271904 5.708267 6.534335

1 6.265827 14.270259 8.504035 19.018449

2 16.153471 10.670680 5.117230 17.295859

Bunday yozuv bilan bog'liq bir nechta muammolar mavjud: birinchidan: bu unchalik emas

ingl (garchi hamma ham bunga qo'shilmasa ham), ikkinchidan: bu funktsiyalar

o'qishni qiyinlashtiradigan qo'shimcha dalillar bo'lishi mumkin va

kod satrining ma'nosini tushunish; uchinchidan: bunday yozuvni o'zgartirish,

ishlatiladigan ko'plab funktsiyalar bilan, bu qiyinchiliklarni keltirib chiqaradi.

Siz () funktsiyasi, argument yordamida ushbu qiyinchiliklardan o'tishingiz mumkin

bu ishlov berish funktsiyasi:

>>> (df.pipe (sqr))

.pipe (minus10)

.pipe (root3))

0

1

2018-04-02 121 2

3

0 5.117230 8.271904 5.708267 6.534335

1 6.265827 14.270259 8.504035 19.018449

2 16.153471 10.670680 5.117230 17.295859

**7.4.2 Funktsiyani satr yoki ustun elementlariga qo'llash**

Statistikani hisoblash bo'limida biz foydalanganmiz

ni hisoblaydigan maxsus funktsiyalar (o'rtacha (), std () va boshqalar)

qator yoki ustun elementlari bo'yicha raqamli ko'rsatkichlar. *Pandalar*

94

|  |
| --- |
| **Sahifa 96** |

o'zingizning funktsiyalaringizdan foydalanish qobiliyatini ta'minlaydi

shunga o'xshash yondashuvni qo'llash. Buning uchun apply () usuli qo'llaniladi.

Tajribalar df tuzilishi bilan amalga oshiriladi (oqimga qarang

ma'lumotlarni qayta ishlash):

>>> df.apply ( **lambda** x: sum (x) / len (x))

0 31.000000

1 37.666667

2 17.000000

3 57.333333

dip: float64

Yuqoridagi misolda funktsiya argumentlari

lambda x: sum (x) / len (x) - ustunlar ro'yxati

asl tuzilish.

Bu erda yig'indining kvadrat ildizini hisoblaydigan funktsiyalarning bir varianti mavjud

ustunlar bo'yicha:

>>> df.apply ( **lambda** x: sum (x) \*\* ( 0.5 ))

0 9.643651

1 10.630146

2 7.141428

3 13.114877

dip: float64

Xuddi shu operatsiyani satrma-bosqich bajarish mumkin:

>>> df.apply ( **lambda** x: sum (x) \*\* ( 0.5 ), o'q = 1 )

0 8.185353

1 13.341664

2 13.564660

dip: float64

95

|  |
| --- |
| **Sahifa 97** |

**7.4.3 Birlashtirish ( *API* )**

Taqdim etadigan yana bir funktsional kengaytma

*pandalar* kutubxonasi birlashma. Pastki chiziq siz qila olasiz

biz qilganimizdek bir nechta funktsiyalardan foydalaning

oldingi qismda, (() funktsiyasini tahlil qilganimizda, faqat shu erda

Bunday holda, agg () funktsiyasi bizga yordam beradi.

Biz allaqachon ma'lum bo'lgan df tuzilmasi bilan ishlaymiz (oqimga qarang

ma'lumotlarni qayta ishlash):

>>> df

0 1 2 3

0 12 24 14 17

1 16 54 25 83

2 65 35 12 72

Birinchidan, elementlarning yig'indisini ustunlar bo'yicha topamiz, siz allaqachon bilishingiz kerak

buni qanday qilishning kamida 2-3 usuli, bu erda cho'chqachilik bankida yana bir qancha narsalar:

>>> df.agg ( **'sum'** )

0 93

1 113

2 51

3,172

dip turi: int64

96

|  |
| --- |
| **Sahifa 98** |

Ammo biz darhol summani, o'rtacha qiymatni va hisoblashni xohlasak nima bo'ladi

standart og'ishmi? Birlashtirishdan foydalanish juda oddiy

bajaring:

>>> df.agg ([ **'sum'** , **'mean'** , **'std'** ])

0

12

3

sum 93.000000 113.000000 51.0 172.000000

o'rtacha 31.000000 37.666667 17.0 57.333333

std 29.512709 15.176737 7.0 35.360053

O'z vazifalarimiz haqida unutmaylik:

>>> g'alati = **lambda** x: sum (x) \*\* ( 0,5 )

>>> min\_div\_5 = **lambda** x: min (x) / 5.0

>>> df.agg ([ max , g'alati, min\_div\_5])

0

1

2018-04-02 121 2

3

max 65.000000 54.000000 25.000000 83.000000

< **lambda** > 9.643651 10.630146 7.141428 13.114877

< **lambda** > 2.400000 4.800000 2.400000 3.400000

Ushbu misolda ta'kidlaganingizdek, funktsiya o'rniga g'alati va

min\_div\_5 <lambda> yozuvi mavjud;

oddiy *python* funktsiyalari sifatida ishlaydi:

>>> **def** g'alati (x):

**qaytish** summasi (x) \*\* ( 0,5 )

>>> **def** min\_div\_5 (x):

min (x) / 5.0 **qaytish**

>>> df.agg ([ max , g'alati, min\_div\_5])

0

1

2018-04-02 121 2

3

maksimal

65.000000 54.000000 25.000000 83.000000

g'alati 9.643651 10.630146 7.141428 13.114877

min\_div\_5 2.400000 4.800000 2.400000 3.400000

97

|  |
| --- |
| **99-bet** |

Endi hammasi joyida! Agar oldingi variantga qaytsangiz

*lambda* funktsiyalari, biz tayinlash orqali kerakli natijaga erishishimiz mumkin

mos keladigan nomlarni ishlaydi (ism bo'lishi mumkin

har qanday satr):

>>> g'alati = **lambda** x: sum (x) \*\* ( 0,5 )

>>> min\_div\_5 = **lambda** x: min (x) / 5.0

>>> g'alati. \_\_name\_\_ = **'g'alati kulgili'**

>>> min\_div\_5. \_\_name\_\_ = **'min / 5'**

>>> df.agg ([ max , g'alati, min\_div\_5])

0

1

2018-04-02 121 2

3

maksimal

65.000000 54.000000 25.000000 83.000000

g'alati qiziqarli 9.643651 10.630146 7.141428 13.114877

min / 5 2.400000 4.800000 2.400000 3.400000

**7.4.4 Ma'lumotlarni o'zgartirish**

Transformatsiya funktsiyasini chaqirish formati shunga o'xshash

birlashtirishda ishlatiladi. Ikkinchisidan farqli o'laroq, transformatsiya -

bu strukturaning har bir elementiga funktsiyaning qo'llanilishi, natijada

bir xil o'lchamdagi (yoki kattaroq) struktura qaytariladi, lekin bilan

o'zgargan elementlar.

Keling, ikkita funktsiyani yarataylik:

>>> mul2 = **lambda** x: x \* 2

>>> mul2. \_\_name\_\_ = **'mul2'**

>>> div2 = **lambda** x: x / 2

>>> div2. \_\_name\_\_ = **'div2'**

98

|  |
| --- |
| **100-bet** |

Elementlarni o'zgartirish uchun transformatsiya funktsiyasidan foydalanish

df tuzilmalari (ma'lumot oqimini ko'ring):

>>> df.transform ([mul2])

0 1 2 3

mul2 mul2 mul2 mul2

0 24 48 28 34

1 32 108 50 166

2 130 70 24 144

O'zgartirish uchun ikkita funktsiyali variant:

>>> df.transform ([mul2, div2])

0

1

2018-04-02 121 2

3

mul2 div2 mul2 div2 mul2 div2 mul2 div2

0 24 6.0 48 12.0 28 7.0 34 8.5

1 32 8.0 108 27.0 50 12.5 166 41.5

2 130 32,5 70 17,5 24 6,0 144 36,0

Siz alohida ustunlar bilan ishlashingiz mumkin:

>>> df [ 0 ] .transform ([mul2, div2])

mul2 div2

0 24 6.0

1 32 8.0

2 130 32.5

**7.5 Ishlash uchun *str* tipidagi usullardan foydalanish**

**matnli ma'lumotlar**

Ko'pincha *pandalar* ma'lumotlar tuzilmalari matnni saqlash uchun ishlatiladi

ma'lumotlar - satrlar va belgilar. Matnli ma'lumotlar ham uning bir qismi bo'lishi mumkin

ustunlarning sarlavhalari va boshqalar kabi strukturaning o'zi. *Pandalar* ruxsat beradi

ning ko'chasi turdagi o'yinga foydalanish *Python tili* kabi bilan ish qilish

ma'lumotlar.

99

|  |
| --- |
| **101-sahifa** |

Amalda bir nechta misollarni ko'rib chiqamiz. Birinchidan, *ketma-ket* yarataylik ,

matnli ma'lumotlarni o'z ichiga olgan:

>>> s = pd.Series ([ **'hellO'** , **'abcABC'** , **'one'** , **'TWO'** , **'tHRee'** ])

>>> lar

0 jahannam

1 abcABC

2018-04-02 121 2

bitta

3

Ikki

4 tHRee

turi: ob'ekt

Str atributi orqali biz barcha usullarga kirishimiz mumkin

Struktura a'zolariga qo'llaniladigan *Python-* dan str ma'lumotlar

*Seriya* .

Keling, struktura elementlarining barcha harflarini kichik harfga o'tkazamiz:

>>> s.str.lower ()

0 salom

1 abcabc

2018-04-02 121 2

bitta

3

ikkitasi

4 uchta

turi: ob'ekt

yuz

|  |
| --- |
| **Sahifa 102** |

Bo'sh joylarni olib tashlang va so'zlarning birinchi harflarini bosh harf bilan yozing:

>>> s.str.lower (). str.strip (). str.title ()

0 salom

1 Abcabc

2 bitta

3 Ikki

4 Uch

turi: ob'ekt

Str tipidagi ob'ekt uchun mavjud bo'lgan usullarning to'liq ro'yxatini topish mumkin

*Python* hujjatlari .

101

|  |
| --- |
| **103-sahifa** |

**8-bob. *Pandalarni*sozlash**

**8.1 *API pandalar* sozlamalari bilan ishlash uchun**

*Pandas* kutubxonasi unga sozlamalar bilan ishlash uchun *API* beradi

8.1-jadvalda keltirilgan funktsiyalarni o'z ichiga oladi.

**8.1-jadval - *pandalar* sozlamalari bilan ishlash funktsiyalari**

**Funktsiya**

**Tavsif**

get\_option ()

Parametr qiymatini olish

set\_option ()

Yangi parametr qiymatini o'rnatish

reset\_option ()

Parametrni asl qiymatiga qaytarish

description\_option ()

Parametrning matnli tavsifini ko'rsatish

option\_context ()

Parametrlarga yangi qiymatlarni berish

kodning ma'lum bir blokida. Tomonidan ishlatilgan

bilan bayonot bilan

Atributlarda bo'lgani kabi sozlamalar bilan ham ishlash mumkin (nuqta orqali).

Keling, birinchi variantni ko'rib chiqaylik: funktsiyani chaqirish orqali sozlamalarni o'zgartirish.

Parametr qiymatini olish uchun funktsiyadan foydalaning

get\_option (), argument sifatida u odatiy tarzda qabul qilinadi

matn shaklidagi parametr.

Ko'rsatiladigan qatorlarning maksimal soni:

>>> pd.get\_option ( **'display.max\_rows'** )

60

Ishlatilgan kodlash:

>>> pd.get\_option ( **'display.encoding'** )

**'cp1251'**

102

|  |
| --- |
| **104-bet** |

Ko'rsatilgan ma'lumotlar ustunlarining maksimal soni:

>>> pd.get\_option ( **'display.max\_columns'** )

0

Yangi qiymatni o'rnatish funktsiya yordamida amalga oshiriladi

set\_option (): uning birinchi argumenti - parametr nomi

matn shakli, ikkinchisi - parametrning yangi qiymati.

Namoyish uchun *ketma-ket* tuzilmani yarataylik :

>>> s = pd seriyalari ([ 10 , 3 , 46 , 1 , 312 , 344 , 193 , 42 , 39 , 77 , 3 ])

>>> lar

0 10

13

2 46

3 1

4312

5344

6193

7 42

8 39

9 77

10 3

dip turi: int64

Display.max\_rows parametrining joriy qiymati (maksimal

ko'rsatiladigan qatorlar soni):

>>> pd.get\_option ( **'display.max\_rows'** )

60

Keling, ushbu qiymatni 5 ga o'zgartiring:

>>> pd.set\_option ( **'display.max\_rows'** , 5 )

>>> pd.get\_option ( **'display.max\_rows'** )

besh

103

|  |
| --- |
| **Sahifa 105** |

S tuzilmasi tarkibini yangi sozlamalar bilan chiqarish:

>>> lar

0 10

13

..

9 77

10 3

Uzunlik: 11 , dip turi: int64

Oldingi qiymatni o'rnatamiz:

>>> pd.set\_option ( **'display.max\_rows'** , 60 )

>>> pd.get\_option ( **'display.max\_rows'** )

60

>>> lar

0 10

13

2 46

3 1

4312

5344

6193

7 42

8 39

9 77

10 3

dip turi: int64

Parametr qiymati uchun standart qiymatga qaytarilishi mumkin

bu reset\_option () funktsiyasidan foydalanadi.

104

|  |
| --- |
| **Sahifa 106** |

Boshlang'ich display.max\_rows qiymati:

>>> pd.get\_option ( **'display.max\_rows'** )

60

Display.max\_rows = 10 ga yangi qiymat tayinlaymiz:

>>> pd.set\_option ( **'display.max\_rows'** , 10 )

>>> pd.get\_option ( **'display.max\_rows'** )

o'n

Display.max\_rows-ni standart qiymatiga qaytaramiz:

>>> pd.reset\_option ( **'display.max\_rows'** )

>>> pd.get\_option ( **'display.max\_rows'** )

60

Parametrning tavsifini funktsiya yordamida olish mumkin

description\_option ():

>>> pd.describe\_option ( **'display.max\_rows'** )

display.max\_rows: int

Max\_rows bo'lsa **bo'ladi** oshdi, TRUNCATE ko'rinishga o'tish. Bog'liq holda

`large\_repr`, ob'ektlar ham markaziy olishadi **yoki** bosilgan **sifatida**

qisqacha ko'rinish. **"Yo'q"** qiymati cheksiz degan ma'noni anglatadi.

Ishi Python / IPython **bo'ladi** yugurish **ham** bir terminal **va** `large\_repr`

**"kesilgan"** ga teng bo'lsa **,** bu 0 ga o'rnatilishi mumkin **va** pandalar avtomatik ravishda aniqlanadi

terminalning balandligi **va** mos keladigan kesilgan ob'ektni chop eting

ekran balandligi. IPython daftarchasi, IPython qtconsole **yoki**

IDLE yo'q **emas** ishlatish **ham** bir terminal **va** shuning u **emas** , albatta mumkin

to'g'ri avtomatik aniqlash.

[standart: 60] [hozirda: 60]

105

|  |
| --- |
| **107-bet** |

Agar sizga kerak bo'lgan kod blokini bajarish kerak bo'lsa

bir qator belgilangan parametrlarni vaqtincha belgilash

qiymatlari, keyin bu holda funktsiyadan foydalanish tavsiya etiladi

option\_context ():

>>> **bilan** pd.option\_context ( **«display.max\_rows»** , 25 ):

chop etish (pd.get\_option ( **'display.max\_rows'** ))

25

>>> pd.get\_option ( **'display.max\_rows'** )

60

Sozlamalar bilan ishlashning ikkinchi variant - bu xususiyatdan foydalanish

imkoniyatlari. Bizga tanish bo'lgan sozlamalarning qiymatlarini namoyish etamiz:

>>> pd.options.display.max\_rows

60

>>> pd.options.display.max\_columns

0

Parametrga yangi qiymat beraylik:

>>> pd.options.display.max\_columns = 3

>>> pd.options.display.max\_columns

3

>>> pd.options.display.max\_columns = 20

>>> pd.options.display.max\_columns

20

**8.2 *pandalar* kutubxonasi sozlamalari**

Quyida, 8.2-jadvalda, sozlanishi parametrlar

*pandalar* bilan ishlashda mavjud .

106

|  |
| --- |
| **Sahifa 108** |

**Jadval 8.2 - *pandalar* sozlanishi variantlar**

**Xususiylashtirish**

**Qiymat bo'yicha**

**sukut bo'yicha**

**Tavsif**

display.chop\_threshold Hech narsa

Agar *float* bu ostida bo'lsa

eshik, keyin bo'ladi

0 sifatida ko'rsatish

display.colheader\_

oqlash

to'g'ri

Sarlavhalarni tekislang

ustunlar. Tomonidan ishlatilgan

*DataFrameFormatter.*

display.date\_dayfirst

Yolg'on

Agar parametr *True bo'lsa* , unda uchun

sanalarni ko'rsatish va tahlil qilish

quyidagilardan foydalaniladi

buyurtma: kun / oy / yil.

display.date\_yearfirst False

Agar parametr *True bo'lsa* , unda uchun

sanalarni ko'rsatish va tahlil qilish

quyidagilardan foydalaniladi

buyurtma: yil / oy / kun.

displey.kodlash

UTF-8

Satrlar uchun kodlashni belgilaydi

usuli orqali qaytib keldi

*to\_string* , shuningdek uchun

konsolda satrlarni ko'rsatish.

display.max\_columns

20

Agar *Python / IPython* ishlayotgan bo'lsa

terminal, keyin bu parametr

0 ga, ga o'rnatilishi mumkin

bu holda *pandalar* bo'ladi

avtomatik ravishda aniqlash

terminal kengligi va

boshqa formatdan foydalaning

agar barcha ustunlar bo'lsa

vertikal ravishda mos kelmadi.

*Yo'q* qiymati belgilaydi

cheksiz miqdor

belgilar.

display.max\_colwidth

50

Maksimal ustun kengligi

(belgilar bilan) ifodalash

*pandalar* ma'lumotlar tuzilmalari . Qachon

toshib ketish sodir bo'ladi

belgi bo'yicha ustun, keyin chiqishga

"..." belgilari qo'shiladi.

display.max\_rows

60

Parametrlar to'plami

maksimal chiziqlar soni,

da chiqarilgan bo'ladi

*pandalar* tuzilmalarini namoyish qilish .

107

|  |
| --- |
| **Sahifa 109** |

Cheklovni olib tashlash uchun

qatorlar soni, belgilang

bu parametr *Yo'q* .

display.memory\_usage

To'g'ri

Yo'q yoki yo'qligini bildiradi

band bo'lganlar miqdori

*DataFrame* uchun xotira

*df.info ()* usulini chaqirish .

display.notebook\_repr\_

HTML

To'g'ri

Agar parametr *True bo'lsa* , u holda

*IPython notebooki* bo'ladi

*html* -view-dan foydalaning

*pandalar* moslamalari uchun .

displey. aniqlik

6

Belgilar sonini belgilaydi

raqamlarni ko'rsatishda o'nli kasrdan keyin

suzuvchi nuqta.

displey kengligi

80

Belgilar bilan kenglikni ko'rsatish.

Agar *python / IPython* ishlayotgan bo'lsa

terminal, parametr mumkin

o'rnatilgan bo'lishi *hech* bu,

ishi *pandas* avtomatik ravishda

kengligini aniqlaydi.

display.html.border

1

Uchun qiymatni belgilaydi

atribut *chegarasi = qiymat* , qaysi

joylashtirilgan bo'ladi *<table> tegi* uchun

*HTML* tuzilish ko'rinishlari

*DataFrame* .

io.excel.xls.writer

xlwt

Odatiy

*Excel* bilan ishlash uchun vosita -

" *xls* " formatidagi fayllar .

io.excel.xlsm.writer

openpyxl

Odatiy

*Excel* bilan ishlash uchun vosita -

" *xlsm* " formatidagi fayllar .

io.excel.xlsx.writer

openpyxl

Odatiy

*Excel* bilan ishlash uchun vosita -

" *xlsx* " formatidagi fayllar .

mode.sim\_interactive

Yolg'on

Modellashtirish interaktiv

rejimi, qachon ishlatiladi

sinov.

108

|  |
| --- |
| **Sahifa 110** |

**9-bob. Ma'lumotlar bilan ishlash vositalari**

**9.1 Sürgülü oyna. Statistlar**

Biz ko'rib chiqadigan birinchi vositalar to'plami

surma statistikasi (yoki oyna) deb atash mumkin bo'lgan guruh

funktsiyalar). Ularning mohiyati shundan iboratki, turli xil statistik ma'lumotlar

o'rtacha, o'rtacha, kovaryans, standart sifatida

og'ish va boshqalar. strukturaning barcha ob'ektlari uchun emas, balki hisoblab chiqilgan

faqat ketma-ket qiymatlar guruhi uchun, ushbu guruhning kattaligi

qo'lda o'rnatilishi uchun oldindan o'rnatilgan (bu parametr

nomi - oyna hajmi).

**9.1-rasm - toymasin oyna**

109

o'n

3

besh

1

to'qqiz

23

...

45

3

27

o'n bir

2018-04-02 121 2

O'lcham oynasi

3

Yo'nalish

oyna harakatlari

|  |
| --- |
| **Sahifa 111** |

Statistikani hisoblash (masalan, o'rtacha) quyidagicha bo'ladi:

**9.2-rasm - Statistikani hisoblash (o'rtacha)**

Tajribalarni o'tkazish uchun *ketma-ket* tuzilmani yarating :

>>> pandalarni pd **sifatida import qilish**

>>> tasodifiy **import**

>>> arr = [random.randint ( 0 , 50 ) **uchun** I **-yilda** qator ( 500 )]

>>> s = pd.Series (arr)

>>> s.shape

( 500 ,)

>>> lar [ 0 : 5 ]

0 28

1 48

2 8

3 4

4 21

dip turi: int64

110

o'n

3

besh

o'rtacha = (10 + 3 + 5) / 3 =

6

3

besh

1

o'rtacha = (3 + 5 + 1) / 3 = 3

besh

1

to'qqiz

o'rtacha = (5 + 1 + 9) / 3 = 5

...

|  |
| --- |
| **112-bet** |

Birinchidan, berilgan oyna bilan har xil statistik ma'lumotlarni hisoblash uchun,

Rolling sinfining ob'ektini yarataylik. Buning uchun biz funktsiyadan foydalanamiz

pandas.DataFrame.rolling ():

DataFrame.rolling (oyna, min\_periods = **Yo'q** , freq = **Yo'q** , markaz = **False** ,

win\_type = **Hech** = haqida **hech** , o'qi = 0 , yopiq = **Bo'sh** )

Ushbu funktsiya quyidagi parametrlarga ega:

• oyna *:* int

◦ Oynaning kattaligi.

• min\_periods *:* int; sukut bo'yicha: yo'q

◦ Qabul qilish uchun oynadagi elementlarning minimal soni

statistik qiymatlar.

• freq: str yoki DateOffset ob'ekti; standart qiymat: Yo'q

◦ Hozirgi vaqtda ushbu parametr maqomga ega

Eskirgan va foydalanish uchun tavsiya etilmaydi.

• center: bool; standart qiymat: noto'g'ri

◦ Oynaning o'rtasiga yorliq qo'yadi.

• win\_type: str; standart qiymat: Yo'q

◦ Oyna turi (ushbu parametr batafsilroq ko'rib chiqiladi

keyinroq).

• str: str; standart qiymat: Yo'q

◦ Hisoblanadigan ustunni belgilaydi.

• yopiq: str; standart qiymat: Yo'q

The yopiq intervalning so'nggi nuqtalarini belgilaydi ('o'ng',

"chap", "ikkalasi", "na").

• o'qi: int yoki str; standart qiymat: 0

The Hisob-kitoblar bajariladigan o'q (0 -

qatorlar bo'yicha hisob-kitoblar, 1 - ustunlar bo'yicha).

111

|  |
| --- |
| **113-bet** |

Standart sozlamalari va o'rnatilishi bilan Rolling ob'ekti yarataylik

oyna hajmi 10 ga teng:

>>> roll = s.rolling ( oyna = 10 )

>>> rulon

Rolling [oyna = 10 , markaz = **False** , o'q = 0 ]

Ushbu ob'ekt harakatlanishni hisoblash usullarini taqdim etadi

statistik. Usullar ro'yxati deyarli nima bilan bir xil

7-bobda keltirilgan ("statistika" bo'limi). Quyida, 9.1-jadvalda,

eng ko'p ishlatiladigan funktsiyalar keltirilgan.

**9.1-jadval - harakatlanuvchi statistikani hisoblashda ishlatiladigan usullar**

**Usul**

**Tavsif**

murojaat qilish ()

Umumiy maqsad vazifasi

corr ()

O'zaro bog'liqlik

cov ()

Kovaryans

hisoblash ()

*NA bo'lmagan* ob'ektlar soni

kurt ()

Ortiqcha (4-tartibning momenti)

maksimal ()

Maksimal

anglatadi ()

O'rtacha qiymati

o'rtacha ()

Median

min ()

Eng kam

miqdoriy ()

Miqdor (%)

qiyshiq ()

Noqulaylik (uchinchi tartib momenti)

std ()

Standart og'ish

sum ()

Miqdor

var ()

Xolis farq

112

|  |
| --- |
| **Sahifa 114** |

Endi siz nisbatan statistikani hisoblash funktsiyalaridan foydalanishingiz mumkin

berilgan oyna bilan yaratilgan ma'lumotlar bazasiga:

>>> pd.options.display.max\_rows = 20

>>> roll.median ()

0 NaN

1 NaN

2 NaN

3 NaN

4 NaN

5 NaN

6 NaN

7 NaN

8 NaN

9 24.5

...

490 29.5

491 24.0

492 18.5

493 21.0

494 23.0

495 23.0

496 23.0

497 23.0

498 23.0

499 23.0

Uzunlik: 500 , dip: float64

Ko'rsatilgan ma'lumotni cheklash uchun biz foydalanganmiz

max\_rows sozlamalari, bu nima va undan qanday foydalanish kerak

parametr 8-bobda yozilgan. Ko'rib turganingizdek: o'ndan oldin

element (bizning holatlarimizda, bu hisoblashdan buyon 9 indeksli element

noldan boshlanadi) tuzilish a'zolari NaN. U ulangan

113

|  |
| --- |
| **Sahifa 115** |

oyna hajmi 10 ga teng, statistika esa o'nga teng

oldingi elementlar, bunday lahzaga etib borganimizda keladi

indeks 9.

10 o'lchamdagi oyna uchun standart og'ishni hisoblash:

>>> roll.std ()

0

NaN

1

NaN

2018-04-02 121 2

NaN

3

NaN

4

NaN

besh

NaN

6

NaN

7

NaN

8

NaN

9 16.445533

...

490 15.307587

491 16.308144

492 17.049275

493 17.034605

494 15.486195

495 16.642983

496 16.578433

497 17.678927

498 17.142864

499 16.892141

Uzunlik: 500 , dip: float64

114

|  |
| --- |
| **Sahifa 116** |

Ularning orasida eng yuqori raqamli qiymatni aniqlash

oyna elementlari:

>>> roll.max ()

0 NaN

1 NaN

2 NaN

3 NaN

4 NaN

5 NaN

6 NaN

7 NaN

8 NaN

9 48.0

...

490 48.0

491 48.0

492 48.0

493 48.0

494 48.0

495 48.0

496 48.0

497 48.0

498 48.0

499 48.0

Uzunlik: 500 , dip: float64

Siz o'zingizning funktsiyalaringizdan statistika hisoblash uchun foydalanishingiz mumkin, buning uchun ular

avval yaratib, so'ngra quyidagicha o'tkazishingiz kerak

*apply ()* usuli uchun parametr .

115

|  |
| --- |
| **Sahifa 117** |

*Pandalar* sizga oyna turini o'rnatishga imkon beradi, bu parametr yordamida amalga oshiriladi

*ning win\_type prokatlash* . Quyida mumkin bo'lgan jadval mavjud

ushbu parametrning qiymatlari.

**9.2-jadval - Statistikani hisoblash uchun oyna turini belgilaydigan parametrlar**

**Parametr qiymati Tavsif**

vagon

To'rtburchak oyna (Dirichlet oynasi)

uchburchak

Uchburchak oyna

qora odam

Blackman oynasi

zarba berish

Xemming oynasi

Bartlett

Barletta oynasi

parzen

Parzenning oynasi

bohman

Bohman oynasi

karmanharris

Blekman-Xarris oynasi

nuttall

Nattelning oynasi

barthann

Barletta-Xanna oynasi

kaiser

Kaiser oynasi

gauss

Gauss oynasi

general\_gaussian

Umumlashtirilgan Gauss oynasi

shafqatsiz

Slepianning oynasi

Ushbu parametrlar bilan ishlash xususiyatlarining tavsifi tashqariga chiqadi

kitob ramkalari.

**9.2 Oynani kengaytirish. Statistlar**

Oldingi bo'limda biz toymasin derazalar va

ular bo'yicha statistikani hisoblash. Ushbu bo'lim kengayishga qaratilgan

o'lchamlari, toymasin oynalardan farqli o'laroq, o'zgarib turadigan,

birinchi elementdan kengaymoqda.

116

|  |
| --- |
| **Sahifa 118** |

Kengaygan oynaning printsipi quyidagi rasmlarda keltirilgan.

**9.3-rasm - kengaytirilayotgan oyna**

117

o'n

3

besh

1

to'qqiz

23

...

45

3

27

o'n bir

2018-04-02 121 2

Yo'nalish

kattalashtirish

oyna

|  |
| --- |
| **119-bet** |

Bunday oyna uchun o'rtacha qiymat hisoblab chiqiladi

quyida bayon qilinganidek:

**9.4-rasm - kengayib borayotgan oyna uchun o'rtacha qiymatni hisoblash**

Kengayayotgan oyna bilan ishlash uchun sinf ob'ekti ishlatiladi

Tuzilishni kengaytirish () usuli orqali olinadigan kengaytiruvchi

*pandalar* .

Oldingi bobda yaratilgan *s* strukturasidan foydalanamiz :

>>> lar [ 0 : 5 ]

0 28

1 48

2 8

3 4

4 21

dip turi: int64

118

o'n

3

besh

o'n

3

besh

1

o'n

3

besh

1

to'qqiz

o'rtacha = (10 + 3 + 5) / 3 = 6

o'rtacha = (10 + 3 + 5 + 1) / 4 = 4.75

o'rtacha = (10 + 3 + 5 + 1 + 6) / 5 = 5

|  |
| --- |
| **120-bet** |

Kengaytiruvchi ob'ekt yarataylik:

>>> ex = s.expanding ()

>>> sobiq

Kengaymoqda [min\_periods = 1 , center = **False** , axis = 0 ]

Statistika shunga o'xshash funktsiyalar yordamida hisoblanadi

Rolling moslamalari bilan ishlashda ishlatiladi (1-jadvalga qarang):

>>> pd.options.display.max\_rows = 20

Kengayayotgan oyna uchun o'rtacha qiymatni hisoblaymiz:

>>> ex.mean ()

0 28.000000

1 38.000000

2 28.000000

3 22.000000

4 21.800000

5 21.333333

6 24.285714

7 26.750000

8 24.777778

9 26.300000

...

490 26.348269

491 26.298780

492 26.255578

493 26.248988

494 26.242424

495 26.197581

496 26.207243

497 26.238956

498 26.220441

499 26.252000

Uzunlik: 500 , dip: float64

119

|  |
| --- |
| **121-bet** |

Kengayayotgan oyna uchun standart og'ishni hisoblash:

>>> ex.std ()

0

NaN

1 14.142136

2 20.000000

3 20.264912

4 17.555626

5 15.743782

6 16.357611

7 16.671190

8 16.679162

9 16.445533

...

490 14.573350

491 14.599827

492 14.616493

493 14.602396

494 14.588340

495 14.607777

496 14.594634

497 14.597109

498 14.588310

499 14.590760

Uzunlik: 500 , dip: float64

Kengayayotgan oyna bilan ishlashga yondashuv bir xil

toymasin oyna uchun ishlatiladi.

**9.3 Vaqtga asoslangan slip**

Vaqtga yo'naltirilgan siljishning mohiyati shundaki, bu oyna

vaqt oralig'idan foydalaniladi.

120

|  |
| --- |
| **Sahifa 122** |

*DataFrame-* ni indeks sifatida *vaqt tamg'asi bilan* yarataylik :

>>> pandalarni pd **sifatida import qilish**

>>> df = pd.DataFrame ([ 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 ],

index = pd.date\_range ( **'20180101 00:00:00'** , davrlar = 10 , freq = **'s'** ))

>>> df

0

2018 - 01 - 01 00 : 00 : 00 1

2018 - 01 - 01 00 : 00 : 01 2

2018 - 01 - 01 00 : 00 : 02 3

2018 - 01 - 01 00 : 00 : 03 4

2018 - 01 - 01 00 : 00 : 04 5

2018 - 01 - 01 00 : 00 : 05 6

2018 - 01 - 01 00 : 00 : 06 7

2018 - 01 - 01 00 : 00 : 07 8

2018 - 01 - 01 00 : 00 : 08 9

2018 - 01 - 01 00 : 00 : 09 10

2 soniya oynasi bilan Rolling ob'ektini yarating:

>>> rt = df.rolling ( **'2s'** )

>>> rt.sum ()

0

2018 - 01 - 01 00 : 00 : 00 1.0

2018 - 01 - 01 00 : 00 : 01 3.0

2018 - 01 - 01 00 : 00 : 02 5.0

2018 - 01 - 01 00 : 00 : 03 7.0

2018 - 01 - 01 00 : 00 : 04 9.0

2018 - 01 - 01 00 : 00 : 05 11.0

2018 - 01 - 01 00 : 00 : 06 13.0

2018 - 01 - 01 00 : 00 : 07 15.0

2018 - 01 - 01 00 : 00 : 08 17.0

2018 - 01 - 01 00 : 00 : 09 19.0

121 2

|  |
| --- |
| **123-bet** |

5 soniyali oyna natijasi:

>>> rt = df.rolling ( **'5s'** )

>>> rt.sum ()

0

2018 - 01 - 01 00 : 00 : 00 1.0

2018 - 01 - 01 00 : 00 : 01 3.0

2018 - 01 - 01 00 : 00 : 02 6.0

2018 - 01 - 01 00 : 00 : 03 10.0

2018 - 01 - 01 00 : 00 : 04 15.0

2018 - 01 - 01 00 : 00 : 05 20.0

2018 - 01 - 01 00 : 00 : 06 25.0

2018 - 01 - 01 00 : 00 : 07 30.0

2018 - 01 - 01 00 : 00 : 08 35.0

2018 - 01 - 01 00 : 00 : 09 40.0

**9.4 Ma'lumotlarni birlashtirish**

Birlashtirish funktsiyasi bir vaqtning o'zida hisoblash imkonini beradi

ma'lumotlar to'plami uchun turli xil statistik ma'lumotlar. Tajribalar uchun

oldingi qismdan df tuzilishini olaylik:

>>> df

0

2018 - 01 - 01 00 : 00 : 00 1

2018 - 01 - 01 00 : 00 : 01 2

2018 - 01 - 01 00 : 00 : 02 3

2018 - 01 - 01 00 : 00 : 03 4

2018 - 01 - 01 00 : 00 : 04 5

2018 - 01 - 01 00 : 00 : 05 6

2018 - 01 - 01 00 : 00 : 06 7

2018 - 01 - 01 00 : 00 : 07 8

2018 - 01 - 01 00 : 00 : 08 9

2018 - 01 - 01 00 : 00 : 09 10

122

|  |
| --- |
| **Sahifa 124** |

Buning uchun slayd ob'ekti yarataylik:

>>> rt = df.rolling ( oyna = **'5s'** )

>>> rt

Rolling

[window = 5000000000 , min\_periods = 1 , center = **False** , win\_type = freq, axis = 0 ]

Endi yig’indisini, o’rtachasini va hisoblaymiz

standart og'ish. Biz statistikani hisoblash funktsiyalarini quyidagidan olamiz

qaqshatqich kutubxonalar:

>>> numpy- **ni** np **sifatida import qilish**

>>> rt.agg ([np.sum, np.mean, np.std])

0

sum o'rtacha std

2018 - 01 - 01 00 : 00 : 00 1.0 1.0 NaN

2018 - 01 - 01 00 : 00 : 01 3.0 1.5 0.707107

2018 - 01 - 01 00 : 00 : 02 6.0 2.0 1.000000

2018 - 01 - 01 00 : 00 : 03 10.0 2.5 1.290994

2018 - 01 - 01 00 : 00 : 04 15.0 3.0 1.581139

2018 - 01 - 01 00 : 00 : 05 20.0 4.0 1.581139

2018 - 01 - 01 00 : 00 : 06 25.0 5.0 1.581139

2018 - 01 - 01 00 : 00 : 07 30.0 6.0 1.581139

2018 - 01 - 01 00 : 00 : 08 35.0 7.0 1.581139

2018 - 01 - 01 00 : 00 : 09 40.0 8.0 1.581139

Agar kerak bo'lsa, siz funktsiyalarni mustaqil ravishda amalga oshirishingiz mumkin

zarur statistikani hisoblash.

123

|  |
| --- |
| **Sahifa 125** |

**10-bob. Vaqt qatorlari**

Ma'lumotlar bilan ishlash sohasidagi mashhur yo'nalish

vaqt qatorlarini tahlil qilish. *Pandalar* kutubxonasi to'plamni o'z ichiga oladi

ma'lumotlar bilan ishlashga imkon beradigan vositalar,

vaqt seriyasi sifatida taqdim etilgan.

Vaqt qatorlari bilan ishlash uchun *pandalar* sinflardan foydalanadi,

10.1-jadvalda keltirilgan.

**Jadval 10.1 - vaqtincha ishlash uchun ishlatiladigan pandalar sinflari**

**qatorlarda**

**Sinf**

**Maxluq**

**Tavsif**

Vaqt tamg'asi

to\_datetime,

Vaqt tamg'asi

Yagona vaqtinchalik

yorliq

DatetimeIndex

to\_datetime,

sana\_ qatori,

bdate\_range,

DatetimeIndex

Ob'ektlar to'plami

Vaqt tamg'asi

Davr

Davr

Yagona vaqtinchalik

oraliq

PeriodIndex

davr oralig'i,

PeriodIndex

Period ob'ektlari to'plami

*Pandas* vaqt yorliqli va intervallarni ajrata.

Vaqt tamg'asi - bu ma'lum bir sana / vaqt qiymati, masalan:

2018-01-01 01:20:35. Vaqt oralig'i mavjudlikni nazarda tutadi

muddatni belgilaydigan to'liq bo'lmagan vaqt belgisi va marker,

masalan 2018-01 yorlig'i oyni bildiruvchi 'M' markeriga ega bo'ladi.

DatetimeIndex - bu vaqtinchalik qatorni saqlashga imkon beradigan sinf

qachon indeks sifatida ishlatilishi mumkin bo'lgan yorliqlar

124

|  |
| --- |
| **Sahifa 126** |

*pandalar* ma'lumotlar tuzilmalarini qurish . PeriodIndex - saqlaydigan sinf

ishlatilishi mumkin bo'lgan vaqt oralig'idagi qator

indeks sifatida.

**10.1 Vaqt belgilari bilan ishlash**

**10.1.1 Vaqt tamg'asini yarating**

Vaqt tamg'asini (Timestamp sinfining ob'ekti) yaratish uchun siz qilishingiz mumkin

konstruktordan foydalaning

Vaqt tamg'asi (), yoki usul bo'yicha

to\_datetime ().

Vaqt tamg'asi konstruktori bilan tanishishdan boshlaymiz:

pandas.Timestamp (ts\_input, freq, tz, birlik, ofset)

Quyida argumentlarning tavsifi berilgan.

• ts\_input: datetime *,* str *,* int *,* float

◦ sinf ob'ektiga o'tkaziladigan qiymat

Vaqt tamg'asi.

• freq: str *,* DateOffset

◦ Shift miqdori.

• tz: str *,* pytz.timezone *,* dateutil.tz.tzfile yoki None *;*qiymat

sukut bo'yicha: yo'q

◦ Vaqt zonasi.

• birlik: str; standart qiymat: Yo'q

◦ o'lchov birligi, agar qiymat bo'lsa parametr ishlatiladi

ts\_input int yoki float.

• ofset: str *,* DateOffset: standart qiymat: yo'q

Supported Qo'llab-quvvatlanmaydi, tezlikni ishlating.

125

|  |
| --- |
| **Sahifa 127** |

Vaqt tamg'asi ob'ektini yaratish variantlarini ko'rib chiqamiz.

Satrdan ob'ekt yaratish:

>>> ts = pd.Timestamp ( **'2018-10-5'** )

>>> ts

Vaqt tamg'asi ( **'2018-10-05 00:00:00'** )

>>> ts = pd.Timestamp ( **'2018-10-5 01:15:33'** )

>>> ts

Vaqt tamg'asi ( **'2018-10-05 01:15:33'** )

Datetime ob'ektidan foydalanish:

>>> **dan** DATETIME **import** DATETIME

>>> dt = datetime.now ()

>>> dt

datetime.datetime ( 2019 , 12 , 2 , 21 , 33 , 48 , 621911 )

>>> ts = pd.Timestamp (dt)

>>> ts

Vaqt tamg'asi ( **'2019-12-02 21: 33: 48.621911'** )

Int va float qiymatlaridan foydalanish:

>>> ts = pd.Timestamp ( 1517246359 , birlik = **'s'** )

>>> ts

Vaqt tamg'asi ( **'2018-01-29 17:19:19'** )

>>> ts = pd.Timestamp ( 1517246359.732405 , birlik = **'s'** )

>>> ts

Vaqt tamg'asi ( **'2018-01-29 17: 19: 19.732404947'** )

Timestamp moslamalarini yaratish uchun siz ushbu usuldan foydalanishingiz mumkin

to\_datetime (). Bu keng doiraga ega bo'lgan kuchli vosita

funktsional, biz uning barcha imkoniyatlarini hisobga olmaymiz, bizning

vazifa shunchaki uni bilish uchun:

>>> ts = pd.to\_datetime ( **'2018-01-01 00:01:02'** )

126

|  |
| --- |
| **Sahifa 128** |

>>> ts

Vaqt tamg'asi ( **'2018-01-01 00:01:02'** )

>>> ts = pd.to\_datetime ( **'20180101023215'** , format = **'% Y% m% d% H% M% S'** )

>>> ts

Vaqt tamg'asi ( **'2018-01-01 02:32:15'** )

>>> ts = pd.to\_datetime ( 1517246359 , birlik = **'s'** )

>>> ts

Vaqt tamg'asi ( **'2018-01-29 17:19:19'** )

E'tibor bering, vaqt tamg'asidan foydalanganda

vaqt oralig'idagi ma'lumotlar bilan quyidagi diapazonda ishlash:

>>> pd.Timestamp.min

Vaqt tamg'asi ( **'1677-09-21 00: 12: 43.145225'** )

>>> pd.Timestamp.max

Vaqt tamg'asi ( **'2262-04-11 23: 47: 16.854775807'** )

**10.1.2 Bir qator vaqt markalarini yaratish**

Vaqt tamg'asi ketma-ketligi DatetimeIndex sinfining ob'ekti hisoblanadi

*Series* va *DataFrame tuzilmalarini* yaratishda indeks vazifasini bajaradi .

Bunday sinfning ob'ekti bizga allaqachon ma'lum bo'lgan narsalar yordamida yaratilishi mumkin

to\_datetime () yoki date\_range () usuli.

Birinchidan, to\_datetime () bilan ishlash variantini ko'rib chiqamiz. Agar bo'lsa

argument sifatida ushbu uslubga qatorlar, raqamlar (int

yoki float) yoki Timestamp moslamalari bo'lsa, natija bo'ladi

mos keladigan DatetimeIndex ob'ekti:

>>> dti = pd.to\_datetime ([ **'2018-01-01'** , **'2018-01-02'** ])

>>> dti

DatetimeIndex ([ **'2018-01-01'** , **'2018-01-02'** ], dtype = **'datetime64 [ns]'** ,

frek = **yo'q** )

127

|  |
| --- |
| **Sahifa 129** |

Vaqt tamg'asi ob'ektlari ro'yxatidan foydalanish mumkin:

>>> dti = pd.to\_datetime ([pd.Timestamp ( **'2018-01-01'** ), pd.Timestamp ( **'2018-**

**01-02 '**)])

>>> dti

DatetimeIndex ([ **'2018-01-01'** , **'2018-01-02'** ], dtype = **'datetime64 [ns]'** ,

frek = **yo'q** )

>>> dti = pd.to\_datetime ([ 1517246359 , 1517246360 , 1517246361 ], birlik = **'s'** )

>>> dti

DatetimeIndex ([ **'2018-01-29 17:19:19'** , **'2018-01-29 17:19:20'** ,

**'2018-01-29 17:19:21'**],

dtype = **'datetime64 [ns]'** , freq = **Yo'q** )

Berilgan diapazondan vaqt seriyasini yaratish uchun amal qiling

date\_range () usuli:

pandas.date\_range (boshlash = **Hech** , end = **hech** , davrlar = **hech** , chastotasi = **'D'** , tz = **hech** ,

normalizatsiya = **Yolg'on** , ism = **Yo'q** , yopiq = **Yo'q** , \*\* kvarg)

Ushbu usulning dalillari orasida biz quyidagilarni ta'kidlaymiz:

• start: str; standart qiymat: Yo'q

Generated Yaratilgan ma'lumotlarning chap chegarasi.

• end: str; standart qiymat: Yo'q

Generated Yaratilgan ma'lumotlarning o'ng chegarasi.

• davr: tamsayı; standart qiymat: Yo'q

◦ Yaratilayotgan massivdagi elementlar soni.

• freq: str (yoki DateOffset); sukut bo'yicha: 'D'

Data Ma'lumotlarni yaratish bosqichi.

• tz: str; standart qiymat: Yo'q

◦ Soat zonasi, masalan: "Evropa / Bryussel".

128

|  |
| --- |
| **Sahifa 130** |

Date\_range () usuli DatetimeIndex sinfining ob'ektini qaytaradi

ma'lumotlar tuzilmalarini qurishda indeks sifatida foydalanish mumkin

*pandalar* .

Quyida, 10.2-jadvalda ba'zi mumkin bo'lgan qiymatlar berilgan

parametr tezligi.

**Jadval 10.2 - da qadamni belgilaydigan freq parametrining mumkin bo'lgan qiymatlari**

**date\_range () usuli yordamida diapazon yaratish**

**Freq parametr qiymati**

**Tavsif**

D.

Kun

V

Bir hafta

M

Oy

SM

Yarim oy (15 kun)

Q

Chorak

AY

Yil

H

Soat

T, min

Daqiqa

S

Ikkinchi

L, mil

Millisekund

U, biz

Mikrosaniyadagi

N

Nanosekundiya

129

|  |
| --- |
| **Sahifa 131** |

Bir soatlik qadamlar bilan vaqt tamg'alari qatorini yarating, bu

besh kunlik oraliqni o'z ichiga oladi:

>>> dt\_h = pd.date\_range ( start = **'2017-02-01'** , freq = **'H'** , davrlar = 120 )

>>> dt\_h [: 30 ]

DatetimeIndex ([ **'2017-02-01 00:00:00'** , **'2017-02-01 01:00:00'** ,

**'2017-02-01 02:00:00'**, **'2017-02-01 03:00:00'**,

**'2017-02-01 04:00:00'**, **'2017-02-01 05:00:00'**,

**'2017-02-01 06:00:00'**, **'2017-02-01 07:00:00'**,

**'2017-02-01 08:00:00'**, **'2017-02-01 09:00:00'**,

**'2017-02-01 10:00:00'**, **'2017-02-01 11:00:00'**,

**'2017-02-01 12:00:00'**, **'2017-02-01 13:00:00'**,

**'2017-02-01 14:00:00'**, **'2017-02-01 15:00:00'**,

**'2017-02-01 16:00:00'**, **'2017-02-01 17:00:00'**,

**'2017-02-01 18:00:00'**, **'2017-02-01 19:00:00'**,

**'2017-02-01 20:00:00'**, **'2017-02-01 21:00:00'**,

**'2017-02-01 22:00:00'**, **'2017-02-01 23:00:00'**,

**'2017-02-02 00:00:00'**, **'2017-02-02 01:00:00'**,

**'2017-02-02 02:00:00'**, **'2017-02-02 03:00:00'**,

**'2017-02-02 04:00:00'**, **'2017-02-02 05:00:00'**],

dtype = **'datetime64 [ns]'** , freq = **'H'** )

Xuddi shu tarzda, o'z ichiga olgan DatetimeIndex-ni yaratishingiz mumkin

vaqt shtamplari bir daqiqalik o'sishlarda va umumiy intervalda

soat:

>>> dt\_m = pd.date\_range ( start = **'2017-02-01'** , freq = **'min'** , davrlar = 60 )

>>> dt\_m [: 30 ]

DatetimeIndex ([ **'2017-02-01 00:00:00'** , **'2017-02-01 00:01:00'** ,

**'2017-02-01 00:02:00'**, **'2017-02-01 00:03:00'**,

**'2017-02-01 00:04:00'**, **'2017-02-01 00:05:00'**,

**'2017-02-01 00:06:00'**, **'2017-02-01 00:07:00'**,

**'2017-02-01 00:08:00'**, **'2017-02-01 00:09:00'**,

**'2017-02-01 00:10:00'**, **'2017-02-01 00:11:00'**,

**'2017-02-01 00:12:00'**, **'2017-02-01 00:13:00'**,

130

|  |
| --- |
| **132-bet** |

**'2017-02-01 00:14:00'**, **'2017-02-01 00:15:00'**,

**'2017-02-01 00:16:00'**, **'2017-02-01 00:17:00'**,

**'2017-02-01 00:18:00'**, **'2017-02-01 00:19:00'**,

**'2017-02-01 00:20:00'**, **'2017-02-01 00:21:00'**,

**'2017-02-01 00:22:00'**, **'2017-02-01 00:23:00'**,

**'2017-02-01 00:24:00'**, **'2017-02-01 00:25:00'**,

**'2017-02-01 00:26:00'**, **'2017-02-01 00:27:00'**,

**'2017-02-01 00:28:00'**, **'2017-02-01 00:29:00'**],

dtype = **'datetime64 [ns]'** , freq = **'T'** )

>>> len (dt\_m)

60

>>> len (dt\_h)

120

Keyingi qadamimiz *ketma-ket* tuzilmani yaratishdir , unda

indeks sifatida biz yaratgan birinchisi ishlatiladi

DatetimeIndex:

>>> tasodifiy **import**

>>> rnd = [random.randint (- 5 , 5 ) **uchun** i **ham** qator ( Len (dt\_h))]

>>> s\_dt = pd.Series (rnd, index = dt\_h)

>>> pd.options.display.max\_rows = 20

>>> s\_dt

2017 - 02 - 01 00 : 00 : 00 - 3

2017 - 02 - 01 01 : 00 : 00 4

2017 - 02 - 01 02 : 00 : 00 1

2017 - 02 - 01 03 : 00 : 00 2

2017 - 02 - 01 04 : 00 : 00 1

2017 - 02 - 01 05 : 00 : 00 - 2

2017 - 02 - 01 06 : 00 : 00 - 1

2017 - 02 - 01 07 : 00 : 00 0

2017 - 02 - 01 08 : 00 : 00 5

2017 - 02 - 01 09 : 00 : 00 0

131

|  |
| --- |
| **133-bet** |

..

2017 - 02 - 05 14 : 00 : 00 1

2017 - 02 - 05 15 : 00 : 00 - 4

2017 - 02 - 05 16 : 00 : 00 - 3

2017 - 02 - 05 17 : 00 : 00 3

2017 - 02 - 05 18 : 00 : 00 - 4

2017 - 02 - 05 19 : 00 : 00 - 3

2017 - 02 - 05 20 : 00 : 00 2

2017 - 02 - 05 21 : 00 : 00 2

2017 - 02 - 05 22 : 00 : 00 2

2017 - 02 - 05 23 : 00 : 00 - 3

Tezlik: H, Uzunlik: 120 , dip turi: int64

**10.2 Vaqt oraliqlari bilan ishlash**

**10.2.1 Vaqt oralig'ini yaratish**

Birlik vaqt oralig'i - bu Period sinfining ob'ekti. Siz .. qila olasiz; siz ... mumkin

yaratish, xuddi shu nomdagi konstruktordan foydalanib, asosiy parametrlar bilan

qiymati va tezligi:

• qiymat: str; standart qiymat: Yo'q

◦ Vaqt davri.

• freq: str; standart qiymat: Yo'q

◦ Vaqt oralig'ini belgilaydigan yorliqli qator.

Quyida qanday qilib ob'ektlarni qurish mumkinligiga misollar keltirilgan

*Davr* :

>>> pd.Period ( **'2018'** )

Davr ( **'2018'** , **'A-DEC'** )

>>> pd.Period ( **'2018-01'** )

Davr ( **'2018-01'** , **'M'** )

>>> pd.Period ( **'2018-01-01'** )

Davr ( **'2018-01-01'** , **'D'** )

132

|  |
| --- |
| **Sahifa 134** |

>>> pd.Period ( **'2018'** , freq = **'M'** )

Davr ( **'2018-01'** , **'M'** )

Period sinfining ob'ekti yaratilganidan so'ng siz ishlab chiqarishingiz mumkin

turli xil arifmetik amallar. Masalan: bilan ob'ekt yarataylik

kuniga oralig'ida:

>>> prd = pd.Period ( **'2018'** , freq = **'D'** )

>>> prd

Davr ( **'2018-01-01'** , **'D'** )

Endi unga 7 raqamini qo'shing, bu etti qo'shishni anglatadi

joriy sanaga kunlar:

>>> prd + 7

Davr ( **'2018-01-08'** , **'D'** )

Agar biz 31 dan katta sonni qo'shsak, oy o'zgarganligini ko'ramiz:

>>> prd + 53

Davr ( **'2018-02-23'** , **'D'** )

Xuddi shunday, agar biz interval joylashgan ob'ektni yaratadigan bo'lsak

oy, keyin arifmetik amallar oylar davomida amalga oshiriladi

belgilangan sanaga nisbatan:

>>> prd\_m = pd.Period ( **'2018'** , freq = **'M'** )

>>> prd\_m

Davr ( **'2018-01'** , **'M'** )

>>> prd\_m + 5

Davr ( **'2018-06'** , **'M'** )

133

|  |
| --- |
| **Sahifa 135** |

**10.2.2 Bir qator vaqt oralig'ini yaratish**

Qurilishda vaqt oralig'ining massivlaridan foydalanish mumkin

*pandalar* ma'lumotlar tuzilmalari . Bunday massivlar bilan ishlash uchun mavjud

maxsus sinf - PeriodIndex.

Ushbu sinf ob'ekti tomonidan yaratilgan

PeriodIndex () konstruktori yoki period\_range () usuli.

PeriodIndex konstruktori bilan ishlash

bilan ishlashga o'xshaydi

konstruktor Davr, faqat qiymat sifatida biz o'tishimiz kerak

vaqt tamg'alari ro'yxati:

>>> pd.PeriodIndex ([ **'2018'** , **'2017'** , **'2016'** ], freq = **'M'** )

PeriodIndex ([ **'2018-01'** , **'2017-01'** , **'2016-01'** ], dtype = **'davr [M]'** ,

freq = **'M'** )

>>> pd.PeriodIndex ([ **'2018'** , **'2017'** , **'2016'** ], freq = **'D'** )

PeriodIndex ([ **'2018-01-01'** , **'2017-01-01'** , **'2016-01-01'** ],

dtype = **'davr [D]'** , freq = **'D'** )

Vaqt seriyasini yaratish uchun qulayroq vosita

intervallar - bu printsipga muvofiq period\_range () usuli

u bilan ishlash "Qator yaratish" bo'limidagi date\_range () ga o'xshash

vaqt belgilari ":

pandas.period\_range ( boshlash = **Yo'q** , tugatish = **Yo'q** , davrlar = **Yo'q** , frekans = **'D'** ,

ism = **yo'q** ):

• start: str; sukut bo'yicha: yo'q

Generated Yaratilgan ma'lumotlarning chap chegarasi.

• end: str; standart qiymat: Yo'q

Generated Yaratilgan ma'lumotlarning o'ng chegarasi.

134

|  |
| --- |
| **136-bet** |

• davr: tamsayı; standart qiymat: Yo'q

◦ Massivdagi elementlar soni.

• freq: str (yoki DateOffset); sukut bo'yicha: 'D'

Data Ma'lumotlarni yaratish bosqichi.

• ism: str; standart qiymat: Yo'q

◦ PeriodIndex ob'ektining nomi.

Keling, ushbu usuldan foydalanishning bir nechta misollarini ko'rib chiqamiz. Qator

2018 va 2019 yillar oralig'idagi vaqt oraliqlari

bir oy ichida:

>>> pd.period\_range ( **'2018'** , **'2019'** , freq = **'M'** )

PeriodIndex ([ **'2018-01'** , **'2018-02'** , **'2018-03'** , **'2018-04'** , **'2018-05'** ,

**'2018-06'**, **'2018-07'**, **'2018-08'**, **'2018-09'**, **'2018-10'**, **'2018-11'**, **'2018-**

**12 '**, **' 2019-01 '**], dtype = **' davr [M] '**, freq = **' M '**)

2018 va 2019 yillar oralig'idagi qator vaqt oralig'i

bir kunlik o'sish bilan:

>>> pd.period\_range ( **'2018'** , **'2019'** , freq = **'D'** )

PeriodIndex ([ **'2018-01-01'** , **'2018-01-02'** , **'2018-01-03'** , **'2018-01-04'** ,

**'2018-01-05'**, **'2018-01-06'**, **'2018-01-07'**, **'2018-01-08'**,

**'2018-01-09'**, **'2018-01-10'**,

...

**'2018-12-23'**, **'2018-12-24'**, **'2018-12-25'**, **'2018-12-26'**,

**'2018-12-27'**, **'2018-12-28'**, **'2018-12-29'**, **'2018-12-30'**,

**'2018-12-31'**, **'2019-01-01'**],

dtype = **'davr [D]'** , uzunlik = 366 , freq = **'D'** )

135

|  |
| --- |
| **137-bet** |

2018.01.01 dan boshlab qadam bilan o'n vaqt oralig'ini yarating

haftada:

>>> pd.period\_range ( **'2018-01-01'** , davrlar = 10 , freq = **'W'** )

PeriodIndex ([ **'2018-01-01 / 2018-01-07'** , **'2018-01-08 / 2018-01-14'** ,

**'2018-01-15 / 2018-01-21'**, **'2018-01-22 / 2018-01-28'**,

**'2018-01-29 / 2018-02-04'**, **'2018-02-05 / 2018-02-11'**,

**'2018-02-12 / 2018-02-18'**, **'2018-02-19 / 2018-02-25'**,

**'2018-02-26 / 2018-03-04'**, **'2018-03-05 / 2018-03-11'**],

dtype = **'davr [W-SUN]'** , freq = **'W-SUN'** )

**10.3 Vaqt qatorlarini indeks sifatida ishlatish**

Timestamp, DatetimeIndex, Period, PeriodIndex sinflari ob'ektlari mumkin

*pandalar* ma'lumotlar tuzilmalarining indekslari sifatida ishlatiladi .

DatetimeIndex yarataylik - bir kunlik uzilish bilan bir qator vaqt tamg'alari:

>>> dt\_d = pd.date\_range ( start = **'2017-02-01'** , freq = **'D'** , davrlar = 50 )

>>> dt\_d

DatetimeIndex ([ **'2017-02-01'** , **'2017-02-02'** , **'2017-02-03'** , **'2017-02-04'** ,

**'2017-02-05'**, **'2017-02-06'**, **'2017-02-07'**, **'2017-02-08'**,

**'2017-02-09'**, **'2017-02-10'**, **'2017-02-11'**, **'2017-02-12'**,

**'2017-02-13'**, **'2017-02-14'**, **'2017-02-15'**, **'2017-02-16'**,

**'2017-02-17'**, **'2017-02-18'**, **'2017-02-19'**, **'2017-02-20'**,

**'2017-02-21'**, **'2017-02-22'**, **'2017-02-23'**, **'2017-02-24'**,

**'2017-02-25'**, **'2017-02-26'**, **'2017-02-27'**, **'2017-02-28'**,

**'2017-03-01'**, **'2017-03-02'**, **'2017-03-03'**, **'2017-03-04'**,

**'2017-03-05'**, **'2017-03-06'**, **'2017-03-07'**, **'2017-03-08'**,

**'2017-03-09'**, **'2017-03-10'**, **'2017-03-11'**, **'2017-03-12'**,

**'2017-03-13'**, **'2017-03-14'**, **'2017-03-15'**, **'2017-03-16'**,

**'2017-03-17'**, **'2017-03-18'**, **'2017-03-19'**, **'2017-03-20'**,

**'2017-03-21'**, **'2017-03-22'**],

dtype = **'datetime64 [ns]'** , freq = **'D'** )

136

|  |
| --- |
| **Sahifa 138** |

Olingan ob'ekt asosida *ketma-ket* tuzilmani yarating :

>>> tasodifiy **import**

>>> doktor = [random.randint (- 10 , 10 ) **uchun** i **ham** qator ( Len (dt\_d))]

>>> s = pd.Series (dr, index = dt\_d)

Strukturaga kirish uchun siz raqamli raqamdan foydalanishingiz mumkin

indekslar:

>>> lar [: 5 ]

2017 yil - 02 - 01 10

2017 yil - 02 - 02 - 7

2017 yil - 02 - 03 - 5

2017 yil - 02 - 04 - 4

2017 yil - 02 - 05 5

Tezlik: D, dip turi: int64

Yoki matnli vaqt tamg'asi:

>>> s [ **'2017-02-01'** ]

o'n

Ma'lum vaqt oralig'idan ma'lumotlarni olish uchun ruxsat beriladi

tilimlardan foydalanish:

>>> s [ **'2017-02-01'** : **'2017-02-10'** ]

2017 yil - 02 - 01 10

2017 yil - 02 - 02 - 7

2017 yil - 02 - 03 - 5

2017 yil - 02 - 04 - 4

2017 yil - 02 - 05 5

2017 yil - 02 - 06 5

2017 yil - 02 - 07 - 6

2017 yil - 02 - 08 8

2017 yil - 02 - 09 - 8

2017 yil - 02 - 10 3

Tezlik: D, dip turi: int64

137

|  |
| --- |
| **139-bet** |

Vaqt seriyasidan indekslar sifatida foydalanish

ma'lumotlar namunalarini olish uchun qo'shimcha imkoniyatlar. Biz ma'lumot olamiz

2017 yil Fevral yuqoridagi tuzilmadan:

>>> pd.options.display.max\_rows = 10

>>> s [ **'2017-02'** ]

2017 yil - 02 - 01 10

2017 yil - 02 - 02 - 7

2017 yil - 02 - 03 - 5

2017 yil - 02 - 04 - 4

2017 yil - 02 - 05 5

..

2017 yil - 02 - 24 - 1

2017 yil - 02 - 25 - 10

2017 - 02 - 26 5

2017 yil - 02 - 27 - 4

2017 - 02 - 28 4

Tezlik: D, uzunlik: 28 , dip turi: int64

Sana qatori bilan tasvirlash o'rniga ob'ektlarga ruxsat beriladi

datetime:

>>> **dan** DATETIME **import** DATETIME

>>> s [datetime ( 2017 , 2 , 1 ): datetime ( 2017 , 2 , 8 )]

2017 yil - 02 - 01 10

2017 yil - 02 - 02 - 7

2017 yil - 02 - 03 - 5

2017 yil - 02 - 04 - 4

2017 yil - 02 - 05 5

2017 yil - 02 - 06 5

2017 yil - 02 - 07 - 6

2017 yil - 02 - 08 8

Tezlik: D, dip turi: int64

138

|  |
| --- |
| **140-bet** |

**11-bob. Ma'lumotlarni vizuallashtirish**

**11.1 Rejalashtirish**

*Pandalar* kutubxonasida ma'lumotlarni vizualizatsiya qilishning asosiy vositasi

bu tuzilish ob'ekti bo'yicha chaqirilishi mumkin bo'lgan plot () usuli

*Series* yoki *DataFrame* . Agar siz ushbu usulni parametrlarsiz ishlatsangiz, unda

chiziqli grafik tuziladi. Grafik ko'rinishi uchun argument javobgardir.

mehribon. Uning qiymatiga qarab shakl o'zgaradi

ma'lumotlarning grafik taqdimoti. Buning mumkin bo'lgan qiymatlari

parametrlari 11.1-jadvalda keltirilgan.

**Jadval 11.1 - turdagi parametr qiymatlarining tavsifi**

**Parametr qiymati**

**Tavsif**

"bar" yoki "barh"

Shtrixli jadval tuzish

"tarix"

Qurilish gistogrammalari

"quti"

Kassa diagrammasi (mo'ylovli quti)

'kde'

Plotirovka zichligi

"maydon"

Maydon diagrammasi

"tarqoq"

Nuqta uchastkasi

"hexbin"

Ma'lumotlarni vizualizatsiya qilish

olti burchakli

"pirog"

Doira diagrammasi

139

|  |
| --- |
| **141-bet** |

Yuqoridagi yondashuv qanday ishlashini misol bilan ko'rsatamiz:

>>> pandalarni pd **sifatida import qilish**

>>> s = pd seriyalari ([ 1 , 2 , 3 , 4 , 3 , 2 , 1 ])

>>> s.plot ()

>>> s.plot ( kind = **'bar'** )

*Pandalar* diagrammalar *tuzishning* yana bir usulini taqdim etadi. Uning mohiyati

biz plot () usulining yaxshi argumentidan foydalanmasligimiz,

va biz kerakli diagrammani qurish uchun maxsus usulni chaqiramiz

<pandas tarkibi> .plot. <chizish usuli> formatida.

140

|  |
| --- |
| **142-bet** |

Shtrixli jadval uchun quyidagicha ko'rinadi:

>>> s.plot.bar ()

**11.1.1 Chiziqlar**

Keling, qurilish jadvallarini batafsil ko'rib chiqaylik. Buning uchun

ma'lumotlar to'plamini tayyorlaylik:

>>> tasodifiy **import**

>>> rnd = [random.randrange (- 10 , 10 ) **uchun** i **ham** qator ( 100 )]

>>> s = pd.Series (rnd)

Grafik tuzamiz:

>>> s.plot ()

141

|  |
| --- |
| **143-bet** |

Agar biz DatetimeIndex-dan indeks sifatida foydalansak, u holda abstsissa bo'ylab

tegishli vaqt tamg'alari o'rnatiladi:

>>> datetime\_index = pd.date\_range ( **'2017.11.01'** , freq = **'H'** ,

davrlar = len (rnd))

>>> s\_dt = pd.Series (rnd, index = datetime\_index)

>>> s\_dt.plot ()

Hisoblangan statistika uchun grafikalar tuzish qulay

Masalan, kengaygan oynaning yig'indisini olaylik:

>>> ex = s\_dt.expanding ()

>>> ex.sum (). fitna ()

142

|  |
| --- |
| **144-bet** |

Agar siz *DataFrame* bilan *ishlasangiz* , unda bitta maydonda ko'rsatishingiz mumkin

bir vaqtning o'zida bir nechta grafikalar, ularning har biri mos keladi

tuzilish ustuni:

>>> d\_arr = [[random.randrange (- 10 , 10 ), random.randrange ( 0 , 30 )] **uchun** i

**yilda** qator ( 100 )]

>>> df = pd.DataFrame (d\_arr, ustunlar = [ **'A1'** , **'A2'** ])

>>> df.plot ()

**11.1.2 ustunlar jadvallari**

Ma'lumotlarni taqqoslash uchun ustunli jadvallar (chiziqlar) yaxshi.

agar ular (ma'lumotlar) shunga muvofiq tayyorlangan bo'lsa.

Bo'lsin *»s* misol sifatida DataFrame bir görselleştirme olish . Uchun,

10x3 DataFrame yaratish uchun dan usulidan foydalaning

avval import qilishingiz kerak bo'lgan numpy kutubxonasi (va

agar u allaqachon kompyuteringizda bo'lmasa o'rnatib qo'ying):

>>> numpy- **ni** np **sifatida import qilish**

>>> df1 = pd.DataFrame (np.random.randint ( 5 , size = ( 5 , 5 )), ustunlar = [ **'A1'** ,

**'A2'**, **'A3'**, **'A4'**, **'A5'**])

143

|  |
| --- |
| **Sahifa 145** |

>>> df1

A1 A2 A3 A4 A5

0 1 4 0 2 1

1 4 1 1 0 4

2 2 3 4 3 3

3 2 0 1 3 0

4 4 2 4 3 4

>>> df1.plot ( kind = **'bar'** )

>>> df1.loc [ 3 ] .plot ( kind = **'bar'** )

144

|  |
| --- |
| **146-bet** |

Gorizontal chiziqli jadvalni tuzish uchun muloyim dalil

"barh" qiymatini belgilashingiz kerak *:*

>>> df1.loc [ 3 ] .plot ( kind = **'barh'** )

>>> df1.plot ( turdagi = **'barh'** )

145

|  |
| --- |
| **147-bet** |

Ma'lumotlar to'plamini ko'rsatish uchun parametrdan foydalaning

stacked = plot () usuli haqiqiy:

>>> df1.plot ( kind = **'barh'** , stacked = **True** )

**11.1.3 Gistogrammalar**

*Pandalar* bar diagrammasi imkoniyatlarini joriy etish

0 dan 20 s gacha bo'lgan raqamlarni o'z ichiga olgan 50x5 DataFrame yarating

kvadratchalar bo'yicha taqsimlash:

>>> df2 = pd.DataFrame (np.random.chisquare ( 20 , size = ( 50 , 5 ))),

ustunlar = [ **'A'** , **'B'** , **'C'** , **'D'** , **'E'** ])

Keling, ko'rib chiqqan ikkinchisidan foydalanib, gistogramma tuzamiz

bobning boshlanishi:

>>> df2.plot.hist ()

146

|  |
| --- |
| **148-bet** |

Diagrammaning yo'nalishini o'zgartirish uchun siz hist () uslubiga o'tishingiz kerak

orientatsiya = "gorizontal" parametr:

>>> df2.plot.hist ( orientatsiya = **'gorizontal'** )

**11.1.4 to'ldirilgan grafik**

To'ldirilgan jadval - bu *maydon uchastkasining* bo'sh tarjimasi , u shunga o'xshash

egri chiziq ostida bo'sh joy to'ldirilgan oddiy grafika kabi.

Tasodifiy ma'lumotlar bilan *ketma-* ketlikni yaratamiz va to'ldirilgan grafikani tuzamiz :

>>> s = pd.Series ([random.randrange ( 2 , 10 ) **uchun** I **-yilda** qator ( 7 )])

>>> s.plot.area ()

147

|  |
| --- |
| **149-bet** |

To'ldirishning shaffofligini alfa parametri orqali o'zgartirishingiz mumkin:

>>> s.plot.area ( alfa = 0.7 )

Keling, ushbu vosita *DataFrame* bilan qanday ishlashini ko'rib chiqamiz :

>>> df3 = pd.DataFrame (np.random.chisquare ( 10 , size = ( 7 , 3 )), ustunlar = [ **'A'** ,

**'B'**, **'C'**])

>>> df3.plot.area ( alfa = 0.7 )

148

|  |
| --- |
| **Sahifa 150** |

>>> df3.plot.area ( stacked = **False** , alfa = 0.3 )

**11.1.5 nuqta uchastkasi**

Tarqoqlik sxemasini argument turiga qiymat berish orqali olish mumkin

'tarqatish' yoki scatter () usulidan foydalaning.

Namoyish uchun *DataFrame* tuzilishini avvalgisidan o'zgartiramiz .

misol:

>>> df4 = pd.DataFrame (np.random.normal ( 10 , size = ( 70 , 2 )), ustunlar = [ **'X'** ,

**'Y'**])

>>> df4.plot.scatter ( x = **'X'** , y = **'Y'** )

149

|  |
| --- |
| **Sahifa 151** |

Rangni belgilaydigan yana bir ustun qo'shamiz

(intensivlik) punktlari:

>>> df5 = pd.DataFrame (np.random.chisquare ( 10 , size = ( 70 , 3 )),

ustunlar = [ **'X'** , **'Y'** , **'Z'** ])

>>> df5.plot.scatter ( x = **'X'** , y = **'Y'** , c = **'Z'** )

**11.1.6 Dairesel jadval**

Dairesel diagramma miqdoriy ko'rinishni tasavvur qilish imkonini beradi

ma'lum bir ma'lumotlar to'plamining elementlari o'rtasidagi munosabatlar.

Keling, strukturani yarataylik, uning elementlari mahsulot miqdori bo'ladi:

>>> s = pd.Series ([ 10 , 15 , 7 ], index = [ **'prod\_A'** , **'prod\_B'** , **'prod\_C'** ])

Keling, ushbu to'plam uchun dairesel diagramma tuzamiz:

>>> s.plot.pie ()

150

|  |
| --- |
| **Sahifa 152** |

Diagrammaning o'lchamini figsize parametri yordamida o'rnatish mumkin:

>>> s.plot.pie ( figsize = ( 5 , 5 ))

Diagrammaning sektorlarida siz qo'shimcha ravishda raqamli ko'rsatishingiz mumkin

sektor hajmiga mos keladigan qiymatlar:

>>> s.plot.pie ( figsize = ( 5 , 5 ), autopct = **'% d'** , fontsize = 12 )

151

|  |
| --- |
| **Sahifa 153** |

Diagrammaning nomini ko'rsatish uchun avval uning (nomi) bo'lishi kerak

tegishli *pandalar* tuzilishini tayinlang . Bu bilan amalga oshirilishi mumkin

argumentga kerakli qiymatni o'rnatish orqali strukturani yaratish:

>>> s = pd.Series ([ 10 , 15 , 7 ], index = [ **'prod\_A'** , **'prod\_B'** , **'prod\_C'** ],

name = **'Do'kon'** )

Yoki uni allaqachon yaratilgan tuzilishga tayinlang:

>>> s.name = **"Do'kon"**

Ushbu ism chizilgan diagramma yonida ko'rsatiladi:

>>> s.plot.pie ( figsize = ( 5 , 5 ), autopct = **'% d'** , fontsize = 12 )

**11.1.7 olti burchakli diagramma**

Olti burchakli diagramma juda qiziq. Uchun

Namoyish uchun qo'shimcha *DataFrame* yarataylik :

>>> df6 = pd.DataFrame (np.random.laplace ( 10 , size = ( 70 , 3 )), ustunlar = [ **'X'** ,

**'Y'**, **'Z'**])

152

|  |
| --- |
| **Sahifa 154** |

Tegishli diagrammani tuzamiz:

>>> df6.plot.hexbin ( x = **'X'** , y = **'Y'** , panjara = 15 )

Olti burchakning kattaligi panjara parametri orqali o'rnatiladi.

**11.2 Diagramma ko'rinishini sozlash**

*Pandalardagi* jadvallar bilan ishlashda bizning ixtiyorimizda

tashqi moslashtirish uchun juda ko'p sonli vositalar

diagrammaning o'zi va qo'shimcha chiqishi ko'rinishi

elementlar (jadvallar, afsona va boshqalar).

Oldingi bo'limda har xil turdagi grafikalar haqida umumiy ma'lumot berilgan,

qurilishi mumkin. Tashqi ko'rinishini o'zgartirish uchun

diagrammalar - rang, chiziq turi va boshqalar uzatilishi kerak

argument orqali kerakli xususiyatga mos keladigan qiymat.

**11.2.1 Chiziqli diagrammaning ko'rinishini sozlash**

Agar parametrlarni standart qiymatlarida qoldirsangiz, u holda chiziqli

grafik quyidagicha ko'rinadi:

>>> rnd = [random.randrange (- 10 , 10 ) **uchun** i **ham** qator ( 20 )]

>>> s = pd.Series (rnd)

>>> s.plot ()

153

|  |
| --- |
| **Sahifa 155** |

Grafika ko'rinishini o'zgartirish uchun sizga dalil kerak

tegishli qiymatni belgilash uchun fitna () uslubining uslubi.

Masalan, s tuzilmasidagi ma'lumotlarni yulduzlarsiz ko'rsatamiz

ulanish liniyalari:

>>> s.plot ( style = **'r \*'** )

154

|  |
| --- |
| **Sahifa 156** |

Agar siz grafikani chiziqli chiziq sifatida ko'rsatishingiz kerak bo'lsa, unda buning uchun

uslub argumenti *"r--" ga* o'rnatilishi kerak :

>>> s.plot ( style = **'r--'** )

Quyidagi jadvallarda tashqi uchun belgilar ko'rsatilgan

grafik turi.

**Jadval 11.2 - Rangni belgilaydigan belgilar**

**Belgilar**

**Rang**

"b"

ko'k

"g"

yashil

"r"

qizil

"c"

ko'k

"m"

siyohrang

"y"

sariq

"k"

qora

"w"

oq

155

|  |
| --- |
| **Sahifa 157** |

**Jadval 11.3 - chiziq / nuqta turini belgilaydigan belgilar**

**Belgilar**

**Tavsif**

'-'

qattiq chiziq

'-'

kesilgan chiziq

'-.'

nuqta-chiziq chizig'i

':'

nuqta chiziq

'.'

marker: nuqta

','

marker: piksel

"o"

marker: doira

"v"

marker: yo'naltirilgan uchburchak

pastga tepaga

'^'

marker: yo'naltirilgan uchburchak

yuqoriga

'<'

marker: yo'naltirilgan uchburchak

chap yuqori

'>'

marker: yo'naltirilgan uchburchak

yuqori o'ng

'1'

marker: uch qirrali yulduz,

pastga nur

"2"

marker: uch qirrali yulduz,

yuqoriga ko'tarilgan nur

"3"

marker: uch qirrali yulduz,

nur chap

"4"

marker: uch qirrali yulduz,

o'ng nurli

bu

marker: kvadrat

"p"

marker: beshburchak

'\*'

marker: yulduz

"h"

marker: olti burchak (1-tur)

"H"

marker: olti burchakli (2-toifa)

"+"

marker: ortiqcha

"x"

marker: x

"D"

olmos belgisi

"d"

ingichka\_ olmosli marker

'|'

vline markeri

'\_'

hline markeri

156

|  |
| --- |
| **Sahifa 158** |

**11.2.2 Grafiklarni turli tekisliklarda aks ettirish**

Grafiklarni turli sohalarda ko'rsatish uchun sizga dalil kerak

subplotlari True ga, agar kerak bo'lsa o'rnatildi

ko'rsatilgan jadvallarning hajmi.

Keling, yangi *DataFrame* yarataylik :

>>> d\_arr = [[random.randrange (- 10 , 10 ), random.randrange ( 0 , 30 )] **uchun** i

**yilda** qator ( 10 )]

>>> df = pd.DataFrame (d\_arr, ustunlar = [ **'A1'** , **'A2'** ])

Grafiklarni bitta maydonda namoyish qilaylik:

>>> df.plot ()

Endi ularni turli sohalarga ajratamiz:

>>> df.plot ( subplots = **True** )

157

|  |
| --- |
| **Sahifa 159** |

Ularni ko'rishga qulay bo'lishi uchun biz kerakli narsani o'rnatamiz

hajmi:

>>> df.plot ( pastki chiziqlar = **To'g'ri** , shaklsiz = ( 7 , 5 ))

Grafiklarning tartibini o'zgartirishingiz mumkin, bu parametr orqali amalga oshiriladi

layout, unga qiymat sifatida tuple beriladi, birinchisi

argumenti qatorlar soni, ikkinchisi ustunlar soni,

yaratiladigan maket:

>>> df.plot ( pastki chizmalar = **Rost** , layout = ( 1 , 2 ), figsize = ( 10 , 4 ))

158

|  |
| --- |
| **Sahifa 160** |

**12-bob. Jadvallarning ko'rinishini sozlash**

Agar siz *Yupyter Notebook-da* ishlayotgan bo'lsangiz, ehtimol siz allaqachon baho berishga muvaffaq bo'lgansiz,

*pandalar* ko'rsatilganda ko'rsatiladigan jadvallarning ko'rinishi .

*REPL-* dan foydalanganda nima chiqishini solishtiring :

>>> d = [{ **"name"** : **"pen"** , **"price"** : 3.9 , **"count"** : 8 },

{ **"name"** : **"book"** , **"price"** : 4.5 , **"count"** : 11 },

{ **"name"** : **"folder"** , **"price"** : 10.3 , **"count"** : 7 }]

>>> df = pd.DataFrame (d)

>>> df

ism narxini hisoblash

0 8 qalam 3.9

1 11 kitob 4.5

2 7 papka 10.3

va *Jupyter Notebook-da* bir xil kodni ishlatish natijasi :

**Rasm 12,1 - bir Matnni *DataFrame* bir jadval *Jupyter daftar***

Ammo bu siz olishingiz mumkin bo'lgan eng oson variant. *Pandalar*

jiddiy ruxsat beradigan vositalarni taqdim etadi

ko'rsatilgan ma'lumotlarning vizual komponenti ustida ishlash, bu

ularning o'qilishi va ifodalanishini sezilarli darajada oshiradi.

159

|  |
| --- |
| **Sahifa 161** |

Birinchidan, quyidagilar bilan ishlash uchun ma'lumotlar bazasini yarataylik:

np **sifatida**numpy- **ni import qilish**

np.random.seed ( 123 )

df = pd.DataFrame (np.random.randn ( 7 , 5 ), ustunlar = [ **'a'** , **'b'** , **'c'** , **'d'** ,

**'e'**])

Biz quyidagi jadvalni olamiz:

**12.2-rasm - *DataFrame* jadvali qo'shimcha formatlashsiz**

Uslubni sozlash uchun quyidagi usullardan foydalanish mumkin:

• Styler.applymap (): elementlar bo'yicha ishlashni ta'minlaydi

ma'lumotlar. Usul qo'llaniladigan funktsiyani qabul qiladi

*pandalar* tuzilishining har bir elementi . O'tkazilgan funktsiya

skalar bo'lishi va *CSS* elementini qaytarishi kerak

"attribute: value" formatida.

• Styler.apply (): ishlov berish birligi ustun, satr

yoki butun stol. Bu argument sifatida funktsiyani oladi,

uning argumenti *Series* yoki *DataFrame tuzilmasi* va

u elementlar bilan bir xil o'lchamdagi strukturani qaytarishi kerak

bu formatdagi *CSS* elementlari bo'lgan satrlar

"Xususiyat: qiymat".

160

|  |
| --- |
| **Sahifa 162** |

Ma'lumotlarning uslubi taqdimotini o'zgartirishdan tashqari, siz ham qilishingiz mumkin

to'g'ridan-to'g'ri ma'lumotlarning o'ziga o'zgartirishlar kiritish, masalan:

foiz yoki valyuta belgisini qo'shish, o'nli kasrni cheklash

belgilar va boshqalar, bu Styler.format () funktsiyasi yordamida amalga oshiriladi.

12.1 Ma'lumotlarni taqdim etish formatini o'zgartirish

Ma'lumotlarni taqdim etish formatini o'zgartirish uchun funktsiyadan foydalaning

Formatlashtiruvchini argument sifatida berilgan Styler.format () -

qoidalar qatori

*Format xususiyatlari*

[https://docs.python.org/3/library/string.html#format-specification-mini-](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=https://docs.python.org/3/library/string.html%23format-specification-mini-language#format-specification-mini-language)

[til](https://translate.google.com/translate?hl=uz&prev=_t&sl=auto&tl=uz&u=https://docs.python.org/3/library/string.html%23format-specification-mini-language#format-specification-mini-language), ma'lumotlar elementini qabul qiluvchi lug'at yoki funktsiya va

bu uning formatlangan mag'lubiyat vakolatini qaytaradi.

Format () ning qanday ishlashini ko'rsatish uchun asl ma'lumotlar to'plamini olaylik,

qiymatlarni uchinchi o'nli kasrga aylantiring va belgilar qo'shing,

Selsiy shkalasi bo'yicha haroratni bildiradi:

>>> df.style.format ( **"{: .3} ° C"** )

**12.3-rasm - Dastlabki tuzilmaning ma'lumotlari o'zgartirilgan jadval**

161

|  |
| --- |
| **Sahifa 163** |

Formatlarning tavsiflari bilan lug'atni oldindan tuzishingiz mumkin

ustunlar va keyin uni formatlashtiruvchi sifatida o'tkazing:

f\_dict = { **'a'** : **'{: .3} ° C'** , **'b'** : **'{0: .1}'** , **'c'** : **'{: .2}%'** , **'d'** : **'$ { : .2} '** }

df.style.format (f\_dict)

**12.4-rasm - Dastlabki tuzilmaning ma'lumotlari o'zgartirilgan jadval**

Agar formatlash yanada murakkab mantiqni talab qilsa, unda siz buni qila olasiz

formatlashtiruvchi uzatilganda parametrdan foydalaning

funktsiyasi:

df.style.format ( **lambda** x: **Yo'q, agar** x < 0 **bo'lmasa "{: .2f}"** .format (x))

**12.5-rasm - Asl tuzilma ma'lumotlari o'zgartirilgan jadval**

***lambda* funktsiyasi**

162

|  |
| --- |
| **Sahifa 164** |

12.2 O'z uslublaringizni yaratish

Qanday qilib yaratishingiz mumkinligi haqida turli xil variantlarni ko'rib chiqing

jadval ma'lumotlarining ko'rinishini o'zgartirish uchun o'z uslublari.

Buning uchun biz qaytib keladigan funktsiyani yaratishimiz kerak

ma'lumotlar elementining stilize qilingan qiymati (satr / ustun / butun)

va Styler.applymap () yoki Styler.apply () ga o'tkazing.

**12.2.1 Ma'lumotlar elementlari uchun yorliq rangini belgilash**

Yozuv rangini belgilash uchun biz o'rnatadigan funktsiyani aniqlaymiz

noldan kam qiymatlar uchun qizil va qiymatlar uchun yashil

ikkitadan ko'p:

**def**font\_color\_mod (val):

color\_val = **"qora"**

**agar**val < 0 :

color\_val = **"qizil"**

**elif**val> = 2 :

color\_val = **"yashil"**

**qaytib ": {color\_val} rang" f**

df\_font\_mod = df.style.applymap (font\_color\_mod)

df\_font\_mod

Keling, quyidagi uslubiy jadvalni olamiz:

**12.6-rasm - element yorliqlarining rangi o'zgargan jadval**

163

|  |
| --- |
| **Sahifa 165** |

**12.2.2 Jadval katakchasining rangini sozlash**

Apply () usuli ustunlar / qatorlar bilan ishlash uchun ishlatiladi,

unga funktsiya argument sifatida uzatiladi, while, tomonidan

sukut bo'yicha ustunlar unga *ketma-ket* ob'ektlar shaklida uzatiladi ,

agar chiziqni chiziq bilan ishlov berish kerak bo'lsa, siz o'rnatishingiz kerak

eksa parametri orqali yo'nalish:

• o'q = 0 - ustunlar o'tishini belgilaydi;

• o'q = 1 - chiziqlar o'tishini belgilaydi.

Eng kattasini tanlaydigan funktsiyani yozamiz

eng kichik qiymat:

**deflight\_min\_max\_in\_row**(qator):

min\_map = qator == qator.min ()

max\_map = satr == satr.max ()

style\_table = []

**uchun**m **yilda** zip (min\_map, max\_map):

tmp = **'background-color: skyblue' if** m [ 0 ] **else ''**

tmp = **'background-color: orange' if** m [ 1 ] **else** tmp **if** len (tmp)> 0

**boshqa "**

style\_table.append (tmp)

**Qaytish**style\_table

df.style.apply (ta'kidlash\_min\_max\_in\_row, o'q = 1 )

**12.7-rasm - hujayralar rangi o'zgargan jadval**

164

|  |
| --- |
| **Sahifa 166** |

**12.2.3 Jadval qatorining rangini belgilash**

Siz jadvalning ustuniga yoki qatoriga rang o'rnatishingiz mumkin, buning uchun sizga kerak bo'ladi

ustun / satrning barcha katakchalari uchun fon rang qiymatini o'rnating.

Keling, jadvalning barcha qatorlarini tanlaymiz, ularda qiymat

modul ikkitadan ko'p:

**deflight\_row**(qator):

style\_table = [ **'background-color: mercan'** ] \* len (satr) **agar** abs (satr) .max ()

> 2 **boshqa** [ **'** ] \* Len (satr)

**Qaytish**style\_table

df.style.apply (light\_row, axis = 1 )

**12.8-rasm - qator o'zgargan jadval**

**12.3 Ichki uslublar vositalari**

*Pandalar* kutubxonasi

juda ko'p turli xillarni o'z ichiga oladi

stol ko'rinishini o'rnatish uchun o'rnatilgan vositalar.

Keling, ulardan ayrimlarini ko'rib chiqaylik.

165

|  |
| --- |
| **Sahifa 167** |

**12.3.1 Minimal va maksimal qiymatlarni ajratib ko'rsatish**

Oldingi misollardan birida biz o'zimiz minimalni aniqladik /

maksimal element va *CSS* atributi orqali hujayra rangini *o'rnating* .

*Pandalar* ushbu vazifani bajarish uchun ichki usullarni taqdim etadi:

df.style.highlight\_max ( o'q = 1 )

**12.9-rasm - belgilangan maksimal qiymat bilan jadval**

df.style.highlight\_min ( o'q = 1 )

**Shakl 12.10 - belgilangan minimal qiymat bilan jadval**

166

|  |
| --- |
| **Sahifa 168** |

**12.3.2 bo'sh elementlarni ajratib ko'rsatish**

Mavjud vositalar orasida null elementlarni ta'kidlashni ta'kidlaymiz.

Keling, strukturaning ba'zi elementlarini null qiymatga o'rnatamiz:

df.iloc [ 3 , 4 ] = **Hech narsa**

df.iloc [ 4 , 1 ] = **Hech narsa**

**12.11-rasm - Dastlabki ma'lumotlar keltirilgan jadval**

Endi biz null- ni ta'kidlash uchunlight\_null funktsiyasidan foydalanamiz

elementlar:

df.style.highlight\_null ( null\_color = **'qip-qizil'** )

**12.12-rasm - nol qiymatlarni ta'kidlagan jadval**

167

|  |
| --- |
| **Sahifa 169** |

**12.3.3 Issiqlik xaritasini o'rnatish**

Jadvalni palitraga rang berish uchun funktsiyadan foydalaning

background\_gradient ():

df.style.background\_gradient ( cmap = **'plazma'** )

**12.13-rasm - issiqlik xaritasi ko'rsatilgan jadval**

**12.3.4 Ustunlar jadvalining ustki qatlami**

Deb yana bir qiziqarli xususiyatlar *pandas* beradi deb

jadvalda ustma-ust qo'yilgan ustunli jadval bilan, buning yordamida siz qilishingiz mumkin

raqamli ma'lumotlarning taqsimlanishini taxmin qiling:

df.style.bar ( subset = [ **'a'** , **'c'** ], color = **'cadetblue'** )

**12.14-rasm - Shtrixli jadval**

168

|  |
| --- |
| **Sahifa 170** |

**12.3.5 Tuning uchun *usul zanjiri***

**stol ko'rinishi**

Agar siz bir vaqtning o'zida stol ko'rinishini sozlashingiz kerak bo'lsa, unda bu

hisoblash zanjirini qurish orqali amalga oshirilishi mumkin, bu

uslublarni birin ketin qo'llash usullarini chaqirishga imkon beradi:

(df.style

.applymap (font\_color\_mod)

.bar ( subset = [ **'a'** , **'c'** ], color = **'cadetblue'** )

.highlight\_null ( null\_color = **'qip-qizil'** ))

**12.15-rasm - ketma-ket ketma-ketliklardan foydalangan holda jadval**

**o'zgartirishlar**

169

|  |
| --- |
| **Sahifa 171** |

**Xulosa**

Shu o'rinda kutubxona imkoniyatlarini ko'rib chiqamiz.

*pandalar* , biz muhimlariga e'tibor qaratishga harakat qildik, bizning fikrimizcha,

ushbu vositaning jihatlari. Umid qilamizki, sizga ma'lumot

kitob sahifalarida uchrashdi, foydali bo'ldi. agar sizda bo'lsa

sharhlar yoki tarkib bo'yicha takliflar, keyin bizga yozing

*devpractice.mail@gmail.com* , biz siz bilan *gaplashishni* istaymiz.

170