PYTHONNING MAP, FILTER, REDUSE FUNKSIYALARINING IMKONIYATLARI

Siddiqov Murodali Yo‘ldoshali o‘g‘li
Isaxonov Xushnidbek Murodiljon o‘g‘li
Sotvoldiyev Asadbek Abrorjon o‘g‘li
Muhammad al-Xorazmiy nomidagi TATU Farg‘ona Filiali talabalari

Annotatsiya: *Yuqori tartibli funksiyalar map, filter va reduse dasturlashda eng keng tarqalgan va kuchli vositalardir. Ular sizga ma’lumotlar to‘plami bilan ishlash imkonini beradi. Ushbu funktsiyalar ma’lumotlar bilan ishlashni soddalashtiradi va tezlashtiradi va sizga toza va samaraliroq kod yozish imkonini beradi.*

Kalit so‘zlar: *for, map, reduse, filter, iterable, initial.*

Aytaylik, biz raqamlar ro‘yxatining har bir elementiga funksiya yoki amalni qo‘llamoqchimiz. Ro‘yxat elementlarini takrorlash va har bir elementga amallarni qo‘llash uchun for siklidan foydalanishimiz mumkin.

Masalan, bizda raqamlar ro‘yxati mavjud. Biz yangi ro‘yxat yaratmoqchimiz, unda har bir elementni kvadrat ko‘tarib, keyin beshga ayiriladi. Biz bu muammoni for sikli orqali hal qilishimiz mumkin:

```
sonlar= [1, 2, 3, 4, 5]
yangi_sonlar = []
for son in sonlar:
kvadrat= son ** 2
ayirma = kvadrat – 5
yangi_sonlar.append(ayirma)
print (yangi_sonlar)
# [-9, -6, -1, 6, 15]
```

Ushbu yondashuvning kamchiliklari bor: kod hajmi katta bo‘ladi, o‘qish va

saqlash qiyin va ishlashda ko‘p vaqt oladi. Bunday holda siz to‘plamning har bir elementiga funktsiyani qo‘llash imkonini beruvchi map funksiyasidan foydalanishingiz mumkin. O‘zgartirilgan qiymatlar bilan yangi ro‘yxatni qaytaradi.

Ushbu sikl yuqoridagi misolga o‘xshaydi. Bu erda biz bo‘sh ro‘yxat natijasini yaratamiz va keyin o‘tkazilgan takrorlanadigan ob’ektning elementlari bo‘ylab aylanamiz. Har bir element uchun funksiyani ishlatamiz va uning natijasini natijalar ro‘yxatiga qo‘shamiz. Loop tugagach, biz natijalar ro‘yxatini qaytaramiz.

Keling, map funksiyasini muammomizga qo‘llashga harakat qilaylik:

```
def kvadrat_va_ayirma(son):
kvadrat = son ** 2
ayirma = kvadrat – 5
return ayirma
sonlar = [1, 2, 3, 4, 5]
yangi_sonlar = map (kvadrat_va_ayirma, sonlar)
print (yangi_sonlar)
# [-9, -6, -1, 6, 15]
```

Ushbu misolda biz birinchi navbatda ro‘yxatning har bir elementiga qo‘llamoqchi bo‘lgan **kvadrat_va_ayirma** funksiyasini aniqladik. Keyin biz uni raqamlar ro‘yxati bilan map funktsiyasiga o‘tkazdik.

Map funktsiyasi bizga ma’lumotlar to‘plamini yanada samarali va qisqacha qayta ishlashga imkon berdi. Bu funktsiyani to‘plamning har bir elementiga qo‘lladi va o‘zgartirilgan qiymatlar bilan yangi ro‘yxatni qaytardi.

Ko‘pincha dasturlashda ma’lumotlar to‘plamini filtrlash kerak – ya‘ni to‘plamdan faqat ma’lum shartlarni qondiradigan elementlarni tanlang.

Misol uchun, agar bizda raqamlar ro‘yxati bo‘lsa va biz faqat beshdan katta bo‘lgan raqamlarni olishni istasak, biz ro‘yxat elementlarini takrorlash va har bir element uchun shartlarni tekshirish uchun for siklidan foydalanishimiz mumkin:

```
sonlar = [2, 7, 1, 8, 4, 5]
natija = []
for son in sonlar:
```

```
if son > 5:
natija.append(son)
print(natija)
# [7, 8]
```

Biroq, bu yondashuvning kamchiliklari ham bor: kod noqulay ,o‘qish qiyin bo‘ladi va sekin ishlaydi.

Bunday holda biz filter funksiyasidan foydalanishimiz mumkin. U bizga berilgan shart asosida to‘plam elementlarini filtrlash va shu shartni qondiradigan elementlar bilan yangi to‘plamni qaytarish imkonini beradi.

```
def filter(func, iterable):
result = []
for item in iterable:
if func(item):
result.append(item)
return result
```

Bu yerda biz bo‘sh ro‘yxat natijasini yaratamiz va keyin o‘tkazilgan takrorlanadigan ob’ektning elementlari bo‘ylab aylanamiz. Har bir element uchun func funksiyasini chaqiramiz va u berilgan shartni qanoatlantirishini tekshiramiz.

Agar shart rost bo‘lsa, biz elementni natijalar ro‘yxatiga qo‘shamiz. Loop tugagach, biz natijalar ro‘yxatini qaytaramiz.

Endi filter funksiyasidan foydalanish misolini ko‘rib chiqamiz. Aytaylik, bizda raqamlar ro‘yxati bor va biz faqat beshdan katta raqamlarni olishni xohlaymiz:

```
def beshdan_kop(son):
return son > 5
sonlar = [2, 7, 1, 8, 4, 5]
natija = filter(beshdan_kop, sonlar)
print(natija)
# [7, 8]
```

Bu erda biz birinchi bo'lib beshdan_ko‘p funksiyani aniqladik, agar unga berilgan argument beshdan katta bo‘lsa, True qiymatini qaytaradi. Keyin biz uni

raqamlar ro‘yxati bilan filtr funktsiyasiga o‘tkazdik.

Shunday qilib, filter funktsiyasi kodni soddalashtirdi va uning o‘qilishi va tushunish darajasini oshirdi.

Bizning trioning so‘nggi funktsiyasi – ma’lumotlarni yig‘ish uchun ishlatiladigan reduce (ular “konvolyutsiya” deyishadi). Aggregatsiya - bu butun ma’lumotlar to‘plamiga bog‘liq bo‘lgan qiymatni hisoblaydigan operatsiya.

Reduse funksiyasidan foydalanib, bitta qiymatni olish uchun ro‘yxat elementlariga amallarni ketma-ket qo‘llashingiz mumkin. Aytaylik, bizda raqamlar ro‘yxati bor va biz ularning yig‘indisini olmoqchimiz. Bunday holda, biz ro‘yxatning har bir elementini ketma-ket qo‘shish uchun for siklidan foydalanishimiz mumkin. Masalan:

```
sonlar = [1, 2, 3, 4, 5]
natija= 0
for son in sonlar:
natija += son
print(natija)
# 15
```

Ushbu misolda biz o‘zgaruvchan natijani e’lon qilamiz va 0 qiymatini beramiz. Keyin raqamlar ro‘yxatining barcha elementlarini for sikli yordamida aylantiramiz. Raqamlar ro‘yxatining har bir elementini natija o‘zgaruvchisining qiymatiga qo'shamiz. Loop tugagandan so‘ng, natija o‘zgaruvchisi raqamlar ro‘yxatining barcha elementlari yig‘indisini saqlaydi.

Keling, ro‘yxatdagi barcha elementlarni ko‘paytirishimiz kerak bo‘lgan misolni ko‘rib chiqaylik. Biz bu muammoni for sikli orqali ham hal qilishimiz mumkin:

```
sonlar = [1, 2, 3, 4, 5]
natija = 1
for son in sonlar:
natija *= son
print(natija)
```

```
# 120
```

Bu yerda bizning kodimiz ro‘yxatning barcha elementlarini qo‘shishga o‘xshaydi. Farqi natija o‘zgaruvchisining boshlang‘ich qiymati va biz bajaradigan operatsiya.

Keling, ushbu ikkita misol uchun kodni yaxshilaymiz. Keling, bitta qiymatni qaytargan holda ro‘yxat elementlariga amalni ketma-ket qo‘llash imkonini beruvchi reduse funksiyasidan foydalanamiz.

Keling, reduse funktsiyasini qo‘llashni ko‘rib chiqaylik:

```
def reduce(func, iterable, initial):
result = initial
for item in iterable:
result = func(result, item)
return result
```

Ushbu misolda reduse funktsiyasiga uchta argument uzatiladi:

- func - biz takrorlanadigan elementlarga qo‘llaydigan funksiya
- iterable — takrorlanuvchi obyekt, uning elementlarini funksiyasi bilan qayta ishlaymiz.
- initial - func chaqirilganda birinchi marta ishlatiladigan boshlang‘ich qiymat

Funksiya ichida biz o‘zgaruvchan natijani yaratamiz, unga boshlang‘ich qiymatni boshlang‘ich qiymat sifatida o‘tkazamiz.

Keyin, for siklidan foydalanib, biz takrorlanadigan ob‘ektning barcha elementlaridan o‘tamiz va har bir elementni natija qiymati bilan birga func funktsiyasiga o‘tkazamiz. func funktsiyasining natijasi natija o‘zgaruvchisiga yoziladi. Reduse funktsiyasining natijasi o‘zgaruvchisining yakuniy qiymati bo‘ladi.

Endi ro‘yxat elementlarining yig‘indisi va mahsulotini topish uchun reduse funksiyasidan foydalanish misolini ko‘rib chiqamiz:

```
def qoshish(x, y):
  return x + y
def kopaytirish(x, y):
```

```
    return x * y
sonlar = [1, 2, 3, 4, 5]
natija = reduce(qoshish, sonlar, 0)
print(natija)
# 15
natija = reduce(kopaytirish, sonlar, 1)
print(result)
# 120
```

Bu yerda biz ikkita sonni qo‘shish va ko‘paytirish uchun ikkita qoshish va kopaytirish funktsiyasini yaratdik. Keyinchalik, raqamlar ro‘yxati elementlariga qo‘shish va ko‘paytirish funktsiyalarini qo‘llash natijasini olish uchun kamaytirish funktsiyasidan foydalanamiz.

Birinchi marta qisqartirish funksiyasini chaqirganimizda, biz boshlang‘ich qiymati 0 ni o‘tkazamiz va ikkinchi qo‘ng‘iroqda biz qiymatini o‘tkazamiz. Shu tarzda biz kodni qisqartirish va o‘qilishi mumkin bo‘ldi.

Biz har qanday mantiqni for siklida amalga oshirishimiz mumkin, ammo bu vosita har doim ham qulay yoki tushunarli emas. Misol uchun, agar biz ro‘yxatning har bir elementini o‘zgartirishimiz kerak bo‘lsa, u holda for siklidan foydalanish kelajakda kodni o‘qish va saqlab turish bilan bog‘liq muammolarni keltirib chiqarishi mumkin.

Bunday holda, biz ro‘yxatning har bir elementiga funktsiyani qo‘llash va o‘gartirilgan qiymatlar bilan yangi ro‘yxatni qaytarish imkonini beruvchi map funktsiyasidan foydalanishimiz mumkin. Shuningdek, elementlarni muayyan shart asosida tanlash uchun for sikli o‘rniga belgilangan shartni qanoatlantiradigan elementlarni qaytaruvchi filter funksiyasidan foydalanishingiz mumkin.

Agar biz ro‘yxatning qiymatlarini jamlashimiz kerak bo‘lsa, masalan, uning barcha elementlarini qo‘shish yoki ko‘paytirish, u holda for siklidan foydalanish shart emas. Bunday holda, berilgan funktsiyani ro‘yxat elementlarining juftlariga ketma-ket qo‘llaydigan va yakuniy natijani qaytaradigan reduse funktsiyasidan foydalanishingiz mumkin.

Loopni har doim map, filter yoki reduse funktsiyalari bilan almashtirib bo‘lmaydi. Ammo oddiy ro‘yxat manipulyatsiyasi haqida gap ketganda, ushbu funktsiyalardan foydalanish kodni o‘qish, yozish va tushunishni ancha osonlashtirishi mumkin. Bu, shuningdek, mantiqdagi xatolardan qochishga yordam beradi va kelajakda kodni yanada o‘qilishi va davom ettirilishiga yordam beradi, chunki har bir funktsiya bitta vazifani bajaradi.

FOYDALANILGAN ADABIYOTLAR

1. 1. A.Nazrullayev Pythonda dasturlash asoslari-Toshkent: “Akademnashr”, 2021.-336b

2. 2. Normurodov Ch.B. Mengliyev Sh.A. PHP7 dasturlash tili - O‘quv qo‘llanma – Termiz: “Xamidi xususiy firmasi”, 2020, 218 bet.

3. 3. Vasilev A. N. Python na primerax. Prakticheskiy kurs po programmirovaniyu. — SPb. Nauka i Texnika, 2016. — 432 bet.

4. SARVINOZ, T. (2023). DESIGN OF THE PREPARATION PROCESS SYSTEM FOR EVALUATION SYSTEMS IN SCHOOLS. International Multidisciplinary Journal for Research & Development, 10(11).

5. Toxirova, S. (2023, November). Python dasturida lug’atlar bilan ishlash. In Conference on Digital Innovation:" Modern Problems and Solutions".

6. Toxirova, S. (2023). MA'LUMOTLAR TUZILMASI VA ALGORITMLAR TUSHUNCHASI. Engineering problems and innovations.

7. Mahmudova, M., & Toxirova, S. (2023, October). MULTISERVISLI TARMOQ XAVFSIZLIGIDA NEYRON TARMOQLARINI O ‘RNI. In Conference on Digital Innovation:" Modern Problems and Solutions".

8. TOJIBOEV, I., RAYIMJONOVA, O., ISKANDAROV, U., MAKHAMMADJONOV, A., & TOKHIROVA, S. МИРОВАЯ НАУКА. МИРОВАЯ НАУКА Учредители: ООО" Институт управления и социально-экономического развития", (3), 26-29.

9. Muhammadjonov, A., & Toxirova, S. (2023). YARIMO ‘TKAZGICHLARNING TURLARI. Ichki va tashqi yarimo ‘tkazgichlar. Research and implementation.

10. Обухов, В. А. Тохирова Сарвиноз Гайратжон кизи, & Исахонов Хушнидбек Муродилжон угли.(2023). ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ТЕКСТА. Ta'lim Innovatsiyasi Va Integratsiyasi, 7 (1), 52–57. ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ТЕКСТА. Ta'lim Innovatsiyasi Va Integratsiyasi, 7(1), 52-57.

11. Обухов, В. А. Тохирова Сарвиноз Гайратжон кизи, & Сотволдиев Асадбек Аброржон угли. (2023). МЕТОДЫ РАСПОЗНАВАНИЯ И ЭТАПЫ ОБРАБОТКИ ИЗОБРАЖЕНИЯ. Ta'lim Innovatsiyasi Va Integratsiyasi, 7 (1), 40–44.

12. Muhammadjonov, A., & TURLARI, T. S. Y. T. ICHKI VA TASHQI YARIMO 'TKAZGICHLAR. Research and implementation.–2023.

13. Porubay, O., & Khasanova, M. (2023). Model of innovative progress in the power sector. Engineering problems and innovations.

14. Порубай О. В., Хасанова М. У. К. Обзор процесса принятия решений в условиях риска и неопределенности //Universum: технические науки. – 2022. – №. 7-1 (100). – С. 17-19.

15. Porubay, O., & Khasanova, M. (2023). Formation of new technologies for innovation management in the modern competitive environment. Engineering problems and innovations.

16. Porubay, O., & Khasanova, M. (2023). Model of innovative progress in the power sector. Engineering problems and innovations.

17. Porubay O., Siddikov I., Madina K. Algorithm for optimizing the mode of electric power systems by active power //2022 International Conference on Information Science and Communications Technologies (ICISCT). – IEEE, 2022. – С. 1-4.

18. Porubay O., Khasanova M. Machine learning as a tool of modern pedagogical technologies //Science and innovation. – 2022. – Т. 1. – №. B3. – С. 840-843.

19. Порубай О. В., Хасанова М. У. Концепция безопасности в теории и практике принятия решений //Просвещение и познание. – 2022. – №. 7 (14). – С. 11-20.

20. Порубай О. В., Хасанова М. СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ С ИНТЕЛЛЕКТУАЛЬНЫМИ МЕХАНИЗМАМИ ПОИСКА ДЛЯ ОПЕРАТИВНОДИСПЕТЧЕРСКОГО УПРАВЛЕНИЯ В ЭЛЕКТРОЭНЕРГЕТИКЕ. – 2021.

21. Tokhirova Sarvinoz Gayratjon Kizi, Sotvoldiyev Asadbek Abrorjanovich, & Isakhanov Khushnidbek Murodiljanovich. (2023). DATA STRUCTURE AND ALGORITHM ANALYSIS PROCESS. Best Journal of Innovation in Science, Research and Development, 2(11), 722–724. Retrieved from https://www.bjisrd.com/index.php/bjisrd/article/view/943

22. Tokhirova Sarvinoz Gayratjon Kizi. (2023). Ethernet and Fast Ethernet network architecture. Best Journal of Innovation in Science, Research and Development, 175–179. Retrieved from https://www.bjisrd.com/index.php/bjisrd/article/view/985

23. МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ И ИХ ПРОИСХОЖДЕНИЕ. (2023). Journal of Technical Research and Development, 1(2), 32-37. https://jtrd.mcdir.me/index.php/jtrd/article/view/80

24. МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ И ИХ ПРОИСХОЖДЕНИЕ. (2023). Journal of Technical Research and Development, 1(2), 32-37. https://jtrd.mcdir.me/index.php/jtrd/article/view/80