**AMALIY MASHG‘ULOT UCHUN O‘QUV MATERIALLARI**

**1-Mavzu:** “Python dasturlash tili” faniga kirish va asosiy tushunchalari.

**9-mashg‘ulot.** Pythonda “Lug‘at” bilan ishlash.

O‘quv savollari:

1. Lug‘atlar va ularning qo‘llanilishi.
2. Lug‘atlarni yaratish usullari.
3. Lug‘atlar bilan ishlovchi metodlar.

**1. Lug‘atlar va ularning qo‘llanilishi.**

**Pythondagi lug‘at (dictionary)**  elementlar to‘plamini saqlaydi. Lug‘at tarkibidagi har bir element o‘ziga xos, yagona kalitga va u bilan bog‘liq bo‘lgan qiymatga ega bo‘ladi**.**

Lug‘at ta’rifi quyidagi sintaksisga ega:

dictionary={kalit1: qiymat1, kalit2: qiymat2, ....}

Lug‘atning barcha elementlari jingalak qavslar ichiga joylanadi. Har bir element bir-biri bilan vergul orqali ajratildi. Har bir elementning yagona kaliti va unga bog‘langan qiymat mavjud bo‘ladi. Elementning kaliti birinchi keladi va undan keyin qiymat keladi. Kalit va qiymatini ikki nuqta ajratib turadi.

**2. Lug‘atlarni yaratish usullari.**

**Lug‘atni aniqlash:**

users = {1: "Tomson", 2: "Bobi", 3: "Billi"}

users lug‘ati raqamlarni kalit sifatida va satrlarni qiymat sifatida ishlatadi. Ya’ni, 1-kalitga ega bo‘lgan element "Tomson" qiymatiga teng, 2-kalitga ega element "Bobi" qiymatiga ega va hokazo.

Lug‘atdagi elementlar tarkibi bir xil tipli ma’lumotlar bo‘lishi ham mumkin. Masalan:

e\_mails = {"tomson@gmail.com": "Tomson", "bobi@gmail.com": "Bobi", "semmi@gmail.com": "Semmi"}

e\_mails lug‘atida kalit sifatida satrlardan foydalanildi - foydalanuvchi elektron pochta manzillari va qiymat sifatida ham satrlardan - foydalanuvchi nomlaridan foydalanildi.

Lekin kalitlar va satrlar bir xil turdagi bo‘lishi shart emas. Ular turli xil turlarni ko‘rsatishi mumkin:

objects = {1: "Tomson", "2": False, 3: 150.64}

Bundan tashqari, umuman elementsiz, bo‘sh lug‘atni aniqlash ham mumkin:

dict\_object = {}

yoki shunga o‘xshash:

dict\_object = dict()

**Ro‘yxatlar va kortejlarni lug‘atga aylantirish**

Lug‘at va ro‘yxat tuzilmaviy jihatdan bir-biriga o‘xshamaydigan turlar bo‘lsa-da, standart **dict()** funksiyasi yordamida ma’lum turdagi ro‘yxatlarni lug‘atga aylantirish mumkin. Buning uchun ro‘yxat ichki ro‘yxatlar to‘plamini saqlashi kerak. Har bir ichki ro‘yxat ikkita elementdan iborat bo‘lishi kerak - lug‘atga aylantirilganda birinchi element kalitga, ikkinchisi esa qiymatga aylanadi:

user\_list=[

    ["+554513455", "Tomson"],

    ["+002549557", "Bobi"],

    ["+445454512", "Elison"]

]

user\_dict = dict(user\_list)

print(user\_dict)  *# {"+554513455", "Tomson:", "+002549557", #"Bobi", "+445454512": "Elison"}*

Xuddi shunday, ikkita kortej lug‘atga aylantirilishi mumkin, o‘z navbatida lug‘at ikkita kortejni o‘z ichiga oladi:

user\_tuple = (

    ("+554513455", "Tomson"),

    ("+002549557", "Bobi"),

    ("+445454512", "Elison")

)

user\_dict = **dict** **(user\_tuple)**

print (user\_dict)

**Lug‘at elementlariga murojaat qilish va o‘zgartirish**

Lug‘at elementlariga murojaat qilish uchun uning nomidan keyin element kaliti kvadrat qavs ichida ko‘rsatiladi:

Dictionary\_name[key]

Lug‘atdagi elementlarni olish va o‘zgartirish uchun:

dict\_users = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

# "+11002255" kaliti bilan elementni olish

print (dict\_users ["+11002255"])       # Tomson

# element qiymatini "+15245999" tugmasi bilan belgilash

dict\_users["+15245999"] = "Bob Smit"

print (dict\_users["+15245999"])    # "Bob Smit"

Agar bunday kalit bilan elementning qiymatini belgilashda lug‘atda hech qanday element bo‘lmasa, u qo‘shiladi:

 dict\_users ["+5555555"] = "Semmi"

Ammo agar lug‘atda mavjud bo‘lmagan kalit bilan qiymat olishga harakat qilinsa, Python KeyError xatoligini chiqaradi:

foydalanuvchi=foydalanuvchilar["+4444444"] # KeyError

Bunday holatni oldini olish maqsadida elementga murojaat qilishdan oldin "key in dict" ifodasi yordamida lug‘atda element mavjudligini tekshirish mumkin. Agar kalit lug‘atda bo‘lsa, u holda bu ifoda "True" ni qaytaradi:

key = "+00000044"

**if** key **in** dict\_users:

    dict\_user = dict\_users[key]

    print(dict\_user)

**else**:

    print ("Element topilmadi")

Shuningdek lug‘at elementlariga murojaat qilish uchun ***get*** metodidan ham foydalaniladi.

**3. Lug‘atlar bilan ishlovchi metodlar**

***get(key):*** lug‘atdan kalit tugmasi bo‘lgan elementni qaytaradi. Agar ushbu kalitga ega bo‘lgan element mavjud bo‘lmasa, u holda "False" qiymati qaytariladi;

***get(key, default):*** lug‘atdan key kaliti bo‘lgan elementni qaytaradi. Agar bunday kalitga ega element bo‘lmasa, u ***default*** qiymatni qaytaradi.

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

user1 = users.get("+15415154")

print (user1)     # Alison

user2=users.**get**("+15245999","Noma’lum foydalanuvchi")

print (user2 )     #Bobi

user3=users.**get**("+0000000", "Noma’lum foydalanuvchi")

print(user3)     # Noma’lum foydalanuvchi

**DEL funksiyasi**

**Del** funksiyasi elementni kalit bilan olib tashlash uchun ishlatiladi :

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

**del** foydalanuvchilar["+15415154"]

print (foydalanuvchilar)

*# { "+11002255": "Tomson", "+15245999": "Bobi"}*

Ammo shuni yodda tutish kerakki, agar bunday kalit lug‘atda bo‘lmasa, KeyError xatoligi qaytariladi. Shuning uchun, o‘chirilishi kerak bo‘lgan elementni oldin, berilgan kalitli element mavjudligini tekshirish tavsiya etiladi.

dict\_user = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

key = "+99990000"

**if** key **in** dict\_user:

**del** dict\_user[key]

    print(f"{key} kaliti element olib tashlandi")

**else**:

    print (f"{key} kalitli element topilmadi")

**POP() metodi**

O‘chirishning yana bir usuli - **pop**() usuli. Pop metodidan ikki xil ko‘rinishda foydalanish mumkin:

1. ***pop(key):*** ***key*** kalit berilgan elementni olib tashlaydi va olib tashlangan elementni qaytaradi. Agar berilgan kalit bilan element topilmasa, KeyError xatoligi qaytariladi.
2. ***pop(key, default):***  ***key*** kalitli elementni o‘chiradi va olib tashlangan elementni qaytaradi. Agar berilgan kalit bilan element bo‘lmasa, ***default*** qiymati qaytariladi.

foydalanuvchilar = {

    "+11002255": "Tomson",

    "+15245999": "Bobi",

    "+15415154": "Elison"

}

key = "+11002255"

user1 = users.pop (key)

print (user1)      # Tomson

user1 = users.pop("+00000000", "Noma’lum foydalanuvchi")

print (user1)      # Noma’lum foydalanuvchi

Agar barcha elementlarni olib tashlash kerak bo‘lsa, ya’ni bir so‘z bilan aytganda tozalash kerak bo‘lsa, unda **clear()** metodidan foydalanish kerak bo‘ladi:

per.clear()

**Lug‘atlarni nusxalash va birlashtirish**

**copy()** usuli lug‘at tarkibini nusxalab, yangi lug‘atni qaytaradi:

foydalanuvchilar={"998888":"Tomson","998771": "Bobi"}

talabalar = users.copy()

print (talabalar) # {"998888": "Tomson", "998771": "Bobi"}

**update()** metodi ikkita lug‘atni birlashtiradi:

foydalanuvchilar={"998888":"Tomson","998771": "Bobi"}

users2 = {"222222": "Semmi", "666000": "Kail"}

users.update(users2)

print(users)    *# {"998888": "Tomson", "998771": "Bobi", "222222": "Semmi", "666000": "Kail"}*

print(users2)    *# {"222222": "Semmi", "666000": "Kail"}*

Bunday holda, users2 lug‘ati o‘zgarishsiz qoladi. Faqat foydalanuvchilarning lug‘ati o‘zgartiriladi, unga boshqa lug‘at elementlari qo‘shiladi. Ammo agar ikkala manba lug‘ati ham o‘zgarishsiz qolishi kerak bo‘lsa va birlashmaning natijasi uchinchi lug‘at bo‘lsa, avval bitta lug‘atni boshqasiga nusxalanishi kerak:

users3 = users.**copy**()

users3.**update**(users2)

**Lug‘atni takrorlash**

Lug‘at tarkibini sikldan foydalanib olish uchun for siklidan foydalanish mumkin:

users = {"998888": "Tomson", "998771": "Bobi"}

**for** a **in** users:

    print(f"id: {a}   Foydalanuvchi: {users[a]} ")

Takrorlash orqali elementlarga murojaat qilishda joriy elementning kalitini elementning o‘zini olish uchun ishlatiladi.

Elementlarni olishning yana bir usuli - **items()** metodidan foydalanish**:**

users = {"998888": "Tomson", "998771": "Bobi"}

**for** **key**, **val** **in** users.**items**():

    print(f"id: {**key**}   Foydalanuvchi: {**val**} ")

**items()** usuli kortejlar to‘plamini qaytaradi. Har bir kortej elementning kaliti va qiymatini o‘z ichiga oladi, ularga murojaat qilishda ***key*** va ***val*** o‘zgaruvchilarini chaqirilsa yetarli.

Shuningdek, kalitlar ustida takrorlash va qiymatlarni takrorlash uchun alohida usul ham mavjud. Lug‘atdagikalitlarni olish uchun **keys()** metodidan foydalaniladi:

**for key in dict\_users.keys():**

**print(key)**

Faqat qiymatlarni takrorlash uchun lug‘atning *values()* metodidan foydalanladi*:*

**for val in dict\_users.values():**

**print(val)**

**Murakkab lug‘atlar**

Raqamlar va satrlar kabi eng oddiy ob’ektlardan tashqari, lug‘atlar murakkabroq ob’ektlarni ham saqlashi mumkin. Masalan bir xil ro‘yxatlar, kortejlar yoki boshqa lug‘atlarni o‘z ichiga element sifatida joylashtirishi mumkin:

dict\_users = {

    Tomson: {

        "id": "745",

        "e\_main": "tomson12@gmail.com"

    },

    Bobi: {

        "id": "444",

        "e\_mail": "bobi@gmail.com",

        "skype": "bob123"

    }

}

Bunda lug‘atning har bir elementining qiymati o‘z navbatida alohida lug‘atni ifodalashi mumkin.

Ichki lug‘at elementlariga kirish uchun mos ravishda ikkita kalitdan foydalanish kerak:

old\_e\_mail = dict\_users["Tomson"]["e\_mail"]

dict\_users["Tomson"]["e\_mail"] = "supertomson@gmail.com"

**print** (dict\_users["Tomson"])

# { "id": "745", "e\_main": "supertomson@gmail.com }

Ammo agar lug‘atda bo‘lmagan kalit orqali qiymat olishga harakat qilinsa, Python ***KeyError*** xatoligini chiqaradi:

tom\_skype = dict\_users["Tom"]["skype"]    # KeyError

Xatolikka yo‘l qo‘ymaslik uchun lug‘atda kalit mavjudligini tekshirish kerak:

key = "skype"

**if** key **in** dict\_users["Tomson"]:

**print**(dict\_users["Tomson"]["skype"])

**else**:

**print**("skype lug‘atdan topilmadi!")

Boshqa barcha jihatlarda murakkab va ichki lug‘atlar bilan ishlash oddiy lug‘atlar bilan ishlash kabi bajariladi.

**Nazorat savollari:**

1. Pythonda tug‘at tushunchasiga tarif bering.
2. Pythonda lug‘atlar qanday usullar bilan yaratilishi mumkin?
3. Ro‘yxatlar va kortejlarni lug‘atga aylantirish qanday amalga oshiriladi?
4. Lug‘at Elementlariga murojaat qilish va o‘zgartirish qanday amalga oshiriladi?
5. **get()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
6. **del** metodining vazifasi va ishlash prinsipi haqida gapirib bering.
7. **pop()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
8. **clear()** metodining vazifasi va ishlash prinsipi haqida gapirib bering.
9. **copy()** metodining vazifasi qanday va uning qabul qiluvchi parametrlari qanday?
10. **upgrate()** metodining vazifasi va ishlash prinsipi haqida gapirib bering
11. **items()** metodining vazifasi va ishlash prinsipi haqida gapirib bering
12. **keys ()** metodining vazifasi va ishlash prinsipi haqida gapirib bering
13. Murakkab lug‘atlar qanday hosil qilinadi?