PYTHON — 2-BO‘LIM

MA’LUMOT TUZILMALARI (PYTHONIC)

1. Nazariy qism (chuqur)

---

1. Pythonik yondashuv nima?

Pythonik — bu:

- kam kod
- o‘qilishi oson
- mantiq aniq
- tayyor imkoniyatlardan to‘g‘ri foydalanish

Olimpiadada Pythonic yozuv:

- xatoni kamaytiradi
- tezlikni oshiradi
- kodni qisqartiradi

2. `list` — asosiy ishchi tuzilma

Ta’rif

`list` — tartibli, o‘zgaruvchan (mutable) ketma-ketlik.

```
a = [1, 2, 3]
```

Xususiyatlari:

- indekslanadi (`a[0]`)
- o‘lchami dinamik
- turli tiplarni saqlashi mumkin

Asosiy amallar:

```
a.append(5)
a.pop()
a.insert(1, 10)
len(a)
```

Olimpiadada:

- massiv o‘rnida doim `list`
- tez yurish → `for x in a`

3. `tuple` — o‘zgarmas ketma-ketlik

```
t = (1, 2, 3)
```

Qachon ishlatiladi?

- qiymatlar o‘zgarmas bo‘lsa
- `dict` kaliti sifatida

`tuple`:

- `list` ga qaraganda tezroq
- xavfsizroq

4. `string` — satr ham tuzilma

```
s = "python"
```

- indekslanadi
- kesiladi: `s[1:4]`
- immutable (o‘zgarmas)

Pythonda satrlar bilan ishlash juda kuchli:

```
s[::-1]
s.split()
s.isdigit()
```

5. `set` — noyob elementlar to‘plami

```
s = {1, 2, 3}
```

Xususiyatlari:

- takror yo‘q
- tartibsiz
- juda tez (`O(1)` o‘rtacha)

Asosiy amallar:

```
s.add(x)
s.remove(x)
x in s
```

Qachon ishlatiladi?

takrorlarni olib tashlash

mavjudlikni tez tekshirish

6. `dict` — kalit–qiymat tuzilmasi

```
d = {"a": 1, "b": 2}
```

Xususiyatlari:

- juda tez qidirish
- kalitlar noyob
- qiymatlar ixtiyoriy

Asosiy ishlatish:

```
d[x] = d.get(x, 0) + 1
```

Olimpiadada:

- chastota hisoblash
- indekslash
- moslik saqlash

7. `enumerate` — indeks + qiymat

```
for i, x in enumerate(a):
    print(i, x)
```

`range(len(a))` o‘rniga toza va xavfsiz.

---

8. `zip` — bir nechta tuzilmani bog‘lash

```
for x, y in zip(a, b):
    print(x, y)
```

Juftliklar bilan ishlashda juda qulay.

9. List comprehension — Python kuchi

```
b = [x * 2 for x in a if x % 2 == 0]
```

Bu:

- tezroq
- qisqaroq
- tushunarliroq

Lekin: murakkab mantiqda ishlatmaslik kerak

10. `collections` moduliga kirish

`deque` — ikki tomondan tez ishlov

```
from collections import deque
dq = deque()
```

- `append`, `appendleft`
- `pop`, `popleft`

BFS, sliding window uchun asos.

`Counter` — chastota hisoblash

```
from collections import Counter
cnt = Counter(a)
```

`dict` dan ham qulay.

11. Nusxa olish (copy) muammosi

```
b = a        # XATO (bir xil obyekt)
b = a[:]     # to‘g‘ri
```

Juda ko‘p yashirin xato shu yerda.

12. `in` operatori — tez tekshiruv

```
if x in s:
```

- `list` → sekin (`O(n)`)
- `set/dict` → tez (`O(1)`)

To‘g‘ri tuzilma tanlash — tezlik garovi.

13. Python’da tezlik va xotira muvozanati

Tuzilma	Tezlik	Tartib	Takror
list	o‘rtacha	bor	bor
tuple	tez	bor	bor
set	juda tez	yo‘q	yo‘q
dict	juda tez	bor (3.7+)	kalit yo‘q

2. METODIK YORDAM

1. “Masala → tuzilma” tamoyilini qat’iy singdirish

Boshlovchilar ko‘pincha:

“Qaysi tuzilmani bilaman, shuni ishlataman”
deydi — xato.

To‘g‘ri yondashuv:
Masala nimani so‘raydi?
Tezlik qayerda muhim?
Qaysi tuzilma eng mos?

Misollar:

- mavjudlikni tekshirish → `set`
- chastota → `dict` / `Counter`
- tartibli yurish → `list`

2. `list` ni “default” tuzilma sifatida berish

Metodik qoida:

Agar aniq sabab bo‘lmasa — `list` ishlatiladi.

Tavsiya:

- massiv so‘zini ishlatmaslik
- har doim `list` deb atash

3. `dict` bilan chastota hisoblashni avtomatizmga aylantirish

Standart yozuv:

```
d[x] = d.get(x, 0) + 1
```

Metodik mashq:

- buni 10 marta turli masalalarda ishlatish
- `if x in d` bilan solishtirish

Maqsad: refleks darajasiga chiqish.

4. `set` ni tezlik vositasi sifatida ko‘rsatish

Ko‘p o‘quvchi `set` ni faqat:

“takrorlarni olib tashlash”
deb biladi.

Aslida:

- mavjudlik → O(1)
- kesishma → juda tez

Metodik misol: `x in list` vs `x in set`

5. `enumerate` va `zip` ni majburiy odatga aylantirish

O‘rniga:

```
for i in range(len(a)):
```

Yaxshisi:

```
for i, x in enumerate(a):
```

Xatolar kamayadi, kod tozalanadi.

6. List comprehension’ni “o‘ylab ishlatish”

Qoidasi:

- 1 qatorda oddiy mantiq → mumkin
- murakkab shart → oddiy `for`

Metodik yondashuv: har bir comprehension’ni oddiy siklga ochib ko‘rsatish

7. `collections` ni erta berish (lekin cheklangan)

Boshlang‘ich darajada:

- `deque`
- `Counter`

Yetarli.

`defaultdict` va boshqalar — keyinroq.

8. Nusxa olish muammosini alohida ko‘rsatish

Ko‘p yashirin xato shu yerda:

```
b = a
```

Metodik mashq:

- `id(a)` va `id(b)` ni chiqarish
- farqni ko‘z bilan ko‘rsatish

9. `in` operatori murakkabligini tushuntirish

Ko‘pchilik bilmaydi:

- `x in list` → O(n)
- `x in set` → O(1)

Bu tushuncha algoritmik fikrlashning boshlanishi.

10. Pythonic kod = o‘qiladigan kod

Metodik tamoyil:

Eng qisqa kod emas,
eng tushunarli va toza kod — eng yaxshi kod.

3. TIPIK XATOLAR

1. `list` va `tuple` ni chalkashtirish

```
t = (1, 2, 3)
t[0] = 5   # XATO
```

2. `dict` da mavjud bo‘lmagan kalitni tekshirmasdan o‘qish

```
print(d[x])  # KeyError
```

To‘g‘risi:

```
d.get(x, 0)
```

3. `set` da indeks bilan murojaat qilish

```
s[0]  # XATO
```

4. Nusxa o‘rniga havola olish

```
b = a
```

Buni tushunmaslik — juda ko‘p xato keltiradi.

5. `list.remove()` va `pop()` ni chalkashtirish

- `remove(x)` → qiymat bo‘yicha
- `pop(i)` → indeks bo‘yicha

6. List comprehension’da murakkab mantiq

```
[x for x in a if x > 0 if x % 2 == 0]  # yomon
```

7. `Counter` ni `dict` deb o‘ylash

`Counter`:

- yo‘q kalit → 0
- `dict`:
- yo‘q kalit → xato

8. `zip` uzunliklarini unutish

`zip(a, b)` → eng qisqa bo‘yicha ishlaydi.

9. `in` operatorini noto‘g‘ri tuzilmada ishlatish

```
if x in a:   # a katta list → sekin
```

10. O‘qilmaydigan Pythonic yozuv

Haddan ortiq qisqartirish → tushunarsiz kod.

METODIK XULOSA (2-BO‘LIM)

Agar o‘quvchi:

- masala → tuzilma → kod
- tezlik → tuzilma tanlash
- tozaligi → Pythonic uslub

ketma-ketligini o‘zlashtirsa, Python haqiqiy qurolga aylanadi.