PYTHON

1-BO‘LIM

PYTHON ASOSLARI VA SINTAKSIS

1. Nazariy qism (chuqur)

1. Python nima?

Python — bu:

- interpretatsiya qilinadigan
- yuqori darajadagi
- o‘qilishi juda oson
  dasturlash tili.

Olimpiadada Python:

- tez yoziladi
- kod qisqa
- mantiqga e’tibor kuchli

📌 Kamchiligi: C++ ga qaraganda sekinroq
shuning uchun to‘g‘ri algoritm shart

2. Python dastur qanday ishlaydi?

Python’da:

- kod qatorma-qator bajariladi
- oldindan kompilyatsiya yo‘q

Misol:

```
print("Salom, Python!")
```

Muhim farq:

- C++ → `{}` va `;`
- Python → chekinish (indentation)

3. Indentation — PYTHON YURAGI

Python’da bloklar bo‘sh joy (4 probel) bilan belgilanadi.

To‘g‘ri:

```
if x > 0:
    print("Musbat")
```

Noto‘g‘ri:

```
if x > 0:
print("Musbat")  # XATO
```

Indentation xatosi → Runtime error

4. Chiqarish: `print()`

```
print("Natija")
print(a, b, c)
```

- avtomatik yangi qatordan chiqaradi
- bir nechta qiymatni chiqaradi

Formatlash:

```
print(f"Yig‘indi = {a}")
```

5. Kiritish: `input()`

```
x = input()
```

`input()` har doim satr (`str`) qaytaradi

To‘g‘ri ishlatish:

```
x = int(input())
y = float(input())
```

Bir qatorda:

```
a, b = map(int, input().split())
```

Olimpiadada eng ko‘p ishlatiladigan yozuv

6. O‘zgaruvchilar va ma’lumot turlari

Python’da tur oldindan yozilmaydi.

```
a = 5        # int
b = 3.14     # float
c = "abc"    # str
d = True     # bool
```

Dinamik tiplash — qulay, lekin ehtiyot bo‘lish kerak.

7. Arifmetik amallar

Amal	Belgisi
Qo‘shish	+
Ayirish	-
Ko‘paytirish	*

Amal	Belgisi
Bo‘lish	/
Butun bo‘lish	//
Qoldiq	%
Daraja	**

Misol:

```
7 // 2  # 3
7 / 2   # 3.5
```

8. Taqqoslash amallari

Amal	Ma’nosi
==	teng
!=	teng emas
<	kichik
>	katta
<=	kichik yoki teng
>=	katta yoki teng

Natija: `True` yoki `False`

9. Mantiqiy amallar

Amal	Belgisi
VA	`and`
YOKI	`or`
EMAS	`not`

Misol:

```
if a > 0 and b > 0:
    print("Ikkalasi musbat")
```

10. Shart operatorlari

```
if
```

```
if x > 0:
    print("Musbat")
```

```
if – else
```

```
if x % 2 == 0:
    print("Juft")
else:
    print("Toq")
```

`if – elif – else`

```
if x > 0:
    print("Musbat")
elif x < 0:
    print("Manfiy")
else:
    print("Nol")
```

`elif` — `else if` ning Python’dagi ko‘rinishi.

11. Sikllar

`for` sikli

```
for i in range(5):
    print(i)
```

`range(a, b)` → `a` dan `b-1` gacha

`while` sikli

```
while x > 0:
    x -= 1
```

`while` — shart asosida ishlaydi.

12. `break` va `continue`

- `break` → siklni to‘liq to‘xtatadi
- `continue` → joriy aylanishni o‘tkazadi

13. Python’da satrlar (`str`)

```
s = "python"
```

- indekslanadi: `s[0] == 'p'`
- uzunligi: `len(s)`
- bo‘lish: `s.split()`

14. Python’da tezlik masalasi (muhim!)

Olimpiadada:

```
import sys
input = sys.stdin.readline
```

Katta kirishlarda juda muhim.

1-BO‘LIM NATIJASI

Agar shu nazariya puxta o‘zlashtirilsa, o‘quvchi:

- Python sintaksisini xatosiz yozadi
- oddiy va o‘rta masalalarni tez kodlaydi
- 2-bo‘limga (Ma’lumot tuzilmalari) tayyor holatga keladi

METODIK YORDAM

1. Pythonni “kod yodlash” emas, “mantiq yozish” sifatida berish

Pythonning asosiy ustunligi — o‘qilishi.

Metodik yondashuv:

- Avval masalani oddiy tilda tushuntirish
- Keyin bir satr Python bilan ifodalash

O‘quvchi kodni emas, fikrni yozishni o‘rganadi.

2. Indentationni qattiq nazorat qilish

Dastlabki bosqichda:

- 4 probel qoidasi
- `Tab` va probel aralashmaslik

Tavsiya:

- IDE’da show whitespace yoqish
- Har blokni vizual tekshirish

Bu Python’dagi eng xavfli xato manbai.

3. `input()` har doim `str` ekanini doim takrorlash

Boshlovchilarning 80% xatosi shu yerda.

Metodik mashq:

```
x = input()
print(type(x))
```

Keyin:

```
x = int(input())
```

“O‘qish → o‘zgartirish → ishlatish” zanjiri mustahkamlanadi.

4. Bir qatorda kiritishni erta o‘rgatish

Olimpiada standart yozuvi:

```
a, b = map(int, input().split())
```

Metodik foyda:

- kod qisqaradi
- xato kamayadi
- tezlik oshadi

5. `for` va `while` ni aniq chegaralash

Asosiy savol:

- Necha marta? → `for`
- Qachongacha? → `while`

Mashq:

- bir masalani ikkala sikl bilan yozdirish
- farqini tahlil qilish

6. `range()` ni to‘liq tushuntirish

Ko‘p xato qilinadigan joy:

```
range(1, 5)  # 1,2,3,4
```

Metodik usul:

- sonlar chizig‘ida ko‘rsatish
- `b` hech qachon kirmasligini ta’kidlash

7. `if–elif–else` ni mantiqiy daraxt sifatida berish

`elif` — bu:

“agar bu bo‘lmasa, keyingisini tekshir”

Mashq:

- bir nechta mustaqil `if`
- keyin `elif` bilan qayta yozish

8. Python’da `bool` ni yashirin ishlatish

Python’da:

```
if x:
```

degani:

```
if x != 0:
```

Metodik tavsiya:

- boshida faqat aniq shart yozdirish
- keyin qisqa yozuvga o‘tish

9. Formatlangan chiqarishni erta berish

```
print(f"{a} + {b} = {a + b}")
```

Foydasi:

- tushunarli chiqish
- xatoni tez topish

10. Tez kiritish–chiqarishni vaqtida berish

Olimpiada yondashuvi:

```
import sys
input = sys.stdin.readline
```

Buni oddiy masalalarda emas,
katta kirishlarda berish kerak.

3. TIPIK XATOLAR

1. Indentation xatosi

```
if x > 0:
print(x)   # XATO
```

Python ishlamaydi.

2. `input()`ni son deb o‘ylash

```
x = input()
print(x + 1)  # XATO
```

To‘g‘risi:

```
x = int(input())
```

3. = va == ni chalkashtirish

```
if x = 5:   # XATO
```

Python’da bu SyntaxError.

4. `range()` chegarasini noto‘g‘ri tushunish

```
for i in range(1, 5):
    pass
```

5 kirmaydi.

5. `while` da cheksiz sikl

```
while x > 0:
    print(x)
```

`x` o‘zgarmayapti.

6. `elif` o‘rniga alohida `if`

```
if x > 0:
    ...
if x < 0:
    ...
```

Ba’zi holatlarda mantiq buziladi.

7. `and / or` ustuvorligini unutish

```
if a > 0 and b > 0 or c > 0:
```

To‘g‘risi:

```
if (a > 0 and b > 0) or c > 0:
```

8. `break` va `continue` ni chalkashtirish

- `break` → sikldan chiqadi
- `continue` → shu aylanishni tashlab o‘tadi

9. `print` ichida vergulni noto‘g‘ri ishlatish

```
print(a, + b)  # keraksiz
```

10. Tezlik muammosini e’tiborsiz qoldirish

Katta kirish + oddiy `input()` → TLE.

METODIK XULOSA (1-BO‘LIM)

Agar:

- indentation → qat’iy
- kiritish → ongli
- shart/sikl → mantiq asosida

o‘rgatilsa, Python eng qulay va kuchli qurolga aylanadi.