به نام خدا

مقدمات پایتون

متغیر چیست؟

متغیر (Variable) محفظه‌ای است برای نگهداری داده‌ها. در پایتون، نیازی به تعیین نوع متغیر نیست و نوع آن به صورت خودکار با توجه به مقدار تعیین می‌شود.

In [1]:
```
x = 5        # عدد صحیح (int)
pi = 3.14    # عدد اعشاری (float)
name = 'Ali' # رشته (string)
flag = True  # مقدار بولی (bool)

print(x, pi, name, flag)
```

```
5 3.14 Ali True
```

بررسی نوع داده

برای بررسی نوع یک متغیر می‌توان از تابع `type` استفاده کرد:

In [2]:
```
print(type(x))
print(type(pi))
print(type(name))
print(type(flag))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

انواع داده عددی در پایتون

In [1]:
```
a = 10       # int
b = 3.5      # float

print(type(a), type(b))
```

```
<class 'int'> <class 'float'>
```

رشته‌ها (Strings)

رشته‌ها مجموعه‌ای از کاراکترها هستند که در داخل کوتیشن ('' یا "") قرار می‌گیرند.

In [4]:
```
text = "Hello"
print(text)
print(text.upper())
print(text.lower())
print(text[0])
```

```
Hello
HELLO
hello
H
```

مقادیر بولی (Boolean)

نوع `bool` دو مقدار دارد: `True` و `False`. این نوع در تصمیم‌گیری‌ها و شرط‌ها استفاده می‌شود.

In [5]:
```
a = 10
b = 5
print(a > b)    # True
print(a == b)   # False
```

```
True
False
```

ورودی‌ها و خروجی‌ها در پایتون

در برنامه‌نویسی، یکی از مهم‌ترین بخش‌ها **ورودی و خروجی** است. ما باید بتوانیم از کاربر داده دریافت کنیم و نتیجه را به او نمایش دهیم. در پایتون چند روش ساده برای انجام این کار وجود دارد.

۱. تابع `input()`

تابع **input()** برای دریافت داده از کاربر استفاده می‌شود. هر چیزی که کاربر وارد کند به صورت **رشته (string)** دریافت می‌شود.

In [ ]:
```
name = input("لطفاً نام خود را وارد کنید: ")
print("سلام،", name)
```

```
لطفاً نام خود را وارد کنید: حسین
سلام، حسین
```

اگر بخواهیم عدد دریافت کنیم، باید رشته را به نوع عددی تبدیل کنیم:

In [ ]:
```
age = int(input("سن خود را وارد کنید: "))
print("شما", age, "سال دارید.")
```

```
سن خود را وارد کنید: 44
شما 44 سال دارید.
```

۲. تابع `print()`

تابع **print()** برای نمایش خروجی در کنسول استفاده می‌شود. می‌توان چند مقدار را با کاما جدا کرد یا متن و متغیر را با هم چاپ کرد.

In [ ]:
```
a = 5
b = 10
print("جمع a و b برابر است با:", a + b)
```

```
جمع a و b برابر است با: 15
```

۳. f-string

از نسخه‌ی پایتون ۳.۶ به بعد، می‌توانیم از **f-string** برای ترکیب متن و متغیرها به صورت ساده و خوانا استفاده کنیم.

In [ ]:
```
name = "علی"
age = 25
print(f"سلام {name}! شما {age} ساله هستید.")
```

```
سلام علی! شما 25 ساله هستید.
```

کنترل جریان (Control Flow)

در برنامه‌نویسی، **کنترل جریان** به تصمیم‌گیری و تکرار در اجرای دستورات گفته می‌شود. با استفاده از شرط‌ها، حلقه‌ها و دستورات **break** و **continue** می‌توان مسیر اجرای برنامه را کنترل کرد.

۱. شرط‌ها (if, elif, else)

با دستور **if** می‌توانیم بر اساس یک شرط تصمیم بگیریم که بخش خاصی از کد اجرا شود. از **elif** و **else** برای بررسی شرایط دیگر یا حالت پیش‌فرض استفاده می‌کنیم.

In [ ]:
```
x = 15

if x > 20:
    print("x is greater than 20")
elif x == 15:
    print("x is exactly 15")
else:
    print("x is less than 15")
```

```
x is exactly 15
```

۲. حلقه‌ها (Loops)

در پایتون دو نوع حلقه داریم: **for** و **while**. - حلقه **for** برای تکرار روی یک مجموعه (لیست، رشته، یا رنج عددی) استفاده می‌شود. - حلقه **while** تا زمانی که شرط برقرار است، اجرا می‌شود.

In [ ]:
```
# for loop example
for i in range(5):
    print("Iteration:", i)
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
```

In [ ]:
```python
# while loop example
i = 0
while i < 5:
    print("Iteration:", i)
    i += 1
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
```

Nested Loops

In [ ]:
```python
for i in [1, 2]:
    for j in [10, 20, 30]:
        print(i, j)
```

```
1 10
1 20
1 30
2 10
2 20
2 30
```

۳. دستور `break`

با دستور **break** می‌توان اجرای حلقه را قبل از اتمام کامل آن متوقف کرد.

In [ ]:
```python
for i in range(10):
    if i == 5:
        break
    print("Value:", i)
```

```
Value: 0
Value: 1
Value: 2
Value: 3
Value: 4
```

۴. دستور `continue`

با دستور **continue** می‌توان اجرای حلقه را در همان تکرار متوقف کرد و به تکرار بعدی رفت.

In [ ]:
```python
for i in range(5):
    if i == 2:
        continue
    print("Number:", i)
```

```
Number: 0
Number: 1
Number: 3
Number: 4
```

مثال:

برنامه ای بنویسید که مثلث های زیر را چاپ کند:

```
* ** *** **** *****
```

In [5]:
```python
for i in range(1, 6):
    for j in range(1, i+1):
        print("*", end="")
    print()
```

```
*
**
***
****
*****
```