بخش 2: مفاهیم پایه در پایتون

1. متغیرها و انواع داده‌ها

در پایتون، متغیرها برای ذخیره‌سازی مقادیر مختلف استفاده می‌شوند. همچنین، پایتون دارای انواع داده‌ای مختلفی است که هرکدام ویژگی‌ها و کاربردهای خاص خود را دارند.

تعریف متغیرها

در پایتون، متغیرها به راحتی با انتساب مقداری به نام متغیر ایجاد می‌شوند. شما نیاز به اعلام نوع متغیر ندارید، زیرا پایتون به صورت خودکار نوع داده‌ها را شناسایی می‌کند.

```
x = 5      # عدد صحیح
name = "Alice"  # رشته
```

انواع داده‌های پایه:

1. عدد صحیح (Integer):

   اعداد بدون اعشار هستند. به عنوان مثال، ۵، -۳، ۱۰۲۴.

   ```
   age = 25
   ```

2. عدد اعشاری (Float):

   اعداد با اعشار. به عنوان مثال، 3.14، -2.5.

   ```
   pi = 3.14
   ```

3. رشته‌ها (String):

   مجموعه‌ای از کاراکترها که داخل کوتیشن‌ها (تک یا دوتایی) قرار می‌گیرند. به عنوان مثال، "hello"، "Python".

   ```
   greeting = "Hello, World!"
   ```

4. بولین‌ها (Boolean):

   این نوع داده دو مقدار ممکن دارد:

   ```
   True
   ```

   یا

   ```
   False
   ```

   .

   ```
   is_valid = True
   ```

تبدیل نوع داده‌ها (Type Casting)

گاهی اوقات نیاز دارید که نوع داده‌ها را از یک نوع به نوع دیگر تبدیل کنید. این کار با استفاده از توابع تبدیل مانند `str`, `float()`, `int()` انجام می‌شود.

- تبدیل به عدد صحیح:

```
x = int(3.14)  # تبدیل عدد اعشاری به عدد صحیح
```

- تبدیل به رشته:

```
x = str(100)  # تبدیل عدد صحیح به رشته
```

- تبدیل به عدد اعشاری:

```
x = float("3.14")  # تبدیل رشته به عدد اعشاری
```

2. عملگرها (Operators)

عملگرها برای انجام عملیات مختلف روی داده‌ها و متغیرها استفاده می‌شوند. در پایتون، انواع مختلفی از عملگرها وجود دارند.

عملگرهای ریاضی:

این عملگرها برای انجام عملیات ریاضی مانند جمع، تفریق، ضرب و تقسیم استفاده می‌شوند:

- **+** : جمع
- **-** : تفریق
- ***** : ضرب
- **/** : تقسیم
- **%** : باقیمانده تقسیم (Modulus)
- **//** : تقسیم صحیح (Division)
- ****** : توان

```
a = 5 + 3  # جمع
b = 7 - 2  # تفریق
c = 4 * 3  # ضرب
d = 10 / 2  # تقسیم
e = 10 % 3  # باقیمانده
f = 2 ** 3  # توان
g = 10 // 3 # تقسیم صحیح
```

عملگرهای مقایسه‌ای:

این عملگرها برای مقایسه دو مقدار استفاده می‌شوند:

- **==** : برابر بودن
- **!=** : نابرابر بودن
- **>** : بزرگتر بودن

- < : کوچکتر بودن
- =< : بزرگتر یا برابر بودن
- =>

: کوچکتر یا برابر بودن

```
x = 5
y = 3
result = x > y  # نتیجه True
```

عملگرهای منطقی:

این عملگرها برای انجام عملیات منطقی استفاده می‌شوند:

- **and** : اگر هر دو شرط درست باشند، نتیجه درست است.
- **or** : اگر حداقل یکی از شروط درست باشد، نتیجه درست است.
- not

: معکوس کردن نتیجه شرط.

```
a = True
b = False
result1 = a and b   # نتیجه False
result2 = a or b    # نتیجه True
result3 = not a     # نتیجه False
```

عملگرهای انتساب:

این عملگرها برای انتساب مقادیر به متغیرها استفاده می‌شوند:

- **=** : انتساب ساده
- **+=** : افزودن به مقدار موجود
- **-=** : کم کردن از مقدار موجود
- ***=** : ضرب کردن با مقدار موجود
- =/

: تقسیم کردن با مقدار موجود

```
x = 5
x += 3  # x = x + 3 => 8
x -= 2  # x = x - 2 => 6
x *= 4  # x = x * 4 => 24
x /= 2  # x = x / 2 => 12.0
```

عملگرهای زمان‌بندی:

برای اندازه‌گیری زمان اجرای کد می‌توان از ماژول `time` و `timeit` استفاده کرد. این ابزارها می‌توانند به شما کمک کنند تا عملکرد کد خود را بهینه کنید.

- **ماژول time:** برای گرفتن زمان جاری و اندازه‌گیری زمان اجرای کد.

```
import time
start_time = time.time()  # زمان شروع
# انجام عملیات
end_time = time.time()    # زمان پایان
print(f"زمان اجرا: {end_time - start_time} ثانیه")
```

- **ماژول timeit:** برای اندازه‌گیری دقیق‌تر زمان اجرای قطعه کدها.

```
import timeit
execution_time = timeit.timeit("x = 5 + 3", number=1000)
print(f"زمان اجرای کد: {execution_time} ثانیه")
```

3. ساختارهای کنترلی

ساختارهای کنترلی به شما این امکان را می‌دهند که جریان اجرای برنامه را تغییر دهید، مانند تصمیم‌گیری (شرطی‌ها) و تکرار (حلقه‌ها).

شرطی‌ها (if, elif, else)

این ساختارها برای انجام تصمیم‌گیری‌ها و اجرای بخش‌های مختلف کد براساس شرایط خاص استفاده می‌شوند.

- **if:** بررسی یک شرط و اجرای کد در صورت درست بودن آن.
- **elif:** شرط دیگری برای بررسی اگر شرط قبلی نادرست بود.
- :else

در صورت نادرست بودن تمامی شرایط، این بخش اجرا می‌شود.

```
age = 20
if age >= 18:
    print("بزرگسال هستید.")
elif age > 12:
    print("نوجوان هستید.")
else:
    print("کودک هستید.")
```

حلقه‌ها (for, while)

حلقه‌ها برای تکرار یک بخش از کد چندین بار استفاده می‌شوند.

- **for:** برای تکرار روی عناصر یک دنباله (مثل لیست، رشته یا دیکشنری) استفاده می‌شود.

```
for i in range(5):  # از 0 تا 4
    print(i)
```

- **while:** تا زمانی که یک شرط درست باشد، کد درون حلقه اجرا می‌شود.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

دستور break و continue

- **break:** برای خروج از حلقه به محض برآورده شدن یک شرط خاص.

```
for i in range(10):
    if i == 5:
        break  # حلقه را متوقف می‌کند
    print(i)
```

- **continue:** برای عبور از یک تکرار و ادامه حلقه.

```
for i in range(5):
    if i == 3:
        continue  # این تکرار را نادیده می‌گیرد و به تکرار بعدی می‌رود
    print(i)
```

این بخش از مفاهیم پایه پایتون، شما را با اصول اولیه‌ای مانند تعریف متغیرها، انواع داده‌ها، عملگرها و ساختارهای کنترلی آشنا می‌کند که پایه‌گذار هر برنامه‌ای در پایتون خواهند بود.