Python'da **tip dönüşümleri** (type casting), bir veri tipini başka bir veri tipine dönüştürme işlemidir. Python, dinamik olarak türleri belirlediği için, bu dönüşümleri gerektiğinde elle yapmamız gerekebilir. Tip dönüşümleri iki ana kategoride incelenebilir: **otomatik (implicit)** ve **manuel (explicit)** dönüşümler.

## 1. Otomatik Tip Dönüşümleri (Implicit Type Conversion)

Python, bazı durumlarda veri türleri arasında otomatik olarak dönüşüm yapar. Özellikle sayısal işlemlerde, daha küçük bir tür (örneğin, `int`) daha büyük bir türe (örneğin, `float`) otomatik olarak dönüştürülebilir. Bu tür dönüşümlere müdahale etmemize gerek yoktur.

### Örnek:

```
# Otomatik tip dönüşümü
sayi1 = 10      # int
sayi2 = 2.5     # float
sonuc = sayi1 + sayi2  # int + float -> float
print(sonuc)  # 12.5
print(type(sonuc))  # <class 'float'>
```

Yukarıdaki örnekte, Python `int` ve `float` türleri arasında otomatik dönüşüm yaparak sonucu `float` olarak döner.

## 2. Manuel Tip Dönüşümleri (Explicit Type Conversion)

Python'da bazı veri türleri, diğer türlere **manuel olarak** dönüştürülebilir. Bu işlem, fonksiyonlar kullanılarak gerçekleştirilir. Python'daki bazı temel tip dönüşüm fonksiyonları şunlardır:

- `int()` : Değeri tam sayıya dönüştürür.
- `float()` : Değeri kayan noktalı sayıya dönüştürür.
- `str()` : Değeri metin (string) tipine dönüştürür.

- `list()`: Değeri listeye dönüştürür.
- `tuple()`: Değeri demet (tuple) tipine dönüştürür.
- `dict()`: Değeri sözlüğe dönüştürür.
- `set()`: Değeri kümeye dönüştürür.
- `bool()`: Değeri boolean (`True` veya `False`) tipine dönüştürür.

## Manuel Tip Dönüşümleri İçin Örnekler

### 2.1. `int()` Fonksiyonu

Bir değeri tam sayıya dönüştürür. Eğer değer ondalıklı bir sayı ise, ondalık kısmı atılır. Eğer dönüştürülecek değer string ise, bu string sayısal bir ifadeyi temsil etmeli, aksi takdirde hata alınır.

```
# Ondalıklı sayıyı tam sayıya dönüştürme
sayi = int(5.99)
print(sayi)  # 5

# String'i tam sayıya dönüştürme
sayi = int("10")
print(sayi)  # 10

# Hatalı dönüşüm
# sayi = int("10a")  # ValueError: invalid literal for int()
```

### 2.2. `float()` Fonksiyonu

Bir değeri kayan noktalı sayıya dönüştürür. Tam sayılar ve uygun formatlı string ifadeler ondalıklı sayıya dönüştürülebilir.

```
# Tam sayıyı float'a dönüştürme
sayi = float(5)
print(sayi)  # 5.0

# String'i float'a dönüştürme
```

```python
sayi = float("10.5")
print(sayi)  # 10.5
```

## 2.3. `str()` Fonksiyonu

Herhangi bir veri türünü string'e dönüştürür. Bu dönüşüm, sayılar, listeler ve diğer veri türleri için de kullanılabilir.

```python
# Tam sayıyı string'e dönüştürme
metin = str(100)
print(metin)  # '100'

# Listeyi string'e dönüştürme
liste = [1, 2, 3]
metin = str(liste)
print(metin)  # '[1, 2, 3]'
```

## 2.4. `list()` Fonksiyonu

Bir iterable (yinelenebilir) nesneyi listeye dönüştürür. String, tuple gibi veri türleri listeye dönüştürülebilir.

```python
# String'i listeye dönüştürme
metin = "Python"
liste = list(metin)
print(liste)  # ['P', 'y', 't', 'h', 'o', 'n']

# Tuple'ı listeye dönüştürme
demet = (1, 2, 3)
liste = list(demet)
print(liste)  # [1, 2, 3]
```

## 2.5. `tuple()` Fonksiyonu

Bir iterable nesneyi tuple (demet) tipine dönüştürür.

```python
# Listeyi tuple'a dönüştürme
liste = [1, 2, 3]
demet = tuple(liste)
print(demet)  # (1, 2, 3)

# String'i tuple'a dönüştürme
metin = "Python"
demet = tuple(metin)
print(demet)  # ('P', 'y', 't', 'h', 'o', 'n')
```

## 2.6. `dict()` Fonksiyonu

Belirli yapıları sözlük (dictionary) tipine dönüştürür. Liste veya tuple çiftleri (key-value) sözlüğe dönüştürülebilir.

```python
# Tuple listesi ile sözlük oluşturma
liste = [("isim", "Ahmet"), ("yas", 25)]
sozluk = dict(liste)
print(sozluk)  # {'isim': 'Ahmet', 'yas': 25}
```

## 2.7. `set()` Fonksiyonu

Bir iterable nesneyi kümeye dönüştürür. Kümeler, benzersiz ve sırasız elemanlar içerir.

```python
# Listeden küme oluşturma
liste = [1, 2, 2, 3, 4]
kume = set(liste)
print(kume)  # {1, 2, 3, 4}
```

## 2.8. `bool()` Fonksiyonu

Herhangi bir veri türünü boolean (`True` veya `False`) tipine dönüştürür. Boş ya da sıfır olmayan değerler `True` dönerken, boş veya sıfır değerler `False` döner.

```python
# Sayıları boolean'a dönüştürme
print(bool(1))  # True
print(bool(0))  # False

# Boş ve dolu yapılar
print(bool([]))  # False
print(bool([1, 2, 3]))  # True
```

## Tip Dönüşüm Kuralları

1. Dönüştürülen veri, geçerli bir formatta olmalıdır. Örneğin, "123" gibi bir string `int()` ile tam sayıya dönüştürülebilir ancak "123abc" gibi bir string dönüştürülemez.
2. Her veri türü başka bir veri türüne dönüştürülemez. Örneğin, bir listeyi `int()` ile tam sayıya dönüştüremezsiniz.
3. Bazı dönüşümler veri kaybına yol açabilir (örneğin, `float`'tan `int`'e dönüşüm).

## Tip Dönüşümlerinin Kullanım Alanları

- **Kullanıcı girdilerini işlemek**: Kullanıcıdan alınan veriler genellikle string formatında gelir ve gerektiğinde başka tiplere dönüştürülmesi gerekir.
- **Veri formatları arasında uyum sağlamak**: Farklı veri tipleri arasında dönüşüm yaparak fonksiyonlar ve algoritmalar arasında veri uyumluluğu sağlanır.
- **Matematiksel işlemler**: Sayılar üzerinde doğru işlemler yapabilmek için uygun veri tipine dönüştürme yapılır (örneğin, ondalıklı sayılarda kesin sonuç almak için `float` kullanmak).

Tip dönüşümleri Python'un esnekliği sayesinde yaygın olarak kullanılır ve kodu daha güçlü ve hatasız hale getirir.