Python’da **kod yazım biçimi (code style)**, **PEP 8** (Python Enhancement Proposal 8) adlı belgeye dayanır. Bu belge, Python kodlarının okunabilirliğini ve tutarlılığını artırmak için belirlenen bir dizi kural ve yönergeyi içerir. Aşağıda Python'da iyi kod yazma pratiği olarak kabul edilen bazı önemli noktaları ve yazım biçimi kurallarını bulabilirsiniz:

1. Girintileme (Indentation)

Python, kod bloklarını tanımlamak için girintileme kullanır. Her kod bloğu 4 boşluk ile girintilenmelidir. **Tab** kullanmak yerine, her zaman boşluk (space) kullanılması tavsiye edilir.

Örnek:

```
def fonksiyon():
    if True:
        print("Girintileme 4 boşluk olmalıdır.")
```

2. Satır Uzunluğu

Bir satırın uzunluğu 79 karakteri geçmemelidir. Daha uzun ifadeler için kodu birden fazla satıra bölmek gerekir.

Örnek:

```
if uzun_kod_satiri_veya_aciklama_79_karakteri_geciyorsa:
    print("Kodunuzun uzunluğunu kontrol edin.")
```

3. Fonksiyon ve Değişken İsimleri

- **Fonksiyon ve değişken isimleri** küçük harflerle yazılmalı ve kelimeler arasında alt çizgi (`_`) kullanılmalıdır (snake_case).
- **Sabitler** genellikle tamamen büyük harflerle tanımlanır (UPPER_CASE).

Örnek:

```
def hesapla_yas(dogum_yili):
    GUNCEL_YIL = 2024
    return GUNCEL_YIL - dogum_yili
```

4. Sınıf İsimlendirme

- **Sınıf isimleri**, büyük harfle başlamalı ve her kelimenin ilk harfi büyük olacak şekilde yazılmalıdır (**PascalCase**).

Örnek:

```
class OgrenciBilgisi:
    pass
```

5. Yorum Satırları (Comments)

- **Satır içi yorumlar** çok kısa ve açıklayıcı olmalıdır. Yorumlar kodun ne yaptığını değil, neden yapıldığını açıklamalıdır.
- **Tek satırlık yorumlar** için `#` işareti kullanılır. Bu yorum, kodun üzerinde veya yanında bulunur.
- Daha uzun açıklamalar için **çok satırlı yorumlar** kullanabilirsiniz.

Örnek:

```
# Yaşı hesaplayan fonksiyon
def yas_hesapla(dogum_yili):
```

```
    # Güncel yıla göre yaş hesaplama
    return 2024 - dogum_yili
```

6. Boşluk Kullanımı (Whitespace)

- **Operatörler ve değişkenler arasında boşluk** bırakılmalıdır.
- Parantez, köşeli parantez ve süslü parantezlerin içinde boşluk kullanılmamalıdır.

Örnek:

```
x = 5  # Operatörden önce ve sonra boşluk
liste = [1, 2, 3]  # Parantezler arasında boşluk yok
```

7. Fonksiyon ve Sınıf Tanımlamaları Arasında Boş Satır

- Bir fonksiyon ya da sınıf tanımından önce ve sonra iki satır boşluk bırakılmalıdır.

Örnek:

```
class Sinif:
    pass


def fonksiyon():
    pass
```

8. İç İçe Fonksiyonlar ve Yapılar

İç içe yapıların kullanımı mümkün olduğunca basit tutulmalıdır. Aşırı derinlikte iç içe yapılar, kodun okunabilirliğini zorlaştırabilir.

Örnek:

```
def dis_fonksiyon():
    def ic_fonksiyon():
        pass
    ic_fonksiyon()
```

9. Modül ve Paket İsimlendirme

Modül isimleri genellikle küçük harflerle yazılmalı ve alt çizgi kullanılmamalıdır. Dosya isimlendirme **kısa** ve **anlamlı** olmalıdır.

Örnek:

```
# Modül adı: ogrenci.py
```

10. İthalatlar (Imports)

- İthalatlar (import), dosyanın başında bulunmalı ve her zaman ayrı satırlarda yapılmalıdır.
- Bir modülden birden fazla şey ithal ediyorsanız, her birini ayrı satırlarda belirtmek yerine tek satırda yazabilirsiniz.
- `import` ifadeleri üç bölümde gruplanmalıdır: **standart kütüphaneler**, **üçüncü taraf kütüphaneler** ve **yerel modüller**.

Örnek:

```
# Doğru şekilde ithalat
import os
import sys

# Yanlış şekilde ithalat
import sys, os

# Üçüncü taraf ve yerel modüller
```

```
import numpy as np
import ogrenci
```

11. Dokümantasyon Dizgeleri (Docstrings)

Her fonksiyon, sınıf ve modül için açıklayıcı bir **dokümantasyon dizgesi** eklenmelidir. Bu, fonksiyonun ne yaptığını açıklayan kısa bir paragraf içermelidir. Üç tırnak işareti (`"""`) ile yazılır.

Örnek:

```
def yas_hesapla(dogum_yili):
    """
    Verilen doğum yılına göre kişinin yaşını hesaplar.

    Args:
        dogum_yili (int): Doğum yılı

    Returns:
        int: Yaş
    """
    return 2024 - dogum_yili
```

12. İstisna İşleme (Exception Handling)

Python'da istisna yönetimi için `try`-`except` blokları kullanılır. İstisna türleri belirtilmeli ve her zaman spesifik hatalar yakalanmalıdır. Genel `except` blokları kullanmaktan kaçınılmalıdır.

Örnek:

```
try:
    dosya = open('dosya.txt', 'r')
except FileNotFoundError:
    print("Dosya bulunamadı.")
```

```
except Exception as e:
    print(f"Bir hata oluştu: {e}")
```

Python'da Kod Yazımının Önemi

- **Okunabilirlik**: Python'un temel felsefelerinden biri "okunabilir kod" yazmaktır. İyi biçimlendirilmiş kod, sadece yazan kişi için değil, başkaları tarafından da kolayca anlaşılabilir.
- **Bakım Kolaylığı**: Tutarlı kod yazımı, kodun ileride bakımını ve genişletilmesini kolaylaştırır.
- **Hata Azaltma**: Düzenli ve iyi yapılandırılmış kod, hataların erken fark edilmesine yardımcı olabilir.

Python'da kod yazarken bu kurallara uymak, daha temiz, tutarlı ve anlaşılır bir kod oluşturmanızı sağlar. PEP 8'e uymak, özellikle büyük projelerde ve işbirlikçi çalışmalarda büyük fayda sağlar.