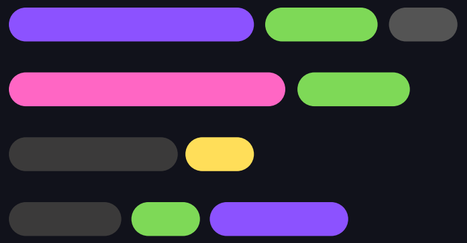
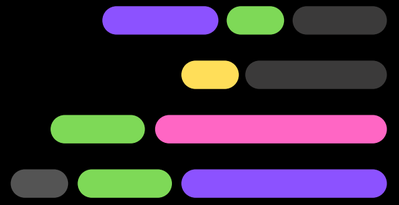
{TEMEL PROGRAMLAMA II}

Samsun Üniversitesi
Teknik Bilimler Meslek Yüksekokulu
Arka-Yüz Yazılım Geliştirme Pr.



Öğr. Gör. Furkan DURMUŞ

Bu Haftanın Ders Kazanımları



01 Temel Tekrar : Fonksiyonlar

02 Parametre Kavramı

03 Return ve Akış

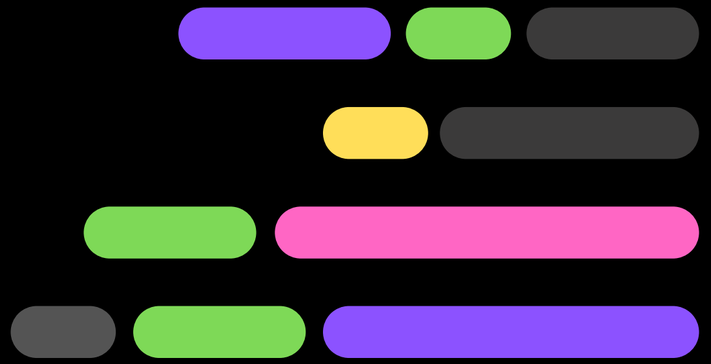
04 SOLID Giriş

05 SOLID Uygulama

Temel Tekrar : Fonksiyonlar

01

Temel Tekrar : Fonksiyonlar

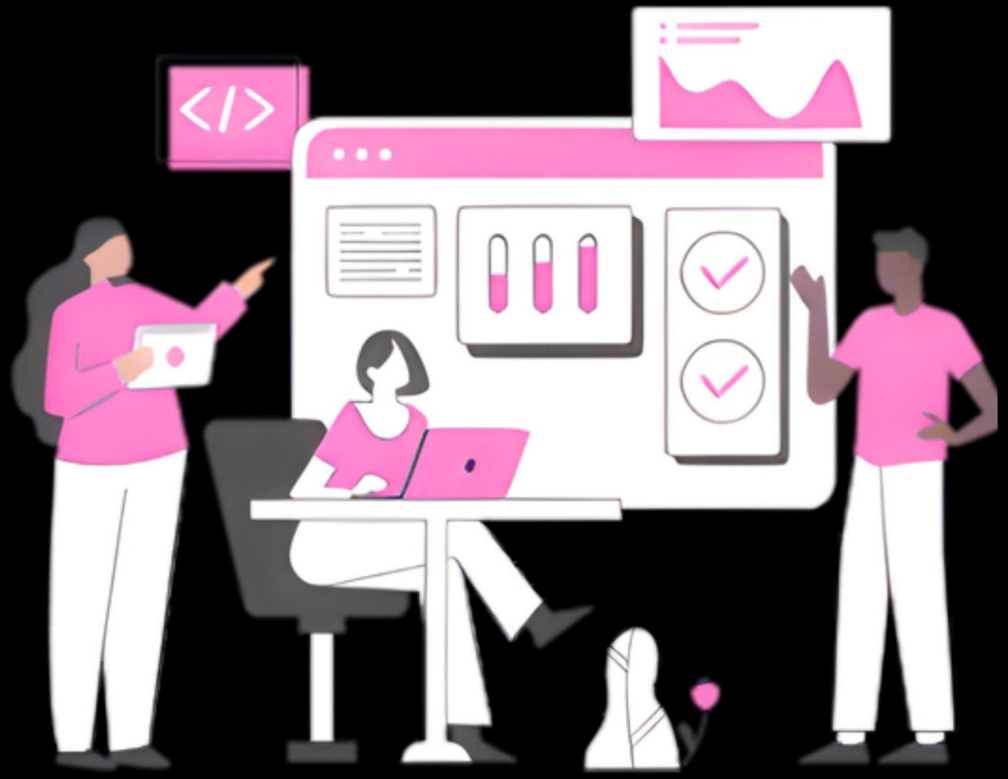


Fonksiyonlara neden ihtiyaç duyarız ?

```python
pi = 3.14159
# 1. Dairenin Alanı
yari_cap1 = 5
alan1 = pi * (yari_cap1 ** 2)
print(f"Yarıçapı 5 olan dairenin alanı: {alan1:.2f}")

# 2. Dairenin Alanı
yari_cap2 = 8
alan2 = pi * (yari_cap2 ** 2)
print(f"Yarıçapı 8 olan dairenin alanı: {alan2:.2f}")
```

01

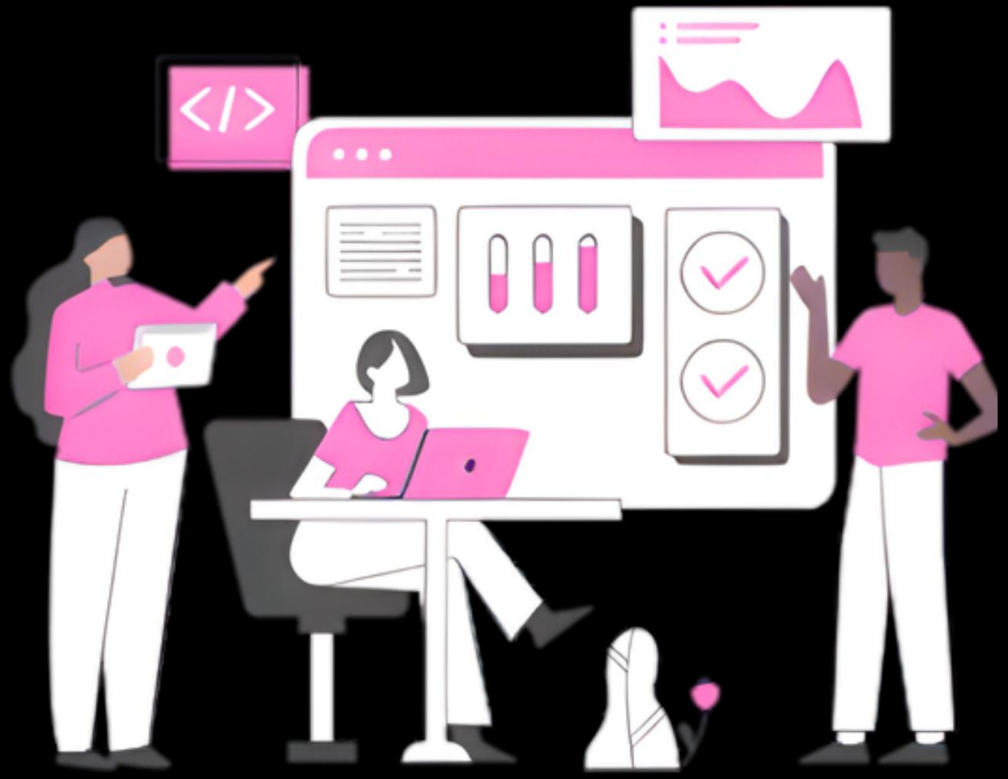
Temel Tekrar : Fonksiyonlar



Fonksiyon Tanımlama

01

Temel Tekrar : Fonksiyonlar



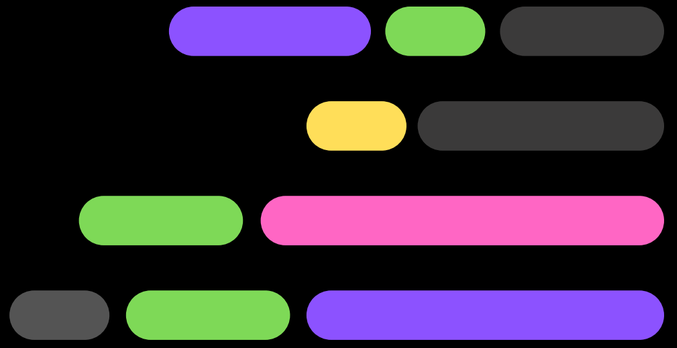
Fonksiyon Tanımlama

- İsimlendirme kurallarına dikkat edilmeli
- Fonksiyon işlevine göre isimlendirilmeli
- def ile tanımlanır
- Parantez ve : zorunlu
- Girinti şart
- Global ve local değişkenler

01 Temel Tekrar : Fonksiyonlar

```python
pi = 3.14159

# 1. Dairenin Alanı
yari_cap1 = 5
alan1 = pi * (yari_cap1 ** 2)
print(f"Yarıçapı 5 olan dairenin alanı: {alan1:.2f}")

# 2. Dairenin Alanı
yari_cap2 = 8
alan2 = pi * (yari_cap2 ** 2)
print(f"Yarıçapı 8 olan dairenin alanı: {alan2:.2f}")

# 3. Dairenin Alanı
yari_cap2 = 4
alan2 = pi * (yari_cap2 ** 2)
print(f"Yarıçapı 4 olan dairenin alanı: {alan2:.2f}")

# 4. Dairenin Alanı
yari_cap2 = 6
alan2 = pi * (yari_cap2 ** 2)
print(f"Yarıçapı 6 olan dairenin alanı: {alan2:.2f}")
```

Fonksiyon olarak yazalım.



Parametre Kavramı

02

Parametre Kavramı

Fonksiyonun çalışırken kullanacağı, dışarıdan aldığı değerlere …… denir.

02

Parametre Kavramı

Temel kavramlar: Parametre vs Argüman

Parametre: Fonksiyon tanımlanırken parantez içinde yazılan isimler.

Argüman: Fonksiyon çağrılırken gönderilen gerçek değerler.

```
def selamla(isim):
    print("Merhaba", isim)

selamla("Ayşe")
```

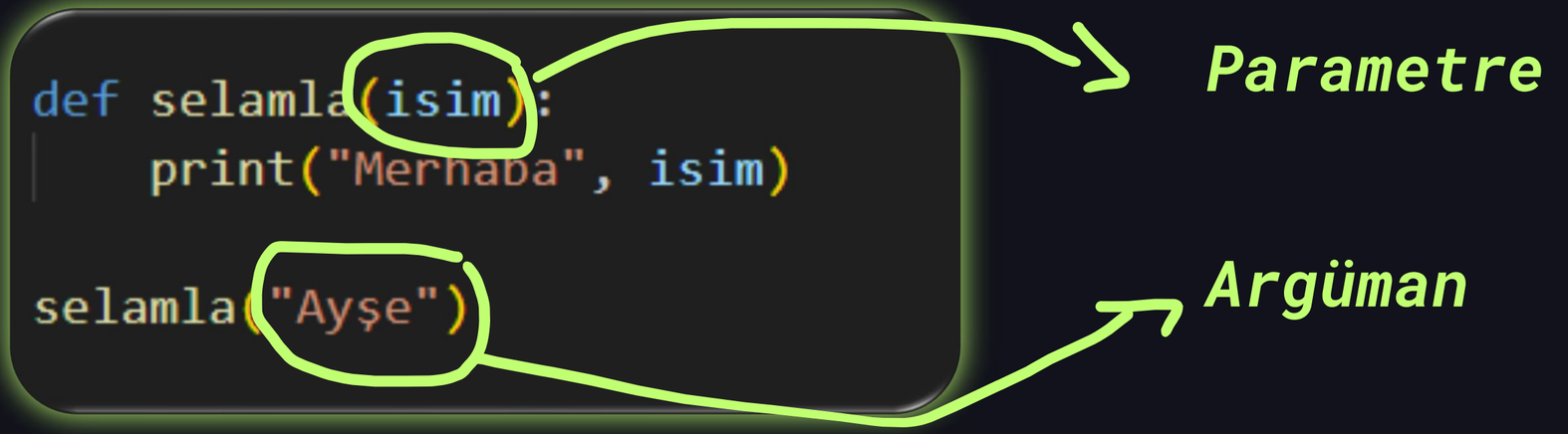
02

Parametre Kavramı

Temel kavramlar: Parametre vs Argüman

Parametre: Fonksiyon tanımlanırken parantez içinde yazılan isimler.

Argüman: Fonksiyon çağrılırken gönderilen gerçek değerler.



02

Parametre Kavramı

Argüman Türleri : ***Positional*** & Keyword

```python
def my_function(sehir, ulke):
    print(f"{sehir} {ulke}'nin bir şehridir.")


my_function("Türkiye","Samsun")
```

Ekran çıktısı nedir ?

02

Parametre Kavramı

Argüman Türleri : ***Positional*** & Keyword

```python
def my_function(sehir, ulke):
    print(f"{sehir} {ulke}'nin bir şehridir.")


my_function("Türkiye","Samsun")
```

Positional argümanlarda sıralama önemlidir !

02

Parametre Kavramı

Argüman Türleri : ***Positional*** & Keyword

```python
def bolme(a, b):
    print(a / b)

bolme(10, 2)
bolme(2, 10)
```

Positional argümanlarda sıralama önemlidir !

02

Parametre Kavramı

Argüman Türleri : Positional & ***Keyword***

```python
def my_function(sehir, ulke):
    print(f"{sehir} {ulke}'nin bir şehridir.")


my_function( ulke="Türkiye", sehir="Samsun" )
```

Ekran çıktısı nedir ?

02

Parametre Kavramı

Argüman Türleri : Positional & ***Keyword***

```python
def my_function(sehir, ulke):
    print(f"{sehir} {ulke}'nin bir şehridir.")


my_function( ulke="Türkiye", sehir="Samsun" )
```

Keyword argüman yapısında parametreler **isimleriyle gönderilir**, sıra önemli değildir.

02

Parametre Kavramı

Kural !

Pozitional ve keyword argüman bir arada kullanılacaksa

önce positional, sonra keyword argümanlar yazılır.

```
def ogrenci(ad, soyad, notu):
    print(ad, soyad, notu)

ogrenci("Ali", soyad="Yılmaz", notu=85)
ogrenci(ad="Ali", "Yılmaz", notu=85)    Positional argument cannot appear after keyword arguments
```

02

Parametre Kavramı

```python
def kullanici_kayit(kullanici_adi, email, sehir, yas):
    print("Kullanıcı:", kullanici_adi)
    print("E-posta:", email)
    print("Şehir:", sehir)
    print("Yaş:", yas)

kullanici_kayit(
    email="ali@mail.com",
    sehir="Ankara",
    kullanici_adi="ali123",
    yas=22
)
```

02

Parametre Kavramı

Kullanıcıdan elektrik tüketim miktarı, birim fiyat ve sabit ücret bilgilerini klavyeden alan bir program yazınız.

```
def elektrik_faturasi(tuketim, birim_fiyat, sabit_ucret):


tuketim     =
birim_fiyat =
sabit_ucret =
```

02

Parametre Kavramı

Kullanıcıdan elektrik tüketim miktarı, birim fiyat ve sabit ücret bilgilerini klavyeden alan bir program yazınız.

```python
def elektrik_faturasi(tuketim, birim_fiyat, sabit_ucret):
    toplam = (tuketim * birim_fiyat) + sabit_ucret
    print("Fatura Tutarı:", toplam)


tuketim     = float(input("Elektrik tüketimini giriniz (kWh): "))
birim_fiyat = float(input("Birim fiyatı giriniz: "))
sabit_ucret = float(input("Sabit ücreti giriniz: "))


elektrik_faturasi(
    tuketim=tuketim,
    birim_fiyat=birim_fiyat,
    sabit_ucret=sabit_ucret
)
```

02

Parametre Kavramı

Default (varsayılan) Değerli Parametreler

```python
def selamla(isim, mesaj="Merhaba"):
    print(mesaj, isim)

selamla("Deniz")
selamla("Deniz", "Günaydın")
selamla(isim="Deniz", mesaj="Hi")
```

Varsayılan değerli parametreler en sonda olmalı.

02

Parametre Kavramı

Verilen iki sayıyı toplayan bir fonksiyon yazmak istiyoruz. Sözlü olarak bu fonksiyonu söyleyiniz.

02

Parametre Kavramı

Verilen iki sayıyı toplayan bir fonksiyon yazmak istiyoruz.

```
def topla_iki_sayi(a, b):
    print ( a + b )
```

Verilen üç sayıyı toplayan bir fonksiyon yazmak istiyoruz. Sözlü olarak bu fonksiyonu söyleyiniz.

02

Parametre Kavramı

Verilen iki sayıyı toplayan bir fonksiyon yazmak istiyoruz.

```
def topla_iki_sayi(a, b):
    print ( a + b )
```

Verilen üç sayıyı toplayan bir fonksiyon yazmak istiyoruz.

```
def topla_uc_sayi(a, b, c):
    print ( a + b + c )
```

Verilen ‘n’ sayıyı (ör. 100 sayıyı) toplayan bir fonksiyon yazmak istiyoruz.

02

Parametre Kavramı

Verilen ‘n’ sayıyı (ör. 100 sayıyı) toplayan bir fonksiyon yazmak istiyoruz.

- Önceki yöntem artık sürdürülebilir değil.
- Nasıl bir yöntem bu problemi çözerdi ?

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

Bir fonksiyona kaç tane positional argüman geleceğini bilmiyorsak ***args** kullanırız. args bir isimdir, önemli olan * işaretidir.

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

Bir fonksiyona kaç tane positional argüman geleceğini bilmiyorsak ***args** kullanırız. args bir isimdir, önemli olan * işaretidir.

args bir tuple'dır.

```
def my_function(*args):
    print(args)

my_function(1, 2, 3)
my_function(5, 10)
my_function(7)
```

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

Bir fonksiyona kaç tane positional argüman geleceğini bilmiyorsak ***args** kullanırız.

args bir isimdir, önemli olan * işaretidir.

args bir tuple'dır.

**** : Bir dizi veya demet (list, tuple) gibi konumsal elemanları paketler veya açar.***

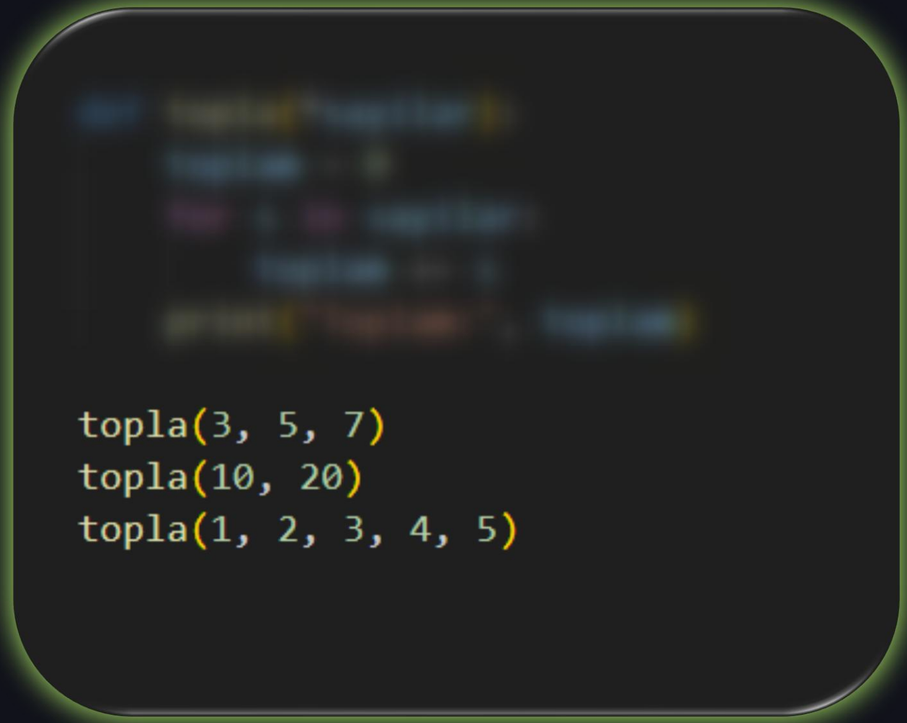
```
def my_function(*args):
    print(args)

sayilar = (1,2,3,4,5)
my_function(*sayilar)
```

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

Kendisine gönderilen argümanların toplamını hesaplayan bir Python programı yazınız.



02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

```python
def yazdir(baslik, *metinler):
    print("Başlık:", baslik)
    for m in metinler:
        print("-", m)

yazdir("Duyuru", "Sınav var", "Saat 10'da", "Sınıf A")
```

Ekran çıktısı nedir ?

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

```python
def yazdir(baslik, *metinler):
    print("Başlık:", baslik)
    for m in metinler:
        print("-", m)

yazdir("Duyuru", "Sınav var", "Saat 10'da", "Sınıf A")
```

```
Başlık: Duyuru
- Sınav var
- Saat 10'da
- Sınıf A
```

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

```python
def yazdir(baslik, *metinler):
    print("Başlık:", baslik)
    for m in metinler:
        print("-", m)

yazdir("Duyuru", "Sınav var", "Saat 10'da", "Sınıf A")
```

```
Başlık: Duyuru
- Sınav var
- Saat 10'da
- Sınıf A
```

Önce normal parametreler, **sonra** *args yazılır.

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

Fonksiyona kaç tane keyword argüman geleceğini bilmiyorsak ** kullanılır.

```python
def yazdir(**kwargs):
    print(kwargs)

yazdir(ad="Ali", soyad="Yılmaz", yas=22)
```

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

```python
def yazdir(**kwargs):
    print(kwargs)

yazdir(ad="Ali", soyad="Yılmaz", yas=22)
```

```
{'ad': 'Ali', 'soyad': 'Yılmaz', 'yas': 22}
```

kwargs = keyword arguments

***** : Bir sözlük (dict) gibi anahtar-değer çiftlerini paketler veya açar.***

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

```python
def ogrenci_bilgileri(**bilgiler):
    print(f"Gelen argümanlar: {bilgiler}")
    print(f"Paketin tipi: {type(bilgiler)}")

    for anahtar, deger in bilgiler.items():
        print(f"{anahtar.capitalize()}: {deger}")

ogrenci_bilgileri(isim="ayşe", soyisim="yılmaz", numara=12345)
```

```
Gelen argümanlar: {'isim': 'ayşe', 'soyisim': 'yılmaz', 'numara': 12345}
Paketin tipi: <class 'dict'>
Isim: ayşe
Soyisim: yılmaz
Numara: 12345
```

02

Parametre Kavramı

Değişken Sayıda Argüman (*args, **kwargs)

args (“arguments”ın kısaltması) ve ***kwargs*** (“keyword arguments”ın kısaltması) ***isimleri zorunlu değildir***, sadece Python topluluğunda yaygın olarak kullanılan bir ***gelenektir*** (convention).

02

Parametre Kavramı

Parametrelerde Tip İpucu (type hints)

Type hint, fonksiyon parametrelerinin ve dönüş değerinin hangi türde olması gerektiğini belirtmek için kullanılan ipucudur.

02

Parametre Kavramı

Parametrelerde Tip İpucu (type hints)

Type hint **zorunlu değildir**, Python bunu çalışma anında **kontrol etmez**.

Daha çok okuyan insan ve editörler (VS Code, PyCharm) içindir.

```python
def topla(a: int, b: int):
    print(a + b)

topla(3, 5)
```

02

Parametre Kavramı

Parametrelerde Tip İpucu (type hints)

Type hint **zorunlu değildir**, Python bunu çalışma anında **kontrol etmez**.

Daha çok okuyan insan ve editörler (VS Code, PyCharm) içindir.

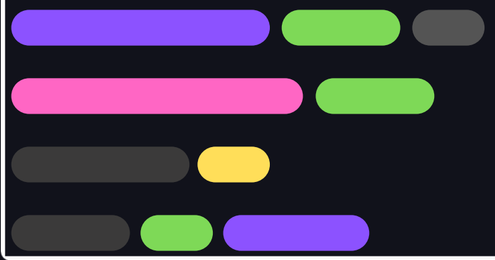
```python
def topla(a: int, b: int):
    print(a + b)

topla("3", "5")
```

Ekran çıktısı nedir ?

02

Parametre Kavramı

Şu an fonksiyonlar hep konuştu(print).
Artık fonksiyonlardan geri değer almayı öğreneceğiz.



Return Kavramı

03

Return Kavramı

return, bir fonksiyonun ürettiği sonucu fonksiyonun dışına göndermesini sağlayan anahtar kelimedir.

03

Return Kavramı

```python
def topla(a, b):
    print(a + b)

sonuc = topla(3, 5)
print(sonuc)
```

```python
def topla(a, b):
    return a + b

sonuc = topla(3, 5)
print(sonuc)
```

Ekran çıktısı nedir ?

03

Return Kavramı

```python
def topla(a, b):
    print(a + b)

sonuc = topla(3, 5)
print(sonuc)
```

```
8
None
```

print ekrana yazar,
return değeri geri verir.

```python
def topla(a, b):
    return a + b

sonuc = topla(3, 5)
print(sonuc)
```

```
8
```

03

Return Kavramı

```python
def topla(a, b):
    print(a + b)

sonuc = topla(3, 5)
print(sonuc)
```

Fonksiyonlarda return yazılmasa bile, Python varsayılan olarak None döndürür.

```python
def topla(a, b):
    print(a + b)
    return None

sonuc = topla(3, 5)
print(sonuc)
```

```
8
None
```

03

Return Kavramı

```python
def kontrol(x):
    print("Başladı")
    return x * 2
    print("Bitti")

kontrol(5)
```

Yandaki kodu bilgisayarınızda çalıştırınız ve yorumlayınız.

03

Return Kavramı

```python
def kontrol(x):
    print("Başladı")
    return x * 2
    print("Bitti")

kontrol(5)
```

Yandaki kodu bilgisayarınızda çalıştırınız ve yorumlayınız.

NOT : ***return çalıştığı anda fonksiyon biter.(break gibi)***

03

Return Kavramı

return Nerelerde Kullanılır?

03

Return Kavramı

return Nerelerde Kullanılır?

- Hesaplama fonksiyonları
- Veri döndüren işlemler
- Başka fonksiyonların girdisi olacak sonuçlar
- Test edilebilir kod yazmak için

03

Return Kavramı

```python
def resit_mi(yas):
    if yas >= 18:
        return True
    else:
        return False

yas = int(input("Yaşınızı giriniz: "))
sonuc = resit_mi(yas)

if sonuc:
    print("Ehliyet alabilir.")
else:
    print("Ehliyet alamaz.")
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def yas_hesapla(dogum_yili):
    return 2026 - dogum_yili
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def bol(a, b):
    if b == 0:
        print("Sıfıra bölme hatası")
        return
    print(a / b)

bol(10, 0)
bol(10, 2)
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def islem(a, b):
    return a + b, a - b, a * b

toplam, fark, carpim = islem(10, 3)
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def ciftler(n):
    sonuc = []
    for i in range(n):
        if i % 2 == 0:
            sonuc.append(i)
    return sonuc
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def ogrenci(ad, yas):
    return {"ad": ad, "yas": yas}
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

```python
def kare(x):
    return x * x

def iki_kati(x):
    return x * 2

sonuc = iki_kati(kare(5))
print(sonuc)
```

Yandaki kodu yorumlayınız.

03

Return Kavramı

Verilen bir sayının mutlak değerini döndüren fonksiyonu return kullanarak yazınız.

```
def mutlak_deger(sayi):




sonuc = mutlak_deger(-10)
print(sonuc)
```

03

Return Kavramı

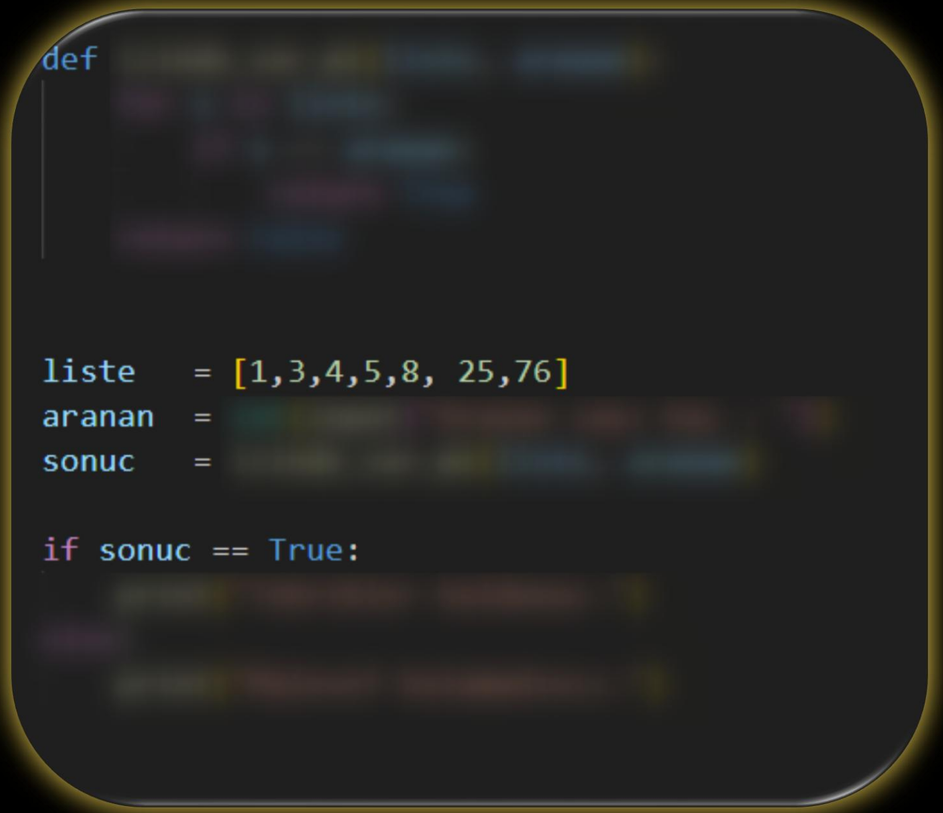
Verilen bir sayının mutlak değerini döndüren fonksiyonu return kullanarak yazınız.

```python
def mutlak_deger(sayi):
    if sayi < 0:
        return -sayi
    return sayi


sonuc = mutlak_deger(-10)
print(sonuc)
```

03

Return Kavramı

1, 3, 4, 5, 8, 25 ve 76 sayılarından oluşan bir liste oluşturunuz. Kullanıcıdan bir sayı alınız ve bu sayıyı liste ile birlikte bir fonksiyona göndererek, girilen sayının listede olup olmadığını return eden bir program yazınız.



03

Return Kavramı

1, 3, 4, 5, 8, 25 ve 76 sayılarından oluşan bir liste oluşturunuz. Kullanıcıdan bir sayı alınız ve bu sayıyı liste ile birlikte bir fonksiyona göndererek, girilen sayının listede olup olmadığını return eden bir program yazınız.

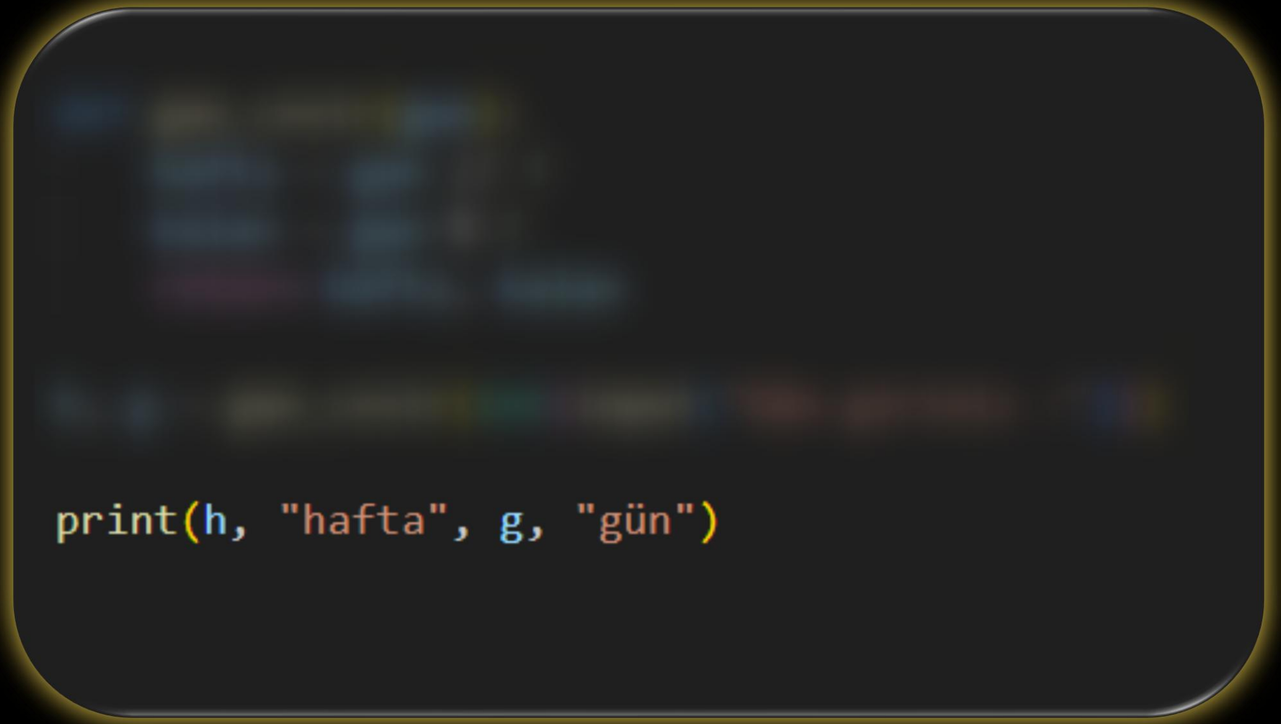
```python
def icinde_var_mi(liste, aranan):
    for x in liste:
        if x == aranan:
            return True
    return False


liste   = [1,3,4,5,8, 25,76]
aranan  = int(input("Aranan sayı kaç : "))
sonuc   = icinde_var_mi(liste, aranan)

if sonuc == True:
    print("Tebrikler buldunuz.")
else:
    print("Malesef bulamadınız.")
```

03

Return Kavramı

Kullanıcıdan alınan gün sayısının kaç hafta kaç gün'e denk geldiğini ekrana yazdıran Python programını yazınız.

Örn :
Girdi -> 17
Çıktı -> 2 hafta 3 gün



03

Return Kavramı

Kullanıcıdan alınan gün sayısının kaç hafta kaç gün'e denk geldiğini ekrana yazdıran Python programını yazınız.

Örn :
Girdi -> 17
Çıktı -> 2 hafta 3 gün

```python
def gun_cevir(gun):
    hafta = gun // 7
    kalan = gun % 7
    return hafta, kalan

h, g = gun_cevir(int(input("Gün giriniz :")))

print(h, "hafta", g, "gün")
```

03

Return Kavramı

```python
def mutlak_deger(sayi) -> int:
    if sayi < 0:
        return -sayi
    return sayi
```

```python
def icinde_var_mi(liste, aranan) -> bool:
    for x in liste:
        if x == aranan:
            return True
    return False
```

```python
def ogrenci(ad, yas) -> dict:
    return {"ad": ad, "yas": yas}
```

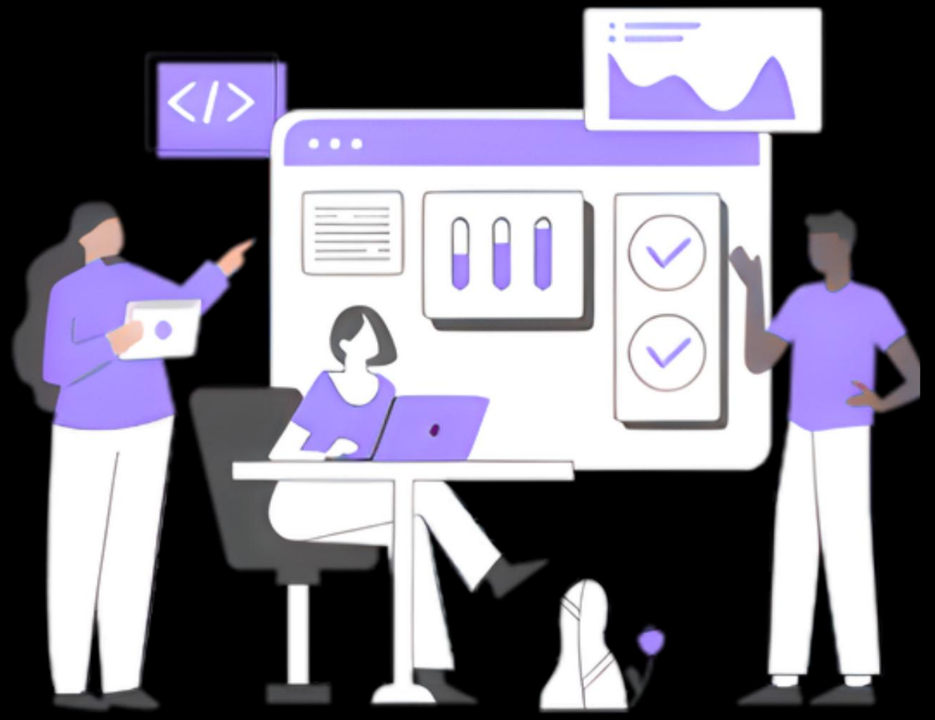
Fonksiyonlarda type hint, parametrelerin ve fonksiyonun döndürdüğü değerin hangi veri tipinde olması gerektiğini belirtmek için kullanılan bir ipucu sistemidir. ***Bilgi amaçlı kullanılır, kural değildir!***

SOLID Giriş

04

SOLID Giriş

SOLID, daha düzgün, anlaşılır ve sürdürülebilir kod yazmak için ortaya atılmış 5 temel prensibin baş harflerinden oluşur.



04

SOLID Giriş

S – Single Responsibility Principle

(Tek Sorumluluk Prensibi)

O – Open / Closed Principle

(Açık / Kapalı Prensibi)

L – Liskov Substitution Principle

I – Interface Segregation Principle

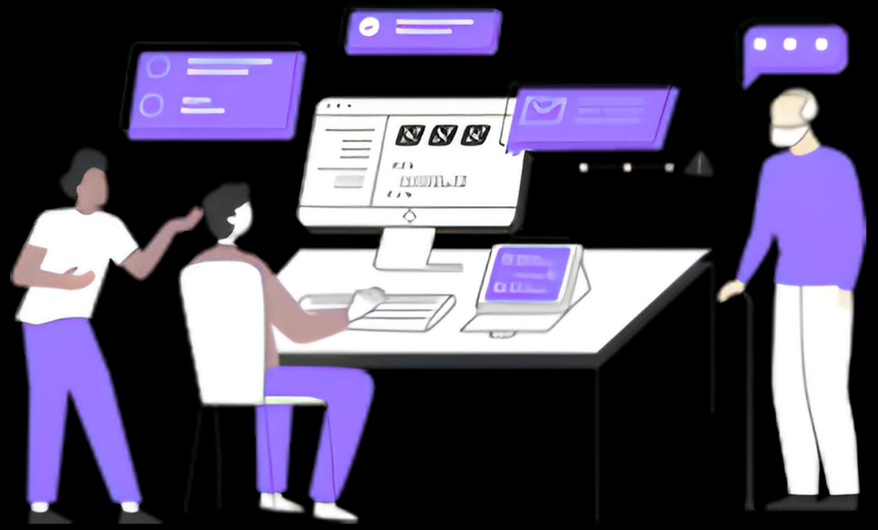
D – Dependency Inversion Principle

04

SOLID Giriş

SOLID ne değildir?

- Bir Python kuralı değildir.
- Kod çalışsın diye zorunlu değildir.
- Yeni başlayan için şart değildir.



SOLID ne içindir?

- Kod büyüdüğünde kontrol edebilebilsin,
- Kod başkası tarafından da okunabilsin,
- Kod değiştirilebilir olsun diye.

04

SOLID Giriş

SOLID ne değildir?

- Bir Python kuralı değildir
- Kod çalışsın diye zorunlu değildir
- Yeni başlayan için şart değildir

SOLID ne içindir?

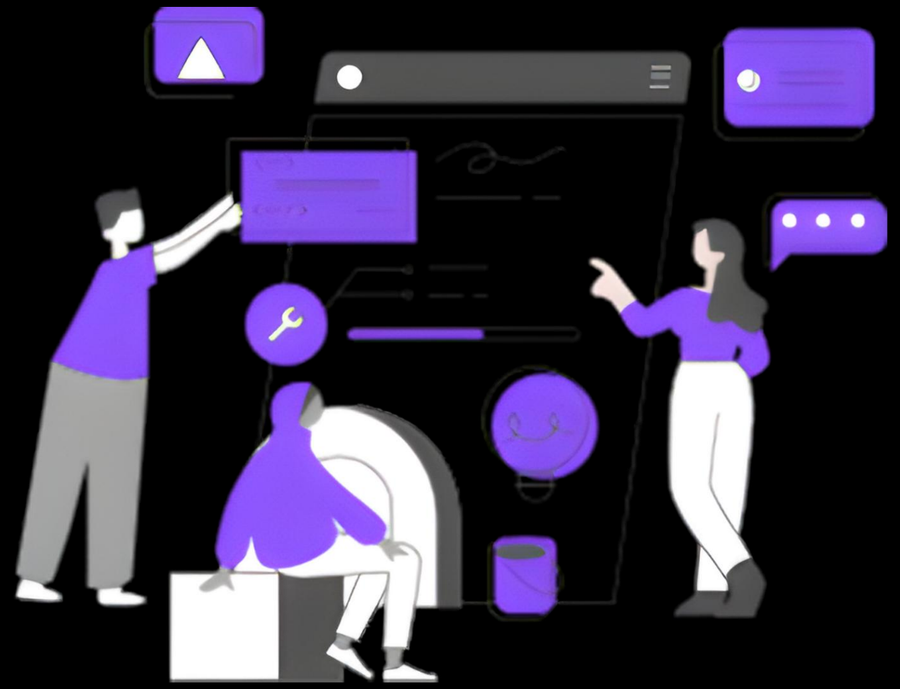
- Kod büyüdüğünde kontrol edebilebilsin
- Kod başkası tarafından da okunabilsin
- Kod değiştirilebilir olsun diye.

04

SOLID Giriş

Single Responsibility Principle
(Tek Sorumluluk) nedir?

Bir fonksiyon / sınıf / yapı
sadece TEK İŞ yapmalıdır.



04

SOLID Giriş

Single Responsibility Principle

```python
def ogrenci_islemleri(ad, not1, not2):
    ortalama = (not1 + not2) / 2
    print("Öğrenci:", ad)
    print("Ortalama:", ortalama)
    with open("sonuclar.txt", "a") as f:
        f.write(ad + " " + str(ortalama) + "\n")
```

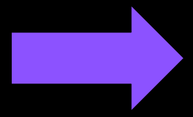
Bir fonksiyon / sınıf / yapı sadece TEK İŞ yapmalıdır.

04

SOLID Giriş

Single Responsibility Principle

```python
def ogrenci_islemleri(ad, not1, not2):
    ortalama = (not1 + not2) / 2
    print("Öğrenci:", ad)
    print("Ortalama:", ortalama)
    with open("sonuclar.txt", "a") as f:
        f.write(ad + " " + str(ortalama) + "\n")
```



```python
def ortalama_hesapla(n1, n2):
    return (n1 + n2) / 2

def ekrana_yaz(ad, ortalama):
    print("Öğrenci:", ad)
    print("Ortalama:", ortalama)

def dosyaya_yaz(ad, ortalama):
    with open("sonuclar.txt", "a") as f:
        f.write(ad + " " + str(ortalama) + "\n")
```

Bir fonksiyon / sınıf / yapı sadece TEK İŞ yapmalıdır.

04

SOLID Giriş

Single Responsibility Principle

```python
def kullanici_kayit(ad, email, sifre):
    if "@" not in email:
        print("Hatalı email")
        return

    if len(sifre) < 8:
        print("Zayıf şifre")
        return

    print("Kullanıcı kaydedildi")
```

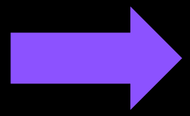
Yukarıdaki kodu Tek Sorumluluk kuralına göre bölünüz.

04

SOLID Giriş

Single Responsibility Principle

```python
def kullanici_kayit(ad, email, sifre):
    if "@" not in email:
        print("Hatalı email")
        return

    if len(sifre) < 8:
        print("Zayıf şifre")
        return

    print("Kullanıcı kaydedildi")
```



```python
def email_gecerli_mi(email):
    return "@" in email

def sifre_gecerli_mi(sifre):
    return len(sifre) >= 8

def kayit_yap():
    print("Kullanıcı kaydedildi")
```

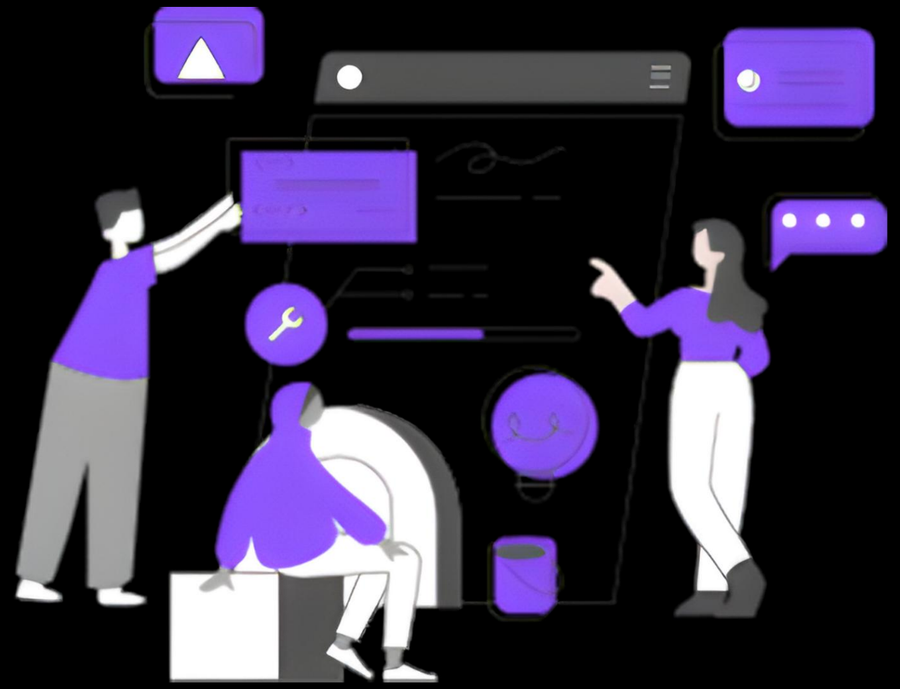
Yukarıdaki kodu Tek Sorumluluk kuralına göre bölünüz.

04

SOLID Giriş

Open / Closed Principle
(Açık / Kapalı)nedir?

Kod genişletilmeye açık,
değiştirilmeye kapalı olmalı.



04

SOLID Giriş

Open / Closed Principle

```python
def bildirim_gonder(tip, mesaj):
    if tip == "email":
        print("Email gönderildi:", mesaj)
    elif tip == "sms":
        print("SMS gönderildi:", mesaj)
```

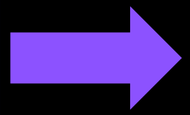
Kod genişletilmeye açık, değiştirilmeye kapalı olmalı. Yani yeni özellik eklerken eski kod riske atılmamalı.

04

SOLID Giriş

Open / Closed Principle

```python
def bildirim_gonder(tip, mesaj):
    if tip == "email":
        print("Email gönderildi:", mesaj)
    elif tip == "sms":
        print("SMS gönderildi:", mesaj)
```



```python
def email_gonder(mesaj):
    print("Email gönderildi:", mesaj)

def sms_gonder(mesaj):
    print("SMS gönderildi:", mesaj)

def whatsapp_gonder(mesaj):
    print("WhatsApp gönderildi:", mesaj)
```

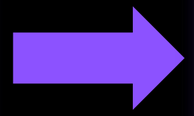
Kod genişletilmeye açık, değiştirilmeye kapalı olmalı. Yani yeni özellik eklerken eski kod riske atılmamalı.

04

SOLID Giriş

Open / Closed Principle

```python
def email_gonder(mesaj):
    print("Email gönderildi:", mesaj)

def sms_gonder(mesaj):
    print("SMS gönderildi:", mesaj)

def whatsapp_gonder(mesaj):
    print("WhatsApp gönderildi:", mesaj)
```



```python
def email_gonder(mesaj):
    print("Email gönderildi:", mesaj)

def sms_gonder(mesaj):
    print("SMS gönderildi:", mesaj)

def whatsapp_gonder(mesaj):
    wp_numarasi = int(input("Tel no giriniz"))
    print(f"{wp_numarasi} numarasına wp mesajı gönderildi:", mesaj)
```

Kod genişletilmeye açık, değiştirilmeye kapalı olmalı. Yani yeni özellik eklerken eski kod riske atılmamalı.

SOLID Uygulama

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.

```
fiyatlar = [300, 400, 500]
```

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.
- Sepetteki ürün fiyatlarını alıp toplam tutarı hesaplayan bir fonksiyon yazınız.

```
tutar = sepet_toplami(fiyatlar)
#Sepetin toplamını hesaplayan fonksiyon sadece bu işi yapar (S – tek sorumluluk).
```

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.
- Sepetteki ürün fiyatlarını alıp toplam tutarı hesaplayan bir fonksiyon yazınız.
- Toplam tutar 1000 TL'den büyükse %10 indirim uygulanacaktır.

```
tutar = indirim_uygula(tutar)
# İndirim kuralı ayrı bir fonksiyondadır; kural değişirse sadece burası etkilenir (S).
```

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.
- Sepetteki ürün fiyatlarını alıp toplam tutarı hesaplayan bir fonksiyon yazınız.
- Toplam tutar 1000 TL'den büyükse %10 indirim uygulanacaktır.
- İndirimden sonra %20 KDV eklenecek.

```
tutar = kdv_ekle(tutar)
#  Vergi hesaplama ayrı tutulduğu için başka projelerde de yeniden kullanılabilir (S).
```

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.
- Sepetteki ürün fiyatlarını alıp toplam tutarı hesaplayan bir fonksiyon yazınız.
- Toplam tutar 1000 TL'den büyükse %10 indirim uygulanacaktır.
- İndirimden sonra %20 KDV eklenecek.
- Her siparişe 50 TL kargo ücreti eklenecek.

```
tutar = kargo_ekle(tutar)
#  Kargo ücreti değişirse sistemin geri kalanı etkilenmez (S + bakım kolaylığı).
```

05

SOLID Uygulama

- fiyatlar = [300, 400, 500] olan bir liste oluşturunuz.
- Sepetteki ürün fiyatlarını alıp toplam tutarı hesaplayan bir fonksiyon yazınız.
- Toplam tutar 1000 TL'den büyükse %10 indirim uygulanacaktır.
- İndirimden sonra %20 KDV eklenecek.
- Her siparişe 50 TL kargo ücreti eklenecek.
- Eğer kullanıcı öğrenci ise %5 ek indirim uygulanacak.

```
tutar = ogrenci_indirimi(tutar, ogrenci_mi)
#  Yeni bir kampanya geldiğinde mevcut kodu bozmadan yeni fonksiyon eklenir (O).
```

05

SOLID Uygulama

```python
def sepet_toplami(fiyatlar):
    toplam = 0
    for f in fiyatlar:
        toplam += f
    return toplam

def indirim_uygula(toplam):
    if toplam > 1000:
        return toplam * 0.90
    return toplam

def kdv_ekle(tutar):
    return tutar * 1.20

def kargo_ekle(tutar):
    return tutar + 50

def ogrenci_indirimi(tutar, ogrenci_mi):
    if ogrenci_mi:
        return tutar * 0.95
    return tutar


fiyatlar = [300, 400, 500]
ogrenci_mi = True

tutar = sepet_toplami(fiyatlar)
tutar = indirim_uygula(tutar)
tutar = ogrenci_indirimi(tutar, ogrenci_mi)
tutar = kdv_ekle(tutar)
tutar = kargo_ekle(tutar)

print("Ödenecek Tutar:", tutar)
```

Son...