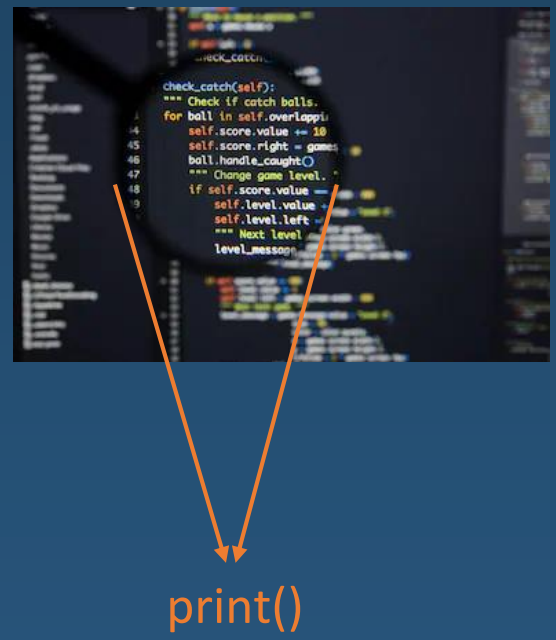
Fonksiyon

- Programlama dilleri için de bir iş ya da görevi temsil eden komutlar fonksiyon olarak anılır.
- Şu ana kadar “fonksiyon” tabirini kullanmadan pek çok fonksiyonun kullanımından bahsettik.
- “ekrana veri yazma işini” yerine getiren print()
- “ekrandan okuma işini” yerine getiren input() gibi

Fonksiyon

- Fonksiyonlar program içerisinde birden çok yerde kullanılacak karmaşık kod yığınlarının bir yerde tanımlanarak çok kez kullanımını sağlar.
- Bu sayede program içi karmaşa azaltılmış, kod tekrarının önüne geçilmiş olur.
- Ekrana veri yazmayı bir koordinat ekseni olan panel üzerinde belirli sıradaki noktaları sıra ile siyah ya da beyaz hale getirerek her bir harfi tek tek ekrana dokumak olarak düşünürsek print() işleminin çok basit olmadığı anlayabiliriz.



Fonksiyon

- Fonksiyonlar bir program içerisinde yer alan alt program parçalarıdır.
- Her bir fonksiyonun içerisinde belirli bir işi yerine getirecek şekilde yazılmış çok sayıda kod yer alır.
- Fonksiyonlar iş yapan komut dizilerinin paketlenmiş halidir.
- Program içerisinde birden çok yerde tekrar eden kod bloklarını bir fonksiyon haline getirmek program içi karmaşıklığı azaltacaktır.

Fonksiyon

- Fonksiyon çağrıları komut isminden sonra gelen ( ) sembolleri ile yapılır.
- Parantez içi fonksiyonun yerine getirdiği işe bağlı olarak boş ya da dolu olabilir.
- Parantez içerisinde yer alan verilere fonksiyonun çalışması için gerekli argümanlar ya da parametreler denir.
- Fonksiyonlar yerine getirdikleri işe bağlı olarak parametreli ya da parametresiz olabilir.

Fonksiyon

- print() fonksiyonunu ele alırsak, fonksiyonun ekrana hangi veriyi yazacağını bizim söylememiz gerekir.

  print(“merhaba”)
- Fonksiyonlar farklı sayıda parametre alabilirler. Örneğin print fonksiyonunu şu şekilde kullanabildiğimizi de hatırlayın:

  print(“merhaba”,”python”,”programlama”)
- Her bir parametre aralarına virgül konularak ayrılır.

Fonksiyon Tanımlama

- Bir fonksiyon şu şekilde tanımlanır:

  def fonksiyon_ismi(varsa parametre-ler):

  fonksiyona ait kodlar

- Fonksiyonlara isim verirken değişken adlandırma kurallarına dikkat edilmelidir.
- Fonksiyon isminin ardından parantez içerisine, fonksiyonun ihtiyaç duyduğu parametreler yazılır.
- Parametre ihtiyacı yoksa parantez içerisi boş bırakılır.

Fonksiyon Tanımlama

- Fonksiyonlar tanımlandıktan sonra çağrılmadıkları sürece işlev göstermezler.
- Program dosyası yalnız fonksiyon tanımlamaları varken çalıştırılırsa ekrana herhangi bir şey yazmaz.
- Fonksiyonun işlev göstermesi için çağrılması gerekmektedir.

Fonksiyon Tanımlama

- Parametresiz bir fonksiyon tanımı şu şekilde yapılır:

```
def mesajyaz():
    print("merhaba")
```

- Bir fonksiyonun parametre alıp alamayacağına karar verirken fonksiyonun gerçekleştireceği iş göz önüne alınmalıdır.

Fonksiyon Tanımlama

- İki sayının toplamını bulup ekrana yazacak bir fonksiyon aşağıdaki şekilde tanımlanabilir:

```
def topla(a,b):
    sonuc=a+b
    print(sonuc)
```

- Burada fonksiyon ürettiği sonucu yine kendi içerisinde ekrana yazdırmıştır.
- Fonksiyon içindeki a, b ve sonuc değişkenleri sadece fonksiyon içerisinden erişilebilir değişkenlerdir.

Fonksiyondan Değer Döndürme

- Fonksiyonlar belirli bir işi yerine getirdikten sonra ürettikleri sonucu kendi içlerinde kullanıp çalışmalarını tamamlayabilirler.
- Bir önceki örnekteki topla() fonksiyonu bu şekilde çalışmaktaydı.

```
def topla(a,b):
    sonuc=a+b
    print(sonuc)
```

```
topla(5,10)
```

Fonksiyondan Değer Döndürme

- Fonksiyon gerçekte yapması gereken iş olan “toplamanın” yanında ekrana yazma işini de yerine getiriyor.
- Fonksiyon yazarken dikkat edilmesi gereken noktalardan biri fonksiyonun asıl görevi dışında ek yüklerinin olmamasıdır.
- Burada ekrana yazma işini fonksiyonun yerine getirmemesi daha uygun olur, çünkü asıl yapılmak istenen toplama işlemidir.

Fonksiyondan Değer Döndürme

- Eğer üretilen sonuç fonksiyonun kendi içinde kullanılmayacaksa fonksiyonu çağıran yerde kullanılmak üzere, çağrılan yere gönderilmesi mümkündür.
- Bu işleme geriye değer döndürme denir ve return anahtar sözcüğü ile yapılır.

```
def topla(a,b):
    sonuc=a+b
    return sonuc
```

```
islem=topla(5,10)
print(islem)
```

Fonksiyondan Değer Döndürme

- Fonksiyon çalışırken return yazılı satıra ulaştığı anda sonlanır ve return sonucu, fonksiyonun çağrıldığı yere gönderir.
- Fonksiyonun çağrıldığı yerdeki topla(5,10) ifadesi işleminin sonucunu üzerinde tutan bir değişken gibi davranır.

```
def topla(a,b):
    sonuc=a+b
    return sonuc
```

```
islem=topla(5,10)
print(islem)
```

Fonksiyondan Değer Döndürme

- return ifadesi bir fonksiyondan geriye değer döndürmek için kullanılır.
- Fonksiyon kodları çalışırken return kodu çalıştığı zaman sonlanır.
- return deyiminin çalışmasından sonra, bu satırın altında yer alan başka kodlar varsa çalışmaz.

```
def topla(a,b):
    sonuc=a+b
    return sonuc
    print("topla fonksiyonu tamamlandı")
```

➡ Bu satır çalışmaz

Çok Sayıda Değer Döndürme

- Bir fonksiyon birden çok ara sonuç üretebilir ve bunların ayrı ayrı geri döndürülmesi gerekebilir.
- Pek çok programlama dili çoklu değer geri döndürmeyi desteklemezken Python'da bu mümkündür.

Çok Sayıda Değer Döndürme

- Kendine parametre olarak gelen iki değişkenin hem toplam hem de çarpım sonuçlarını geri döndüren bir fonksiyon tasarlayalım:

```
def IkiIslem(a,b):
    toplam=a+b
    carpim=a*b
    return toplam, carpim
```

```
s1, s2 = IkiIslem(3,4)
```

Sıralı (isimsiz) ve Sırasız (isimle) parametre çağırma

```
def SekilCiz(karakter, satirsayisi, sutunsayisi):
    for i in range(satirsayisi):
        print(karakter*sutunsayisi)

SekilCiz("*" ,5, 7)  #parametreleri sıralı (isimsiz) çağırma

SekilCiz(karakter="*", satirsayisi=5, sutunsayisi=7)  #isimle çağırma

SekilCiz(satirsayisi=5, sutunsayisi=7, karakter="*")  #isimle çağırma
```

Değişen Sayıda Parametre Alan Fonksiyonlar

- Bir fonksiyon kaç adet parametre alacak şekilde tanımlanmış ise fonksiyonu çağırırken de o kadar parametrenin sağlanması gerekir.
- Varsayılan değerli parametre kullanımı halinde bu durum değişse bile fonksiyonun iç işleyişinde tanımlamasında var olan parametre sayısı kadar parametre işleme girer.

Değişen Sayıda Parametre Alan Fonksiyonlar

```
def topla(a,b):
    sonuc=a+b
    return sonuc
```

```
topla(5,10) #doğru
topla() #hatalı
topla(2,8,12) #hatalı
```

Değişen Sayıda Parametre Alan Fonksiyonlar

- Bu durumda üç sayının toplamını almak istersek yeni bir fonksiyon, beş sayının toplamını almak istersek bir başka fonksiyon yazmak zorunda kalacağız.
- Farklı sayıda parametre alan her bir çağrım için yeni bir fonksiyon yazımı gerekli.
- Peki tek bir fonksiyon tanımlamasını farklı sayıda parametre alacak hale getirmek mümkün müdür?

Değişen Sayıda Parametre Alan Fonksiyonlar

```
def topla(*sayilar):
    sonuc=0
    for i in sayilar:
        sonuc=sonuc+i
    return sonuc
```

Bu fonksiyonun çağrımı istenen kadar parametre ile yapılabilir:

```
topla(5,10) #doğru
topla() #doğru
topla(2,8,12) #doğru
topla(2,8,12,20,25,50) #doğru
```

- Parametrenin başındaki * işaretinin görevi farklı sayılarda gelen parametreleri bir koleksiyon halinde bir araya getirmektir.