Bilişim Teknolojileri ve Yazılım Dersi 2. Dönem Çalışma Notları

1. Fonksiyon Nedir?

• Fonksiyonlar, belirli bir işi gerçekleştiren kod bloklarıdır.

• Bir kez tanımlanır ve ihtiyaç duyuldukça çağrılır.

• Örnek: Bir işlemi tekrar tekrar yapmak yerine, bir fonksiyon yazarak tekrar kullanılabilir hale getirebiliriz.

2. Fonksiyon Tanımlama ve Çağırma

Fonksiyon tanımlaması 'def' anahtar kelimesi ile yapılır.

• Örnek Kod:

```
def merhaba():
        print("Merhaba!")
•merhaba()  # Fonksiyonu çağırma
```

3. Parametreli ve Parametresiz Fonksiyonlar

Parametresiz fonksiyonlar:

Herhangi bir değer almaz, sabit işlemleri gerçekleştirir.

Parametreli fonksiyonlar:

Girdi alarak işlemlerini bu girdilere göre yapar.

Örnek Kod:

```
def selam_ver(isim):  #parametreli fonksiyon
        print(f"Merhaba, {isim}!")
selam_ver("Ali")  # ana programdan fonksiyon çağrılıyor.
```

***Parametreli fonksiyonlar dışarıdan veri alır, parametresiz fonksiyonlar sabit işlemler yapar.

Örnek Kod:

•Parametresiz:

```
def sabit_mesaj():
        print("Bu bir sabit mesaj.")
```

• Parametreli:

```
def mesaj_yazdir(mesaj):
        print(mesaj)
mesaj_yazdir(“Safranbolu”)
```

Örnek 1: Ekrana ana programdan n defa gönderilen metni yazdıran programı fonksiyon kullanarak yapınız.

```
def yaz(metin,adet):
    while adet>0:
        print(metin)
        print()
        adet=adet-1
mesaj=input("mesajı gir:")
tekrar=int(input("kaç defa yazsın:"))
yaz(mesaj,tekrar)
```

Örnek 2: Ana programdan Klavyeden girilen 2 sayının toplamını fonksiyonla hesaplayan programı yazınız?

```
def toplama(sayı1,sayı2):
    return sayı1+sayı2
say1=int(input("Birinci sayıyı giriniz="))
say2=int(input("İkinci sayıyı giriniz="))
print("iki sayının toplamı=",toplama(sayı1,sayı2)
```

Örnek 3: Ana programdan girilen sayının istenilen üs değeri kadar üssünü fonksiyonda hesaplayıp ana programa döndüren programı yazınız.

```
def üsalma(taban,tavan):
    return taban**tavan
say1=int(input("Üssünü almak istediğiniz sayıyı giriniz="))
say2=int(input("kaçıncı üssünü almak istiyorsunuz ="))
print(f"{say1}sayısının{say2}sayısına göre
üssü={üsalma(say1,say2)}")
```

Örnek 4: Klavyeden alınan kenar uzunluğuna göre karenin alanını fonksiyonla hesaplayıp, sonucu yine fonksiyonun içinde gösteren programı yazınız?(ana programda sadece fonksiyon çağrılacak)(parametresiz fonksiyon)

```
def alan():
    kenar=int(input("Karenin kenar uzunluğunu giriniz:"))
    sonuc=kenar*kenar
    print("karenin alanı=",sonuc)

alan()
```

Standart fonksiyon kümelerini ve kütüphanelerini kullanır.

1. Ön Tanımlı Fonksiyonlar

• Python'da birçok fonksiyon önceden tanımlanmıştır ve kullanıma hazırdır.

• Örnek: len(), count(), insert() gibi fonksiyonlar.

• Örnek Kod 1:

liste = [1, 2, 3, 4]

print(len(liste)) # 4 //listenin eleman sayısını verir

print(liste.count(2)) # 1 //listede count() içine yazılan değerin kaç kez tekrarlandığını verir.

liste.insert(2, 5) //listede 2. Elemandan sonra 5 rakamını ekler

print(liste) # [1, 2, 5, 3, 4]

liste.insert(4,5) //listede 4. Elemandan sonra 5 sayısını ekler

liste.append(8) //listenin en sonuna 8 sayısını ekler.

Örnek Kod2: Maaşlar adında verilen listeye aşağıdaki fonksiyonlar uygulandığında oluşacak ekran çıktılarını karşılarına yazınız.

```
maaşlar=[22000,50000,22000,75000,150000]
print(len(maaşlar))  //5 çıktısını üretir.
print(maaşlar.count(22000)) // 2 çıktısını üretir. 22000 değeri listede iki tane olduğu için.
maaşlar.insert(1,30000) // listenin 1. Elemanından sonra 30000 değerini ekler.
print(maaşlar) [22000,30000,50000,22000,75000,150000]
maaşlar.append(200000) //listenin en sonuna iki yüz bin değerini ekler
print(maaşlar) [22000,30000,50000,22000,75000,150000]
```

Örnek Soru tipleri:

a. Pythonda yaygın olarak kullanılan veri bilimi ve makine öğrenimi kütüphanelerine 4 örnek veriniz.

NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow ve PyTorch, **PyQt veya Tkinter**

b. Kütüphanelerden birini python sayfasına dahil etmeye bir örnek veriniz.

İmport numpy as np

İmport pandas as pd

İmport matplotlib.pyplot as plt

Vb….

Python, geniş bir kütüphane yelpazesine sahip, çok yönlü bir programlama dilidir. Bu kütüphaneler, Python'ı sadece temel bir dil olmaktan çıkarıp, birçok farklı alanda kullanılabilecek güçlü bir araç haline getirir. İşte Python'da sıkça kullanılan bazı kütüphaneler ve ne işe yaradıkları:

Veri Bilimi ve Makine Öğrenimi Kütüphaneleri

- **NumPy**: Bilimsel hesaplamalar için temel bir kütüphanedir. Çok boyutlu diziler ve matrisler üzerinde yüksek performanslı işlemler yapmayı sağlar.
- **Pandas**: Veri manipülasyonu ve analizi için kullanılır. Veri çerçeveleri (DataFrames) oluşturma ve bu veriler üzerinde çeşitli işlemler (temizleme, dönüştürme, birleştirme vb.) yapma imkanı sunar.
- **Matplotlib**: Veri görselleştirme için kullanılır. Çeşitli grafikler (çizgi grafikleri, dağılım grafikleri, histogramlar vb.) oluşturma imkanı sağlar.
- **Scikit-learn**: Makine öğrenimi algoritmaları (sınıflandırma, regresyon, kümeleme vb.) ve araçları içerir.
- **TensorFlow ve PyTorch**: Derin öğrenme modelleri oluşturmak ve eğitmek için kullanılan güçlü kütüphanelerdir.
- **PyQt veya Tkinter**: Grafik kullanıcı arayüzü (GUI) uygulamaları geliştirmek için kullanılan kütüphanelerdir.

Web Geliştirme Kütüphaneleri

- **Flask ve Django**: Web uygulamaları geliştirmek için kullanılan popüler framework'lerdir.
- **Requests**: HTTP istekleri göndermek için kullanılır.
- **Beautiful Soup**: HTML ve XML dosyalarını ayrıştırmak için kullanılır.

Diğer Kütüphaneler/ modüller

- **OS**: İşletim sistemi ile ilgili işlemler (dosya yönetimi, dizin işlemleri vb.) için kullanılır.
- **Sys**: Python çalışma zamanı ortamıyla ilgili bilgiler ve fonksiyonlar içerir.
- **Random**: Rastgele sayılar üretmek için kullanılır.
- **Math**: Matematiksel işlemler için fonksiyonlar içerir.
- **Datetime**: Tarih ve saat işlemleri için kullanılır.

Örnek Kullanım

Python

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


# NumPy ile rastgele sayılar oluşturma

data = np.random.randn(100)


# Pandas ile bir DataFrame oluşturma

df = pd.DataFrame({'A': data})


# Matplotlib ile histogram çizme

plt.hist(df['A'])

plt.title('Rastgele Sayıların Histogramı')

plt.xlabel('Değer')

plt.ylabel('Frekans')

plt.show()
```

NESNE YÖNELİMLİ PROGRAMLAMA

- Python, nesne yönelimli bir dildir ve bu yaklaşımda, gerçek hayattaki varlıkları

programlama dünyasına aktarmak için sınıflar (class) ve nesneler (object) kullanılır.

NESNE KAVRAMI

• Nesne, bir sınıfın örneğidir ve veriler ile bu verilere uygulanacak davranışlardan oluşur. Örneğin, bir "Araba" sınıfını ele alırsak, bu sınıftan üretilen bir nesne, bir arabayı temsil eder.

Bir nesnenin iki temel öğesi vardır:

1. Methods (Metotlar):

Nesnenin gerçekleştirebileceği davranışlardır. Metotlar, sınıfın içinde tanımlanan fonksiyonlardır ve genellikle nesnenin niteliklerini işlemek veya onlara erişmek için kullanılır.

Örneğin, "Araba" nesnesi için "hızlan", "yavaşla" veya "kornaya bas" gibi davranışlar metotlardır.

2. Attributes (Nitelikler):

Nesnenin sahip olduğu özelliklerdir. Bu özellikler, sınıf içinde tanımlanır ve bir nesnenin durumunu ifade eder. Örneğin, bir "Araba" nesnesinin rengi, markası, modeli

gibi bilgiler niteliklerdir. Python'da nitelikler, sınıf içinde tanımlanan değişkenlerdir

```python
# Sınıf tanımı
class Araba:
    # Yapıcı metot (__init__) ile niteliklerin tanımlanması
    def __init__(self, marka, model, renk):
        self.marka = marka  # Nitelik
        self.model = model  # Nitelik
        self.renk = renk    # Nitelik

    # Metot tanımı
    def bilgileri_goster(self):
        print(f"Marka: {self.marka}, Model: {self.model}, Renk: {self.renk}")

    def kornaya_bas(self):
        print(f"{self.marka} {self.model} kornaya bastı! Bip bip!")

# Nesne oluşturma
araba1 = Araba("Toyota", "Corolla", "Kırmızı")

# Niteliklere erişim
print(araba1.renk)  # "Kırmızı"

# Metot çağırma
araba1.bilgileri_goster()  # Marka: Toyota, Model: Corolla, Renk: Kırmızı
araba1.kornaya_bas()       # Toyota Corolla kornaya bastı! Bip bip!
```

Nitelikler (Attributes):

• self.marka, self.model, ve self.renk gibi sınıfın yapıcı metodu (__init__) içinde tanımlanır.

• Her nesne, kendi niteliklerine sahiptir ve bu nitelikler, nesneye özel değerler taşır.

Metotlar (Methods):

• Sınıf içinde tanımlanan fonksiyonlardır ve genellikle nesne ile ilgili işlemleri gerçekleştirir.

bilgileri_goster(self) , kornaya_bas(self)

• İlk parametreleri her zaman self olmalıdır, bu parametre, metotların nesneye erişmesini sağlar.

Nesne Yönelimli Programlama İle Yordamsal Programlama Arasındaki Fark

.•İkisi de yazılım geliştirme için kullanılan farklı yaklaşımlardır.

• İkisi arasındaki temel farklar, programların organizasyonu, veri ve davranışların ayrımı, ve çözüm yöntemlerindeki bakış açısıyla ilgilidir.

Yordamsal Programlama (Procedural Programming)

Yaklaşım

• Programlar, bir dizi talimat veya prosedürden (fonksiyonlardan) oluşur.

• Veri ve işlemler (davranışlar) birbirinden bağımsızdır.

Avantajları:

• Basit ve küçük projelerde etkili.

• Anlaması ve uygulaması kolaydır.

Dezavantajları:

• Büyük ve karmaşık projelerde kodun okunabilirliği ve yönetilebilirliği zorlaşır.

• Veri ve davranışların ayrılığı nedeniyle, bir değişiklik yapmak programın diğer bölümlerini etkileyebilir.

Nesne Yönelimli Programlama (Object-Oriented Programming - OOP)

Amaç

Gerçek hayattaki varlıkları yazılım dünyasına yansıtmak ve her bir varlık için bir sınıf (class) tanımlamak.

Veri Yönetimi

Veriler nesneye özgüdür ve nesneler dışarıdan izole edilmiştir. Bu, veri bütünlüğünü artırır.

Avantajları

Büyük ve karmaşık projelerde daha düzenli ve yeniden kullanılabilir bir yapı sunar.

Kod tekrarını önler, çünkü sınıflar bir kez yazılır ve tekrar tekrar kullanılabilir.

Kolay bakım ve genişletile bilirlik sağlar.

Dezavantajları

Küçük projelerde gereksiz yere karmaşık olabilir. OOP'nin yapısını öğrenmek başlangıçta daha zordur.

Örnek1:

Python'da OOP Örneği

Aşağıda bir **Araba** sınıfı ve ondan türetilen nesne örneği bulunmaktadır:

```
class Araba:
    def __init__(self, marka, model, yil):
        self.marka = marka
        self.model = model
        self.yil = yil

    def bilgi_goster(self):
        print(f"Araba: {self.marka} {self.model}, Yıl:
{self.yil}")
# Nesne oluşturma
araba1 = Araba("Toyota", "Corolla","2011","2015")
araba1.bilgi_goster()
```