**Python 4: Nesneye Yönelimli Programlama, Fonksiyonel Programlama, Modül Oluşturmak ve İstisnalar**

**1)Nesneye Yönelimli Programlama**

**A) Sınıflara Giriş ve Sınıf Tanımlamak:**

**Sınıf :** Benzer özellikler, ortak amaçlar taşıyan içerisinde metod ve değişkenler olan yapılardır.

class Sınıf\_İsmi():

tanımlancak değerler

ile tanımlanır.

Örneğin ;

class VeriBilimci():

print("bu bir sınıftır")

**B) Sınıf Özellikleri( Class Attributes ) :**

Sınıfın özelliklerine erişmek için sınıfı tanımladıktan sonra "Sınıf\_İsmi.özellik(fonksiyon)" tanımı kullanılır.

Sınıfın Özelliklerini değiştirmek sınıf tanımlandıktan sonra "Sınıf\_İsmi.özellik = "yeni özellik" tanımı ile yapılır

**C) Sınıf Örneklemesi :**

Genel sınıf özelliklerini barındıran alt kümeler oluşturma işlemine Sınıf Örneklemesi denir.

Örneklemadı = Sınıf\_İsmi() ile tanımlanır.

özelliklerini değiştirmek istediğimizde ;

örneklemadı.özellik.append("eklenecek\_öge") ile değiştiririz.

!!!! Örneklem üzerinde yapılan bir değişiklik tüm sınıfı etkileyebilir !!!!

Bunları olmasını engellemek için bir sonraki bölüme bak.

**D)Örnek Özellikleri :**

Sınıflarda bir örnek üzerinde değişiklilk yaptığımızda bu tüm sınıfı etkiler bunun olmasını engellemek için \_\_init\_\_ ve self komutları kullanılır. Self örneklemleri temsil etmektedir.Peki nasıl mı kullanıcaz ?

Class sınıfismi() :

Tanımlancak değerler

Def \_\_init\_\_(self):

Self.tanımlancak değer = tanımlanan iş

İle kullanılır örneğin ;

Class VeriBilimci() :

Bildiği\_diller = []

Def \_\_init\_\_(self):

Self.bildiği\_diller =[]

**E)Örnek Metodları :**

Örnekler üzerinde çalışan fonksiyonlar yazmak.

İlk önce öreneklemleri tek tek tanımlayacağız yani bir önceki nölümdeki gibi sonra yeniden def açıp fonskiyonumuzu tanımlıyoruz yani :

Class sınıfismi() :

Tanımlancak değerler

Def \_\_init\_\_(self):

Self.tanımlancak değer = tanımlanan iş

Def fonsiyonismi():

Fonksiyon

**F)Miras Yapıları (İnheritance) :**

Daha önce tanımlamış olduğumuz başka bir sınıfın özelliklerini taşıyorsa ve biz onları kullanmak istiyorsak o cllasın özelliklerini miras olarak kullanabiliriz. Bunu ise önceki tanımladığıımız sınıfın adını yeni sınıfımızın parantezine yazarak yaparız. Yani ;

Class sınıfismi1() :

Tanımlancak değerler

Def \_\_init\_\_(self):

Self.tanımlancak değer = tanımlanan iş

Class sınıfismi2(sınıfismi1) :

Tanımlancak değerler

Def \_\_init\_\_(self):

Self.tanımlancak değer = tanımlanan iş

NOT : Eğer bir sınıfta işler kolaylaşsın diye fonsiyonel olarak tanımlamak istersek şöyle yapmalıyız :

Class sınıfismi() :

Tanımlancak değerler

Def \_\_init\_\_(self,tanımlancakdeğer):

Self.tanımlancak değer = tanımlanan iş

**2.Fonksiyonel Programlama :**

1. **Fonksiyonel Programlamaya Giriş :**

Fonksiyonlar dilin baş tacıdır. Birinci sınıf nesnelerdir. Yan etkisiz fonksiyonlar yani bir girdi verildiğinde ancak çıktı ürettiğimiz bildiğimiz fonksiyonlardır. Yüksek seviye fonksiyonlar ise vektörel operasyonlarla birlikte düşünüldüğü zaman daha az emekle daha çok iş yapmamızı sağlar. Yani daha esnek ve bizi daha iyi anlayan bir programlama yaklaşımıdır.

1. **Yan Etkisiz Fonksiyonlar(Püre Functions) :**

Yan etkisiz fonksiyonlar yani bir girdi verildiğinde ancak çıktı ürettiğimiz bildiğimiz fonksiyonlardır.

Dış etmenler yani a değeri diye atadığımız bir değeri değiştrsek bile ya da b değeri eklesek bile sonucunun aynı klacağından emin olduğumuz fonksiyonlardır

Ölümcül yan etki : dışarıdan müdahale edildiğinde bozucu yan etkisi olmasına denir.

1. **İsimsiz Fonksiyonlar :**

Lambda argümanı ile isimsiz fonksiyonlar tanımlanabilir kullanım şekli :

Lambda değişken,değişken … : fonksiyon

Sorted :küçükten büyüğe sıralama için kullanılır

Sorted (liste , key = lambda değişken : fonksiyon )

1. **Vektörel Operasyonlar :**

Range komutu liste içinde gezinmek için kullanılır.

Range(0,len(a))

Çok uzun döngüler halinde yazabileceğimiz fonskiyonları daha kısa ve daha kolay bir şekilde yazma yöntemidir.

Örnek ;

İmport numpy as np

a = np.array([1,2,3,4])

b = np.array([2,3,4,5])

a\*b

1. **Map, Filter ve Reduce Fonksiyonları :**

**Map :** Verilen bir nesnenin üzerinde tanımlanacak bir fonksiyonu çalıştırmamız için kullanılır.

Örneğin;

Liste = [1,2,3,4,5]

List(map(lambda x : x+10 , liste))

**Filter :** Verilen nesne üstünde istediğimiz şekilde filitre uygulamamızı ve orjinallerini gösteren komuttur.

Örneğin;

Liste = [1,2,3,4,5,6,7,8,9,10]

List(filter(lambda x: x%2 ==0, liste))

**Reduce : :** Verilen nesne üstünde istediğimiz şekilde filitre uygulamamızı ve sonucu gösteren komuttur.

Örneğin ;

From functools import reduce

Liste = [1,2,3,4]

Reduce(lambda a,b : a+b , liste)

**3.Modül Oluşturmak Ve İstisnalar :**

**A) Modül Oluşturmak:**

Modül : Kütüphane ve ya pakette denir .Belirli amaçları yerine getirmek için bir arada bulunan fonksiyonlar topluluğudur.

Yeni bir Python sayfası açıp içine kullanacağımız fonksiyonlarımızı ve neslerlerimizi yazıyoruz ve o yeni açtığımız Python sayfası modülümüz olur. İmport komutu ile o çalışma sayfasından şuan ki kullandığımız sayfanın içine dahil etmiş oluruz. Yeni açtığımız Python sayfası içinde olan fonksiyonları 3 şekilde kullanabiliriz. Bunlar ;

1. İmport modülün adı

Modülün adı . fonksiyonun adı (değer)

1. İmport modülün adı as modülün adının kısaktması

Modülün adının kısaltması.fonksiyonun adı (değer)

1. From modülün adı import fonksiyon adı

Fonkisyonun adı (değer)

**B) Hatalar (İstisnalar ,Exceptions) :**

Hatalar genel olarak 3’e ayrılır.

1. Programcının Hataları : Basit hatalardır ve genelde Python söz dizilimi üstünde görülüp yakalanabilirler.
2. Program Hataları (Bug) : Çılgın, uğraşması zor hatalardır. Programın çalışmasına engel olan hatalardır.
3. İstisna Hataları (Exceptions) : Programda bildiğimiz bazı hatalar fakat bu hatalar gözlendiğinde bu hatalar gözlendiğinde durma çalışmaya devam et demektir.

Try :

Fonksiyon

Except hata kodu / türü :

Hatayı yakaladığı zaman ne yapması gerektiğini

Şeklinde kullanılır ve o hatayı geçersiz sayar.