PREMIUM E-JOURNAL OF SOCIAL SCIENCES

2687-5640

Yıl / Year : 2021
Cilt / Volume : 5
Sayı / Issue : 17
ss / pp : 720-727

http://dx.doi.org//pejoss.2206
Araştırma Makalesi / Research Article
Makale Geliş / Received : 27.11.2021
Yayınlama / Published : 31.12.2021

Öğr. Gör. Mustafa OF
Kocaeli Üniversitesi, Kocaeli Meslek Yüksekokulu, Başiskele/KOCAELİ
https://orcid.org/0000-0002-7924-9073

Öğr. Gör. İsmail KILIÇASLAN
Kocaeli Üniversitesi Ali Rıza Veziroğlu Meslek Yüksekokulu, Körfez/KOCAELİ
https://orcid.org/0000-0002-8443-9912

KOLAY BİR PROGRAMLAMA DİLİ OLAN PYTHON İLE NESNEYE DAYALI PROGRAMLAMA

Özet

Nesneye yönelik programlama (NYP), (Object Oriented Programming) bir yazılım programlama felsefesidir. Günümüzde birçok programlama dili tarafından desteklenmektedir. NYP yaklaşımından önce yazılan uygulamaların satır sayısı her geçen gün artmakta ve destek veya bakım kapsamındaki işler oldukça zorlaşmaktaydı. Yazılım, dinamik olarak sürekli büyüyen bir yapıya sahiptir. Bu yüzden yazılımın üretilmesinden sonraki destek için bir maliyet ayırmak gereklidir. 1960'lı yılların sonlarına doğru ortaya çıkan bu yaklaşım uzun yıllar boyunca kullanılmıştır. Günümüzde halen kullanılmaya devam edilmektedir. Yazılımların karmaşıklığını ve destek adını verdiğimiz yenilenme işlemlerini daha kolay bir duruma dönüştürmüştür. Yazılımı daha güvenli olarak geliştirmeyi sağlamıştır. Günlük hayattaki nesne mantığına dayalı bir bakış açısı ile yazılıma yaklaşan bu felsefe, daha anlaşılabilir kodların oluşmasını sağlamıştır. Ayrıca bir takım çalışması içerisinde birden fazla programcının aynı projede çalışabilmesini sağlayan tekniklere sahiptir. Aynı kodların sürekli tekrarlanmasını azaltmıştır. Yazılımları modüler bir yapıya kavuşturmuştur. Büyüyen ama karmaşık olmayan yazılım satırlarının oluşmasını sağlamıştır. Bu çalışmada, nesneye yönelik programlama kavramının önemi vurgulanacaktır. Python programlama dili üzerinde NYP tekniklerinin nasıl kullanıldığına dair metotlar açıklanacaktır.

Keywords: Python, Nyp, Sınıf, Nesne

OBJECT-ORIENTED PROGRAMMING WITH PYTHON, AN EASY PROGRAMMING LANGUAGE

Abstract

Object Oriented Programming (Object Oriented Programming) is a software programming philosophy. Today it is supported by many programming languages. The number of lines of sotware applications written before the NYP approach was increasing day by day and the maintenance under the support was becoming more difficult. The software has a dynamically growing structure. Therefore, it is necessary to allocate a cost for the support after the production of the software. This approach, which emerged in the late 1960s, has been used for many years. It is still used today. It has made the complexity of the software and the renewal processes we call support easier. It has enabled the software to be developed more safely. Approaching to the software with an approach based on object logic in daily life, this philosophy has led to the formation of more understandable codes. It also has techniques that enable multiple programmers to work on the same project within a team work. It has reduced the repeated repetition of the same codes. The software has a modular structure. It has led to the growing but

uncomplicated lines of software. In this study, the importance of object oriented programming concept will be emphasized. Methods of using NYP techniques on Python programming language will be explained.

Keywords: Python, Oop, Class, Object

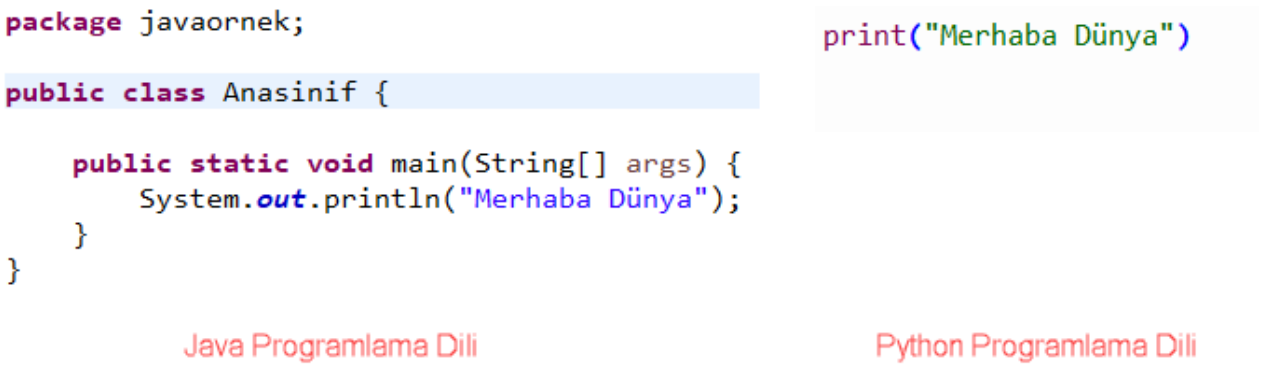
1. GİRİŞ

Bilgisayarın donanım ve yazılımdan meydana geldiğini bir çoğumuz biliriz. Bilgisayarın yazılım olmadan 2 + 2'nin dört ettiğini bile hesaplayamayacağı bilinen temel bir gerçektir. Yazılım kısaca, bilgisayarın temel alt yapısına dayalı olarak yine birer yazılım olan programlama dilleri vasıtasıyla meydana getirilmiş makine dili adını verdiğimiz (İkili sayı sistemi) yapıya dönüştürülmüş uygulamalardır. Bilgisayarın temelinde elektronik bir alt yapı bulunduğu için bu yapının sayılara dönüşmesini sağlayan ikili sayı sistemidir. Eğer iki uç arasında bir gerilim varsa 1 değeri, yoksa 0 değeri elde edilir. Bu değerler farklı sayı sistemleri vb. yapılarla kullanabileceğimiz bir bilgisayar ara yüzüne dönüşmüş olur. Sanal dünyada kullandığımız çok sayıda yazılım bulunmaktadır. Yazılımların mutfağında ise programlama dilleri ve programcılar bulunmaktadır. Programlama dili, kısaca bilgisayarın temel alt yapısına dayalı olarak kendisine has ifade veya komutlarla programcının yapmak istediklerini bilgisayara yaptırabilmeyi sağlayan yazılım topluluğudur. Birçok işlemin bilgisayar veya benzeri aygıtlar üzerinden yapıldığı günümüzde programcı veya yazılım uzmanı ihtiyacı her geçen gün artmaktadır.

1991 yılında Guido Van Rossum tarafından ilk ortaya çıkarılan, Python programlama dili oldukça popüler bir programlama dilidir. Python, sunucu taraflı web uygulamaları, sistem betikleri (Script), matematiksel uygulamalar, veri tabanı uygulamaları, dosya uygulamaları veya genel amaçlı birçok ihtiyaca cevap verebilen öğrenilmesi oldukça kolay olan bir programlama dilidir. Python'u öğrenmek ve anlamak kolaydır. Eğer bir programlama dili geçmişinden geliyorsanız, bu dilin oldukça kolay ve düzenli olduğunu fark edeceksiniz. Python'daki birçok işlem diğer programlama dillerine göre oldukça kolaydır. (BASHIN, s:3)

Python, yazım kurallarının kolay olması, günlük hayatta kullanılan konuşma diline (İngilizce) yakın olması gibi özellikleri ile programlama bilgisi az olan birçok kişinin bile ilgisini çekmeyi başarmıştır. İhtiyaç duyulan paketlerin veya kitaplıkların geniş ve herkese açık bir ağda bulunması (http://pypi.org) gibi öne çıkan özellikleri ile en basit programdan en karmaşık programa kadar birçok uygulama geliştirme imkânı sunmuştur. Programlama dilleri arasında en kolay öğrenilebilen dillerden birisi olarak Python yerini almıştır. (SU, s:4)

Aşağıdaki görsele baktığımızda programlama diline ilk başlayanların ilk uygulaması olan "Merhaba Dünya" (Hello World) uygulamasının iki farklı dilde nasıl değişik olduğu hemen fark edilecektir.



Şekil 1. "Merhaba Dünya" uygulamasının iki farklı dilde oluşturulması

Stackoverflow adlı yazılım geliştiricilerin kodlarını paylaştığı sitenin 2020 yılında yapmış olduğu geliştiricilere yönelik yapmış olduğu ankete göre Pyhton, en çok tercih edilen diller arasında üçüncülüğe yerleşmiştir. Sıralama şöyledir;

1. Rust (%86 Kullanım)
2. TypeScript (%67)
3. Python (%66)
4. Kotlin (%41)

Python, güçlü bir programlama dilidir. Bu yüzden birçok temel yazılım felsefesine alt yapı sağlamaktadır. Nesneye yönelik programlama tekniklerini kullanarak daha yapısal uygulamalar geliştirmek mümkündür. Bu çalışmada Nyp teknikleri üzerinde durulacak ve Python ile Nyp teknikleri arasındaki bağ örneklerle anlatılacaktır.

2. NESNEYE YÖNELİK PROGRAMLAMA

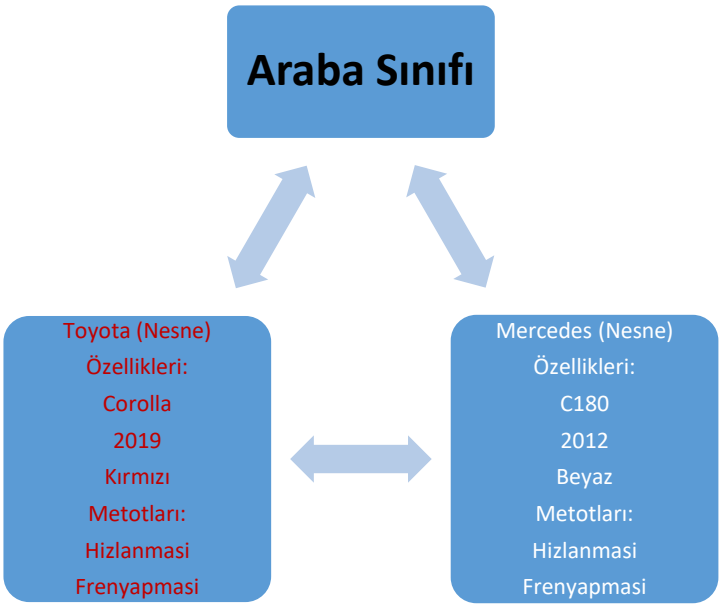
Nesneye yönelik programlama (OOP), programcıların bir veri yapısına, veri türüne ve ayrıca veri yapısına uygulanabilecek işlem türlerini (fonksiyonlarını) tanımladığı bir tür programlama yaklaşımı veya yazılım tasarımı anlamına gelir. Bu sayede, veri yapısı hem veri hem de fonksiyon içeren bir nesne haline gelir. Ek olarak, programcılar bir nesne ile bir diğeri arasında ilişkiler oluşturabilirler. Örneğin, nesneler diğer nesnelerden özellikleri devralabilir. Kodların birden çok tekrarı azaltılmış olur. Bu sayede satır sayısı azalmış ve kontrol edilebilirlik artmış olur. NYP teknikleri ile uygulama yazarken bazı temel bilgilere sahip olunması gereklidir.

2.1. Nyp'nin Temel Kavramları

Nyp'nin temelini sınıf (class) oluşturmaktadır. Temel olarak dört ilkeye sahiptir. Kapsülleme (Encapsulation), Soyutlama (Abstraction), Kalıtım (Inheritance), Çok Biçimlilik (Polymorphism).

Sınıf (Class): NYP'nin temelini temsil eder. Bir nesne şablonudur. Sınıf, kendisine ait olan farklı nesnelerin tüm ortak özelliklerini tanımlar. Günlük hayattaki karşılaştığımız nesnelerle yazılımdaki nesneler arasında bir benzetme kullanılmıştır. Sınıf, soyut bir kavramdır. Fakat sınıftan çoğalan nesne somut bir kavramdır. Bir sınıfın temel olarak özellikleri (Değişkenler) ve metotları (Fonksiyonlar) bulunur. Özellikler, sınıfı şekil olarak ortaya çıkarır. Metotlar sınıfa bir işlev katar. Örneğin bir Araba sınıfına ait model, yıl, renk vb. özellikleri olabilir. Hızlanması, Fren yapması ise bir fonksiyon olarak ele alınmaktadır. Sınıflardan türeyen nesneler, türemiş oldukları sınıfın tüm üyelerini almış olurlar. Bu sayede kod tekrarlaması azaltılmış olur. Sınıftan istenildiği kadar nesne türetilir. Sınıflar, referans türler grubuna üye oldukları için belleğin öbek (Heap) kısmında saklanırlar. Belleğin yığın (Stack) kısmında ise sadece adresleri saklanır. (BASHIN, s:168)

pejoss.editor@gmail.com (ISSN:2687-5640)



Şekil 2. Araba sınıfı ve nesneleri

Kapsülleme (Encapsulation): Herhangi bir nesnenin metotlarını, verilerini diğer nesnelerden gizleyerek ve aynı zamanda erişimini sınırlandırarak hatalı kullanımdan koruyan bir kullanım şeklidir. Kapsülleme bir veri gizleme tekniğidir. Sınıfa ait bir üyenin erişimi temel olarak üç şekilde yapılır. Public: Herkese açık. Private: Sadece o sınıfa özel. Protected: Miras alınan ve miras almış sınıfa özel. Public dışındaki kullanımlarda bir kapsülleme söz konusudur. Daha dikkatli ve güvenli bir kod yazımı meydana gelir.

Soyutlama (Abstraction): Nesnelerin ve prosedürlerin ortak özelliklerini seçme (çıkarma) işlemi. Birden fazla programcının ortak çalışma sürecinde temel bir yapı üzerinden hareket edilmesini kolaylaştırır. Temelde karar verilen metotların boş olarak tanımları oluşturulur. Farklı programcılar, aynı metot ismini kendilerine göre geliştirebilirler. Bu sayede çoklu yazılım geliştirme kolaylaşmış olur.

Kalıtım (Inheritance): Nesneye yönelik programlamada en önemli kavramlardan biri mirastır. Kalıtım, bir sınıfı başka bir sınıf olarak tanımlamamızı sağlar. Bu aynı zamanda kodları tekrar kullanmak için bir kolaylık sağlar ve uygulama süresini hızlandırır. Örneğin Şekiller isimli bir sınıfa ait olan bazı özellik ve metotları farklı bir sınıf olan Dörtgen sınıfında kullanmak mümkündür. Bu sayede Dörtgen sınıfına yeni üyeler (Özellik ve Metot) eklenebilir ve Şekiller sınıfının üyelerinden de yararlanılmış olur.

Çok Biçimlilik (Polymorphism): Polimorfizm kelimesi birçok biçime sahip olmak anlamına gelir. Nesneye yönelik programlama yaklaşımında, polimorfizm genellikle "bir arayüz, çoklu fonksiyonlar" olarak ifade edilir. Polimorfizm statik veya dinamik olabilir. Statik polimorfizmde, bir işleve verilen cevap derleme zamanında belirlenir. Dinamik polimorfizmde çalışma zamanında karar verilir. Kalıtım ilkesi ile yakından ilgilidir.

Kalıtım, kodun yeniden kullanılabilirliğine izin verir. Polimorfizm ise, farklı formda bir fonksiyonun ortaya çıkması demektir. Kalıtım ile polimorfizm arasındaki temel fark, kalıtımın halihazırda var olan kodun bir programda tekrar kullanılmasına izin vermesidir. Polimorfizm, hangi fonksiyonun hangi formun çağrılacağına dinamik olarak karar vermek için bir mekanizma sağlar.

Kalıtım ve polimorfizm, miras kavramını uygulayan sınıflara uygulandığı için birbiriyle ilişkili kavramlardır.

Tablo 1. Kalıtım ve Polimorfizm arasındaki fark

Kalıtım (Inheritance)	Çok Biçimlilik (Polymorphism)
Kalıtım, sınıflara uygulanır.	Polimorfizm ise metotlara/fonksiyonlara uygulanır.
Kalıtım, üyelerini daha önceden var olan bir sınıftan türeten yeni bir sınıf oluşturarak meydana getirir.	Polimorfizm, çoklu formlarda tanımlanabilen bir ara yüzdür.
Kalıtım, türetilmiş bir sınıfın temel sınıfta tanımlanan öğeleri ve metotları kullanmasına izin verdiği için, türetilmiş sınıfın, bu öğeleri veya metodu tekrar tanımlamasına gerek duymaz, böylece kodun yeniden kullanılabilirliğini arttırdığını ve dolayısıyla kodun uzunluğunu azalttığını söyleyebiliriz.	Polimorfizm, bir nesnenin hem derleme zamanında hem de çalışma zamanında ne tür bir metodu çağırmak istediğine karar vermesini mümkün kılar.
'Masa' sınıfı, yapı itibariyle 'mobilya' sınıfının özelliklerini devralabilir.	'calisma_masasi' sınıfı 'rengi_ayarla()' metoduna sahip olabilir ve 'yemek_masasi' sınıfı da 'rengi_ayarla()' metoduna sahip olabilir. Böylece çağrılacak rengi_ayarla() metodunun hangi biçimine hem derleme zamanı (Compile Time) hem de çalışma zamanında (Run Time) karar verilebilir.
Kalıtım, tek kalıtım, çoklu kalıtım, çok düzeyli kalıtım, hiyerarşik kalıtım ve hibrit kalıtım olarak sınıflandırılabilir.	Polimorfizm metot aşırı yükleme (Overloading) ve metot baskın gelme (Overriding) olarak sınıflandırılır.

3. PYTHON PROGRAMLAMA DİLİNDE NYP KULLANIMI

Güçlü bir programlama dili olan Python, Nyp tekniklerine sahiptir. Diğer dillerden farklı olarak daha kolay bir kullanımı bulunmaktadır. Nyp'nin getirmiş olduğu tüm artı özelliklere sahip olan bir programlama dilidir.

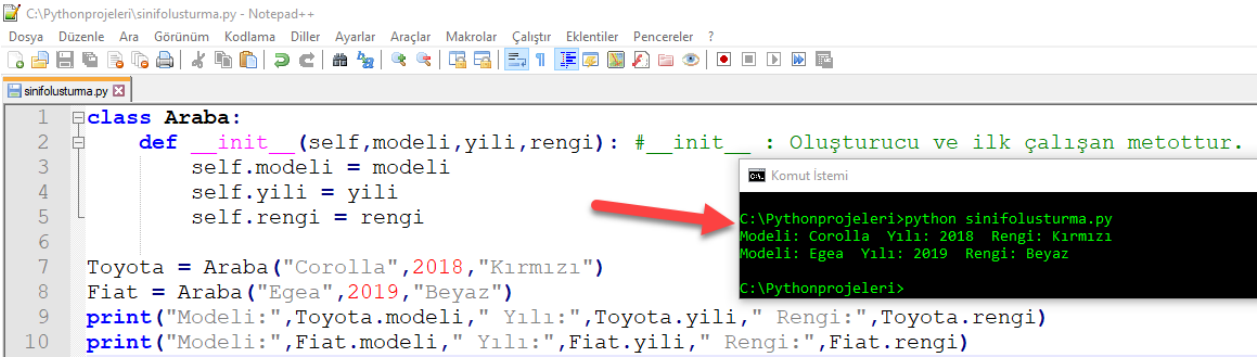
Uygulamaları çalıştırabilmek için öncelikle Python.org adresindeki "Download" bölümünden işletim sistemine uygun olan derleyici/yorumlayıcı paketinin indirilmesi gereklidir. Bu çalışmada işletim sistemi olarak Windows 10 kullanılmıştır. Editör olarak PyCharm editörü kullanılabilir. Ayrıca ücretsiz olarak indirilebilen editörler de kullanılabilir. Örneğin bunlardan biri de Thonny Python geliştirme editörü uygulamasıdır. Py uzantılı olan bir Python programını çalıştırmak için en düşük özellikli bir editör bile yeterlidir. Bu çalışmada Notepad++ editörü ile uygulamalar geliştirilmiştir. Python paketlerinin kurulumunu yaptıktan sonra aşağıdaki gibi Windows işletim sistemi komut satırında sürüm kontrolü yapılabilir.

C:\ python --version
Python 3.10.0 (Sürüm bilgisi görünürse paketler yüklüdür.)

Örnek uygulamaların kodları ve çalışmış halleri aşağıda gösterilecektir.

Sınıf Oluşturma ve kullanımı:

pejoss.editor@gmail.com (ISSN:2687-5640)



Şekil 3. Araba sınıfının oluşturulması ve kullanılması

__init__ fonksiyonu, oluşturucu metottur. Nesne ilk oluştuğu anda çalışan metottur. Nesne adını yazdıktan sonra sınıf adını ve gerekli ise parametrelerini yazmak yeterlidir. Nesne oluşturulmuş olur.

VWPassat = Araba("Passat",2011,"Gri")

"self" isimli parametre, çalışılan sınıfı temsil eder. Diğer dillerdeki "this" kelimesinin benzeridir.

self.isim = isim

Oluşturulan nesneyi silmek için "del" metodu kullanılır. Bu metot, nesneyi bellekten atacaktır.

del Toyota

Sınıf içeriği boş olmamalıdır. Ancak herhangi bir içeriği yoksa "pass" ifadesi kullanılır.

```
class Calisan:
    pass
```

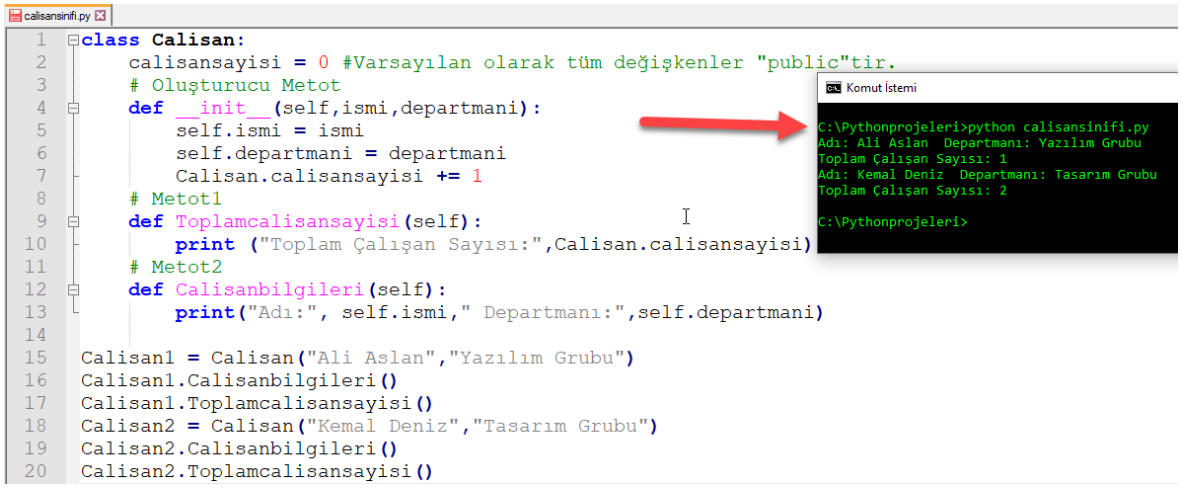
Yukarıdaki örnekte diğer dillerde bulunan public, private ve protected erişim belirleyicilerinden herhangi biri kullanılmamıştır. Bir Python sınıfındaki tüm üyeler varsayılan olarak herkese açıktır. Herhangi bir üyeye sınıf ortamının dışından erişilebilir. Yani public'tir.

Python'un bir değişken değişkeni korumalı (protected) yapma kuralı, ona bir "_" ön eki eklemektir. Bu, bir alt sınıftan gelmediği sürece erişilmesini engeller.

self._isim = isim
self._numara = numara

Benzer şekilde, bir değişken ismine önceden eklenmiş bir çift alt çizgi "__" onu özel (private) kılar. Sınıf dışından erişilmeme konusunda güçlü bir sınırlama koyar.

self.__ogrenciadi = ogrenciadi
self.__bolumu = bolumu



Şekil 4. Sınıf ve metot oluşturma

Kalıtım:
Kalıtım, tüm metotları ve özellikleri başka bir sınıftan miras alan bir sınıf tanımlamamızı sağlar.

Ana sınıf, temel sınıf olarak da adlandırılan miras alınan sınıftır.

Çocuk sınıf ise ana sınıftan üyeleri devralan sınıftır.

sinifkalitimi.py

```
class Calisan: # Ana Sınıf
    def __init__(self, ismi, departmani):
        self.ismi = ismi
        self.departmani = departmani

    def Bilgiyaz(self):
        print("İsmi :", self.ismi, " Çalıştığı Birim:", self.departmani)

class Isci(Calisan): # Calisan sınıfından miras alınıyor. Çocuk Sınıf
    toplamcalisma = 8

    def Calismasaati(self):
        print("Bu iş yerinden en fazla çalışma saati:",self.toplamcalisma," saattir.")

class Memur(Calisan): # Calisan sınıfından miras alınıyor. Çocuk Sınıf
    pass # İçeriği olmadığı için pass kullanılıyor

Isci1 = Isci("Ali Aslan","Kaynak Bölümü")
Isci1.Bilgiyaz()
Isci2 = Isci("Ahmet Deniz","Motor Bölümü")
Isci2.Bilgiyaz()
Isci2.Calismasaati()
Memur1 = Memur("Ayşe Ak","Muhasebe")
Memur1.Bilgiyaz()
```

Komut İstemi

```
C:\Pythonprojeleri>python sinifkalitimi.py
İsmi : Ali Aslan  Çalıştığı Birim: Kaynak Bölümü
İsmi : Ahmet Deniz  Çalıştığı Birim: Motor Bölümü
Bu iş yerinden en fazla çalışma saati: 8  saattir.
İsmi : Ayşe Ak  Çalıştığı Birim: Muhasebe

C:\Pythonprojeleri>
```

Şekil 5. Üst sınıftan miras alma

super() isimli metot yardımı ile ana sınıfın adını kullanmak zorunda değiliz. Otomatik olarak metotları ve özellikleri üst öğesinden miras alır. Çocuk sınıfın oluşturucu metodunda üst sınıfın metotlarını çağırırken üst sınıfın ismi yerine super() metodu kullanılabilir.

supermetodu.py

```
class Sekil: #Ana sınıf
    def __init__(self, genislik, yukseklik):
        self.genislik = genislik
        self.yukseklik = yukseklik

    def Bilgiyaz(self):
        print("Genişlik:", self.genislik, " Yükseklik:", self.yukseklik)

    def Alanibul(self):
        alan = self.genislik * self.yukseklik
        print("Alanı:", alan, " metrekare")

class Dortgen(Sekil): #Çocuk sınıf
    def __init__(self, genislik, yukseklik):
        super().__init__(genislik,yukseklik) #Üst sınıfın metodu çağrılıyor

Dortgen1 = Dortgen(5,10)
Dortgen1.Bilgiyaz()
Dortgen1.Alanibul()
```

Komut İstemi

```
C:\Pythonprojeleri>python supermetodu.py
Genişlik: 5  Yükseklik: 10
Alanı: 50  metrekare

C:\Pythonprojeleri>
```

Şekil 6. super() metodunun kullanımı

SONUÇ

Bir programlama dilini öğrenmek artık bir yabancı dil öğrenmek kadar ihtiyaç haline gelmiştir. Detaylı yardım içeriklerinin bulunduğu İnternet ortamında bunu yapabilmek hiç de zor değildir artık. Yabancı dil öğrenirken en çok çekilen sıkıntı gramer kurallarına ağırlık verilmesidir. Ne kadar çok gramer kuralına bağlıysanız o kadar zor yabancı dil konuşma yeteneğine sahip olursunuz. Programlama dilleri

pejoss.editor@gmail.com (ISSN:2687-5640)

için de aynı şey söylenebilir. Katı kuralları olan ve çalışmayan yazılımlar, programlama dili öğrenmeye olan isteği köreltmektedir. Bu açıdan Python dili öğrenilmesi kolay bir programlama dilidir. Yardım kaynakları bol, kaynak kodları açık, kütüphanesi büyüyen bir dildir. Nesne yönelimli programlama tekniklerini kullanarak yazılan her bir uygulama ileriye dönük ve büyüyen bir uygulama haline gelebilecektir. Python, Nyp konusunda oldukça esnek bir yapıya sahiptir. Bu çalışmada sonuç olarak dışarıdan oldukça zor görünen Nyp yaklaşımlı bir uygulamanın aslında o kadar da zor olmadığı örneklerle ispat edilmiştir. Belli başlı konulara riayet edildiği sürece Nyp yaklaşımlı temel bir Python uygulamasını yazmak için iki dakika bile yeterlidir. Bir ülkedeki eğitim harcamalarının yabancı lisan öğrenimine harcanan oranı ile programlama dilleri için ayrılanı kıyasladığımızda farkın oldukça yüksek olduğu göze çarpacaktır. En azından her bir üniversite öğrencisi Python dili ile bazı temel işlemleri yapabilecek kadar bilgiye sahip olması gerekmektedir. Bugünün öğrencileri, yarının işletmelerde çalışanları olacaktır. Özellikle iş hayatında çalışma hayatının ayrılmaz parçası olan bilgisayardan daha fazla faydalanacaklardır. Programlama dilini öğrenirken analitik düşünme yeteneğinin çok fazla kullanıldığı bir gerçektir. Günlük hayata da yansıyan bu sonuç özellikle gençlerimizin ilerideki hayatlarını da şekillendirebileceğini rahatlıkla söyleyebiliriz.

KAYNAKÇA

BASHIN H. (2019), Python Basics, Boston- U.S.A., Mercury Learning and Information

SU G. (2020), Python Öğreniyorum, İstanbul, Kodlab Yayınları,

Difference Between Inheritance and Polymorphism, https://techdifferences.com/difference-between-inheritance-and-polymorphism.html, (Erişim Tarihi: 01.10.2021)

OOP – Object Oriented Programming, https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html, (Erişim Tarihi: 01.12.2021)

Stackoverflow Developer Survey Results 2020, https://stackoverflow.blog/2020/05/27/2020-stack-overflow-developer-survey-results/, (Erişim Tarihi: 06.12.2021)

Polymorphism, https://www.tutorialspoint.com/csharp/csharp_polymorphism.htm, (Erişim Tarihi: 01.12.2021)

Python Tutorial, https://www.w3schools.com/python/python_classes.asp, (Erişim Tarihi: 01.10.2021)

Python Tutorial, https://www.tutorialspoint.com/python/index.htm, (Erişim Tarihi: 02.12.2021)

PyCharm, https://www.jetbrains.com/pycharm/, (Erişim Tarihi: 01.12.2021)

Thonny Python Editor, https://thonny.org/ (Erişim Tarihi: 01.12.2021)