GİKAL

Sayı 2
ŞUBAT 2022

TEKNOLOJİ VE PROJELER

GÖZTEPE İHSAN KURŞUNOĞLU ANADOLU LİSESİ

- ERASMUS PROJELERİMİZ
- TASARIM BECERİ ATÖLYRLERİMİZ
- PYHTON PROGRAMLAMAYA GİRİŞ EĞİTİMİ

İçindekiler

Yayın Kurulu

Ramazan ÇOBAN

Kadir AKILLI

Etem DURSUN

Python içerikleri hazırlanırken MEB Herkes İçin Python Programlama Dili adlı kitaptan faydalanılmıştır.

©Tüm hakları saklıdır.

Göztepe İhsan Kurşunoğlu Anadolu Lisesi izni olmadan çoğaltılamaz.

TARİHÇE

Okulumuz merhum Osman GÖRENER'in hazineye bağışladığı arsa üzerine Millî Eğitim Bakanlığınca 1964 yılında 12 derslik, 1 laboratuvar ve 1 kapalı spor salonu inşa edilmiştir. 1965-1966 yılı başında 20 Eylül 1965 Pazartesi sabahı törenle açılmıştır.

Okulumuz 1993-1995 öğretim yılında ortaokul öğretiminde son mezunlarını vermiştir.

1994-1995 öğretim yılında okulumuza ek bina yaptıran Sayın İhsan Kurşunoğlu'nun adı verilerek okulumuzun adı Göztepe İhsan Kurşunoğlu Lisesi olarak değişmiş, aynı yıl ''Yabancı Dil Ağırlıklı Lise'' olarak faaliyetine devam etmiştir.

2005-2006 Eğitim-Öğretim yılında Millî Eğitim Bakanlığı'nın 21/06/2005 tarih ve 420/03 sayılı komisyon kararıyla Anadolu Lisesi olmuştur.

2011-2012 Eğitim-Öğretim yılında okulumuz İSMEP kapsamında Şubat 2012'de yıkılmış olup her türlü donanıma sahip olarak eğitim öğretime 01 Nisan 2015 tarihinden itibaren kendi binasında başlamıştır.

2018-2019 öğretim yılında okulumuz **proje okulu** olmuştur.

2019-2020 öğretim yılında okulumuzda 7 Atölyeden oluşan (Ahşap, Ebru, Ses Kayıt Merkezi, Müzik, Robotik, Elektronik, Hat ve Tezhip) **Tasarım ve Beceri Atölyeleri** kurulmuştur.

2020-2021 tarihi itibarı ile Erasmus projeleri, Kardeş okul projeleri, Tübitak projeleri gibi projelerin yer aldığı toplam 96 proje yürütülmüştür ve yeni projelerimiz de devam etmektedir.

Okulumuzda 2021-2022 Eğitim öğretim yılı itibarı ile **Almanca Hazırlık** sınıfları açılmış ve okulumuzun birinci yabancı dili Almanca olmuştur.

Sevgili Gençler;

İhsan Kurşunoğlu Anadolu Lisesi öğrencileriyle, okulumuzun dergisi sayfalarında buluşmanın mutluluğunu ve haklı gururunu yaşıyoruz. Bilinçli bir neslin aynası olmak için özverilerimizi esirgemeyeceğiz. Zihni açık, algıları açık, hayata pozitif bakan bilinçli bir gençlik yetiştirmek amacı ile merhaba demenin mutluluğunu yaşıyoruz. Gayemiz, siz geleceğimizin teminatı gençlerle yeni ve büyük başarılara imza atmak.

Zaman bendedir ve mekân bana emanettir şuurunda; kökü ezelde ve dalı ebedide bir düşüncenin aşkına, vecdine, diyalektiğine, estetiğine, irfanına, idrakine sahip bir gençliğe ışık tutmak ve insan haklarına son derece saygılı, milli ve manevi değerlerine bağlı onları hayatla bütünleşmiş bir biçimde yetiştirmek en büyük hedefimiz.

Misyonumuz; ülkemizin maddi ve manevi değerlerini evrensel değerlerle bütünleştirebilmiş, kendini gerçekleştirmiş, ahlâklı, sosyal insanlar yetiştirmek; düşünen, konuşan, tartışan ama aynı zamanda üreten, sevgiye ve barışa gönülden inanmış bir gençlik yetiştirmek...

Sevgi ve saygılarımla hepinizi selamlıyorum.

Ramazan ÇOBAN
Okul Müdürü

Yorum Satırları, Değişkenler, Veri Tipleri, Operatörler

Yorum satırları, Python yorumlayıcısı tarafından dikkate alınmayan ve yorumlanmayan ifadelerdir. Python'da yorum satırları genel olarak aşağıdaki işlemler için kullanılır:

- Bir hatırlatma eklemek
- Program veya kodlarla ilgili bir açıklama yapmak
- Kullanılmayan bir kod satırını pasif hale getirmek
- Süsleme yapmak

Bu tür açıklama satırları kodun başkaları tarafından daha iyi anlaşılmasını sağlar. Python'da tek satırlık açıklama için "#" işareti kullanılır. "#" işareti kullandığınızda o satırdaki metin kod olarak işlenmez.

Örnek 1

Aşağıdaki yorum satırları Python tarafından dikkate alınmamaktadır. Bu nedenle sadece "print()" komutu çalışır.

```
#Bu kod ekrana yazı yazılmasını sağlamaktadır.
print ("Konu: Yorum satırlarını kullanma")
#Her satırın başına # işareti eklenerek
#alt alta yorum satırları oluşturulabilir.
Konu: Yorum satırlarını kullanma
```

Örnek 2

Yorum satırlarını kod satırının devamında aynı satırda kullanabilirsiniz. Bu kullanımda önce kod gelir, devamında yorum satırı "#" işareti ile başlar (öndeki kod çalışır), daha sonra satır sonuna kadar yorum satırı olarak dikkate alınmaz.

```
print (2+3) # Bu kod satırı ekrana 2 sayının toplamını yazar.
5
```

Her satırın başına "#" işareti ekleyerek alt alta yorum satırları oluşturabilirsiniz. Yorum satırlarında özel karakterler etkisizdir.

Örnek 3

Birden fazla yorum satırı kullanılacaksa yorumlar üçlü tek tırnak veya üçlü çift tırnak blokları arasına yazılır.

```
'''Python'da birden fazla açıklama satırı kullanmak için üçlü tek tırnak veya çift
tırnak kullanılır. Açıklama satırını bitirmek için aynı işaretler kullanılır.'''
'Python'da birden fazla\naçıklama satırı\nkullanmak için üçlü tek tırnak veya çift\n
tırnak kullanılır. Açıklama satırını bitirmek için \naynı şekilde kullanılır.'
```

Konsol çıktısında açıklamalar birden fazla satırda olduğu için satır sonlarına \n kaçış dizisi karakterini konsol otomatik olarak eklemiştir. Aynı yorum satırını """yorum satırları""" üçlü çift tırnak kullanarak da yapılabilir. Yorum satırları içinde kaçış dizisi karakterlerinin de çalışmadığı unutulmamalıdır.

Örnek 4

Yorum satırlarını Python yorumlayıcı dikkate almaz. Ancak aşağıdaki kod satırları yorum satırı olarak değerlendirilmez.

```
#!/usr/bin/env python3 veya #!c:/Python/python.exe
# -*- coding: utf-8 -*-
"#!/usr/bin/env python3" kodu Python 3 yorumlayıcısının Linux için dosya konumunu
belirtir.
"#!c:/Python/python.exe" kodu Python 3 yorumlayıcısının Windows için dosya konumunu
belirtir.
"# -*- coding: utf-8 -*- " Kullanacağınız karakter kodlamasını belirtmek için
kullanılır. utf-8 Türkçe alfabeyi de destekleyen bir karakter kodlama sistemidir.
```

Örnek 5

Yorum satırları süsleme amacıyla da kullanılabilir. Bazı program dosyalarında aşağıdaki gibi süslü açıklamalar, etiketler görülebilir.

```
'''
#########################################################################
#########################################################################
#                        Python Öğreniyorum                             ##
#                             Python 3                                  ##
#########################################################################
#########################################################################
'''
```

Kod yazarken sadece sabit değerler üzerinden işlemler yapılmaz. Kullanıcıdan veya başka kaynaklardan veri alınması gerekir. Örneğin, kullanıcıya ismiyle "merhaba" diyeceğimiz bir kod yazmak istersek her kullanıcıdan ismini girdi olarak almamız gerekir. İşlem yapabilmek için bu girdiler (değerler) bellekte tutulmalıdır. Bu girdileri depolamak amacıyla bellekte belirli bir yer ayrılması gerekir. Bir değişken tanımlandığında yorumlayıcı, veri türüne bağlı olarak bellekte yer ayırır ve ayrılan bölümde hangi türden verilerin saklanabileceğini belirler. Değişkenler bir isim, sayı veya farklı türdeki bir veri için bellekte ayrılan bu yeri temsil eder. Değişkenlere farklı veri tiplerinde değerler atanabilir. Değer atama ile (bir değişkeni bir değere eşleyerek) tam sayı, ondalık sayı, dizi veya karakter dizisi türünde değerler değişkenlerde tutulabilir. Python, bu konuda çok esnektir. Python'da değişkenlerinin veri tiplerini açıkça bildirmeye gerek yoktur. Aynı değişken farklı veri tiplerinde değerler alabilir. Aynı değişkene önce sayı, sonra bir metin daha sonra başka türde bir değer atanabilir. Bir değişkene değer atandığında veri tipi otomatik olarak tanımlanır. Eşittir operatörü "=" değişkenlere değer atamak için kullanılır. "=" Operatörünün solunda değişkenin adı ve "=" operatörünün sağında ise bu değişkene atanacak değer yer alır.

Örnek 6

Değişken oluşturma ve değer atamaya ilişkin örnekler:

```
#sayi1 değişkenine 5 sayısı atandı.
sayi1=5
print('Değişkenin içindeki sayı: ', sayi1)
sayi1=10
print('Değişkenin içindeki sayı: ', sayi1, 'oldu')
sayi1='Murat'
print ('Değişkenin içindeki değer: ', sayi1, 'oldu')
sayi1=10.5
print ('Değişkenin içindeki sayı: ', sayi1, 'oldu')
Değişkenin içindeki sayı:  5
Değişkenin içindeki sayı:  10 oldu
Değişkenin içindeki değer:  Murat oldu
Değişkenin içindeki sayı:  10.5 oldu
```

Örnekte aynı değişkene hem karakter dizisi (string) hem de sayısal değerler atanmıştır.

Bir değişkene değer atanırken birden fazla yöntem kullanılabilir. Örnekte buna ilişkin kodlar verilmiştir.

Örnek 7

Aşağıda 3 değişkene de tek satırda 1 değeri atanmıştır.

```
a = b = c = 1
print ('1. sayı=', a)
print ('2. sayı=', b)
print ('3. sayı=', c)
1. sayı= 1
2. sayı= 1
3. sayı= 1
```

Örnek 8

Değişkenler aralarına virgül eklenerek yan yana yazılır. Değerleri de aynı sıralama ile karşılarına yazılır.

```
adi, soyadi, yasi='Canan', 'DAĞDEVİREN', 34
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
Adı= Canan
Soyadı= DAĞDEVİREN
Yaşı= 34
```

Örnek 9

Değişkenlere değer atamak için başka bir yöntem aralarına noktalı virgül ";" ekleyerek değişken - değer ikilileri şeklinde yazmaktır.

```
adi='Aziz'; soyadi='SANCAR'; yasi=73
print ("Adı=", adi)
print ("Soyadı=", soyadi,)
print ("Yaşı=", yasi)
Adı= Aziz
Soyadı= SANCAR
Yaşı= 73
```

Örnek 10

Değer atanmayan ve/veya tanımlanmamış bir değişken kullanılırsa Python hata verir.

```
print (yenisayi)
#Python değişkenleri değer atandığında tanımlandığı için hata mesajı alırsınız.
# yenisayi değişkeni tanımlanmamış olduğu için hata mesajı alınır.
NameError                 Traceback (most recent call last)
<ipython-input-1-9ff08615337f> in <module>()
----> 1 print (yenisayi)

NameError: name 'yenisayi' is not defined
```

Değer atamadan tanımlamak için yenisayi=int( ) kodu kullanılabilir. Bu durumda değişkene ilk değer olarak "0" sıfır atanır.

```
yenisayi=int( )
print(yenisayi)
0
```

Örnek 11

Değişkenler veri tiplerine göre kullanılmazsa Python hata verir.

```
sayi1=1
metin1='deneme'
print(sayi1+metin1) #Bir sayı ile bir metin, kelime toplanamaz.
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-e585c633881d> in <module>()
      1 sayi1=1
      2 metin1='deneme'
----> 3 print(sayi1+metin1) #Bir sayı ile bir metin, kelime toplanamaz.

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Değişken adı verilirken uyulması gereken bazı kurallar ve kurallar kadar katı olmasa da yararlı kullanım önerileri vardır.

Değişken adlandırma kuralları:

- Değişkenler bir harf (a - z, A - Z) veya alt çizgi (_) ile başlamalıdır. Bunların dışında sayı veya başka bir karakter ile de başlayamaz.
- Değişken adında rakam, alt çizgi(_), büyük veya küçük harf olabilir.
- Değişken adları herhangi bir uzunlukta olabilir.

Python, değişken adlandırmada Türkçe karakterlerin (ç , ğ, ı, ö, ş ve ü) kullanımına izin vermektedir. Ek olarak Python programlama dilinde ayrılmış sözcükler kullanılamaz. Bu özel sözcüklerin listesini görmek için aşağıdaki kod kullanılabilir.

```
import keyword
keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass',
'raise', 'return', 'try', 'while', 'with', 'yield']
```

Büyük harf ve küçük harf kullanılarak tanımlanan değişkenlerin adı aynı olsa bile farklı değişkenler olduğu unutulmamalıdır.

Örnek 12

Python dilinde uygun değişken adı örnekleri:

```
#Uygun değişken isimleri
sayi1=1
Sayi1=2
```

Sayi1' ekrana yazdırılırsa çıktı ne olur?

```
print (sayi1)
print(Sayi1)
#Büyük harf ve küçük harf kullanarak tanımlanan değişkenlerin adı aynı olsa bile
farklı değişkenler olduğunu unutulmamalıdır.
sayı1=3
#Python  değişken adlandırmada Türkçe karakter kullanımına izin vermektedir.
print(sayı1)
1
2
3
```

Örnek 13

Python'da hatalı değişken adı kullanımı örneği:

```
1sayi=5 #Hatalı değişken adı.
print (1.sayi)
  File "<ipython-input-52-a0b35430cdb5>", line 1
    1sayi=5 #Hatalı değişken adı.
        ^
SyntaxError: invalid syntax
```

Değişken adlandırma için bazı standartlar vardır. Bu standartlar değişken adının ve içeriğinin anlaşılmasına yardımcı olarak programcıların daha kolay çalışmasını sağlar. Değişken adı, onun içeriği hakkında bilgi verirse kodun anlaşılması kolaylaşır. Değişken adlarının yazımında bazı standart kullanımlar vardır. Birden fazla kelimenin kullanılacağı değişken adlarında kelimelerin ilk harfi büyük olabilir. Camel standardında (başka standartlar da bulunmaktadır) değişken adlarının görünüşü deve hörgücüne benzetilmektedir. Değişkenin adına küçük harfle başlanır ve sonraki her kelime büyük harfle başlar.

Örnek 14

```
adi= "Elif"
soyadi="Altun"
dogumYili=1981
universiteMezunuMu=True
universiteyeBasladigiYil=1999
mezuniyetNotu=2.00
print ('Adı: ', adi)
print ('Soyadı: ', soyadi)
print('Üniversite Mezunu mu? ', universiteMezunuMu)
print('Üniversiteye Başladığı Yıl: ', universiteyeBasladigiYil)
print('Mezuniyet Notu: ', mezuniyetNotu)
Adı:  Elif
Soyadı:  Altun
Üniversite Mezunu mu?  True
Üniversiteye Başladığı Yıl:  1999
Mezuniyet Notu:  2.0
```

Veri tiplerini anlayabilmek için aşağıdaki kod satırı incelenebilir. Aşağıda bir sayı ile bir karakter dizisi üzerinde operatörleri kullanarak işlemler yapılmıştır.

Örnek 15

Hatalı veri tipi kullanımına örnek: "sayi2" değişkeninin tek tırnak içinde verildiğine ve bir karakter dizisi olduğuna dikkat edilmelidir.

```
sayi1=5
sayi2='3'
print (sayi1+sayi2)
TypeError                          Traceback (most recent call last)
<ipython-input-55-294ee141ba94> in <module>()
      1 sayi1=5
      2 sayi2='3'
----> 3 print (sayi1+sayi2)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Kod çalıştırıldığında bir hata mesajı alınır. Bir aritmetik operatörü kullanılırken bir sayı ile bir karakter dizisi (ikinci değişkenin adı sizi yanıltmasın) toplamaya çalışıldığı için Python hata verir. Veri tipleri, değişkenler üzerinde yapılabilecek işlemleri belirler.

Kod çalıştırıldığında bir hata mesajı alınır. Bir aritmetik operatörü kullanılırken bir sayı ile bir karakter dizisi (ikinci değişkenin adı sizi yanıltmasın) toplamaya çalışıldığı için Python hata verir. Veri tipleri, değişkenler üzerinde yapılabilecek işlemleri belirler.

Örnek 16

Operatörler konusunda * operatörü ile ilgili bir ayrıntıdan bahsedilmiştir.

```
sayi1=5
sayi2='3'
#sayi2 değişkenin tek tırnak içinde verildiğinde bir karakter dizisi olduğuna dikkat ediniz.
print (sayi1*sayi2)
#Sizce nasıl bir sonuç çıkar?
33333
```

Örnekte görüldüğü gibi kod "çarpma" işlemi yapamamıştır. Çünkü ortada iki sayısal değer yoktur. İkinci örnekte 3 sayısını bir karakter olarak 5 defa yazmıştır.

Python'da değişkenlere değer atanırken veri tipleri belirtilmez. Python, atanan değere göre veri türünü kendisi belirler. Ancak programlama yaparken veri tiplerini bilmek ve ona göre kullanmak gerekir.

Python programlama dilinde sıkça kullanılan veri tipleri aşağıdaki tabloda sunulmuştur:

Veri Tipi	Sınıfı	Açıklama
Integer	int	Tam sayı. (Örnek: 3, 5, 369963)
Float	float	Ondalıklı sayı. (Örnek: 10.45)
Complex	complex	Karmaşık sayılar A + Bj şeklinde kullanılır. (Örnek: 4+5j)
Karakter dizisi (String)	str	Karakter dizilerini (metinleri) göstermek için kullanılır. Çift tırnak veya tek tırnak içinde gösterilir. "Merhaba Dünya"
Boolean	bool	Sadece True veya False değeri alır. int(True)=1 iken int(False)=0 dır.
Liste	list	Farklı veri türleri içerebilir. listem=['Çınar', 24, 'Mühendis', True]
Demet (tuple)	tuple	Farklı veri türleri içerebilir. demet1 =('Çınar', 24, 'Mühendis', True)
Sözlük (dictionary)	dict	Farklı veri türleri içerebilir. sozluk={'adi': 'Çınar','yasi'=24, 'meslekUnvani':'Mühendis', 'askerlikDurumu': True}

"int" veri tipi Python'da tam sayıların tutulduğu veri tipidir: 3, 5, 198763 gibi değerleri tutar. En çok kullanılan veri tiplerinden biridir.

"float" veri tipi ondalıklı sayıların tutulduğu veri tipidir: 0.5, 234678.67 gibi değerleri tutar.

"complex" veri tipi ise karmaşık sayıların tutulduğu veri tipidir. A+Bj tipinde veriler tutulur: 4+5j gibi değerleri tutar.

Örnek 17

Python'da bir değişkene değer atandığında veri tipleri atanan değere göre otomatik olarak belirlenir.

```
piSayisi=3.14
#float tipinde bir veri
print ('pi sayısı=', piSayisi)
rCm=2
#integer tipinde veri
alan=3.14*2**2
print ('Alan=', alan)
#sonuç float tipinde
print('Yarıçapı 2 olan dairenin alanı ', alan, ' cm 2 dir' )
karmasikSayi=4+5j
print('Bir karmaşık sayı=', karmasikSayi+3j)
pi sayısı= 3.14
Alan= 12.56
Yarıçapı 2 olan dairenin alanı  12.56  cm 2 dir
Bir karmaşık sayı= (4+8j)
```

Karakter dizisi, kullanıcıdan alınan değerlerin metin formatında tutulduğu veri tipleridir. Python karakter dizisi oldukça kullanışlı işlevlere sahiptir.

Örnek 18

Bir karakter dizisi ekrana yazdırılabilir, başka bir karakter dizisiyle birleştirilebilir.

"len( )" metodu bir karakter dizisinin uzunluğunu vermektedir.

```
metin1 = 'Merhaba '
metin2 = 'Mars'
print (metin1)            # karakter dizisinin tamamını yazar
print (metin1 * 2)        # karakter dizisini 2 defa yazar
print (metin1 + metin2)   # iki karakter dizisini birleştirir
print ('metin1  adlı değişkendeki değerin uzunluğu:', len(metin1))
#Boşluğun da bir karakter olduğunu gözden kaçırmayınız.
Merhaba
Merhaba Merhaba
Merhaba Mars
metin1 adlı değişkendeki değerin uzunluğu: 8
```

Bir karakter dizisinin içindeki karakterlere tek tek veya belirli bir aralıkta erişilebilir. Köşeli parantez içinde [ ] tek bir sayı verildiğinde bu karakter dizisinin indisini ifade eder. İndis numarası "0" dan başlayarak karakterin metindeki kaçıncı sırada yer aldığını gösterir. metin[0] ifadesi metin değişkenindeki 1. karakteri verir. Belirli bir aralıktaki karakteri alırken "[başlangıç indisi:bitiş indisi]" şeklinde ifade edilir. metin[0:5] metin değişkeninde indisi 0, 1, 2, 3 ve 4 olan karakterleri dilimler. Bitiş indisi dilimlemeye dahil edilmez. İndislerin "0" dan başladığı unutulmamalıdır.

Başlangıç indisi verilmeyen karakter dizisinde örnek: metin[:7] başlangıç indisi otomatik olarak sıfır (0) olur. Örnek 0, 1, 2, 3, 4, 5 ve 6. karakterlerden oluşan bir metin verir. Bitiş indisi değeri verilmezse başlangıç indisinden başlanarak son karakter dâhil dilimleme işlemi yapılır.

Negatif İndis Sayıları: Karakter dizisine sondan başlandığını ifade eder. metin[-1] karakter dizisinin en sonundaki karakteri verir. metin [-2] ise sondan ikinci karakteri verir.

Karakter dizilerinde indisler ritmik atlanarak dilimleme yapılabilir. Bunun için [başlangıç indisi: bitiş indisi: ritmik artış miktarı] şeklinde kullanılır. "metin[0:8:2]" : 0'dan başlayarak 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.

Örnek 19

Karakter dizilerinin kullanımı ile ilgili örnekler:

```
metin='Merhaba Mars'
print (metin[0])    # ifadenin ilk karakterini yazar.
print (metin[4:7])  # ifadenin 5, 6 ve 7. karakterlerini yazar.
print (metin[8::])  # 9. karakterden sonuncu karaktere kadar yazar.
print (metin[-2])   # karakter dizisinin en sondan ikinci karakterini yazar.
print (metin [:7])  # indisi 0' dan 7'ye kadar olan (7 dahil değil) karakterleri yazar.
print (metin[8:])   # başlangıç indisinden sonra tüm karakterleri yazar.
print(metin[0:8:2]) # 0, 2, 4 ve 6 indis numaralı karakterleri dilimler.
M
aba
Mars
r
Merhaba
Mars
Mraa
```

Liste, demet ve sözlükler kapsamlı konular olduğu için ayrı bölümlerde verilmiştir.

Python, her ne kadar veri tiplerini otomatik olarak verse de bir değişkenin veri tipini kontrol etmek ve kullanım amacına göre değiştirmek gerekebilir. Bir değişkenin veri tipini öğrenmek için "type( )" komutu kullanılır.

Örnek 20

Bir değişkenin veri tipi type( ) komutuyla kontrol edilir. type (degiskenAdi) şeklinde yazılır.

```
sayi1=5
sayi2=10.556
metin1="1"
#metin1 değişkenine tırnak içinde verilen sayının string tipinde bir değişken
olduğuna dikkat ediniz.
sayi3=4+5j
askerlikYaptiMi=True
#True doğru, 1, evet anlamındadır.
print ("1. değişkenin veri tipi: ", type(sayi1))
print ("2. değişkenin veri tipi: ", type(sayi2))
print ("3. değişkenin veri tipi: ", type(metin1))
print ("4. değişkenin veri tipi: ", type(sayi3))
print ("5. değişkenin veri tipi: ", type(askerlikYaptiMi))
listem=['Çınar', 24, 'Mühendis', True]
print ("6. değişkenin veri tipi: ", type(listem))
demet1=('Çınar', 24, 'Mühendis', True)
print ("7. değişkenin veri tipi: ", type(demet1))
sozluk={'adi': 'Çınar','yasi': 24, 'meslekUnvani':'Mühendis', 'askerlikDurumu': True}
print ("8. değişkenin veri tipi: ", type(sozluk))
1. değişkenin veri tipi:  <class 'int'>
2. değişkenin veri tipi:  <class 'float'>
3. değişkenin veri tipi:  <class 'str'>
4. değişkenin veri tipi:  <class 'complex'>
5. değişkenin veri tipi:  <class 'bool'>
6. değişkenin veri tipi:  <class 'list'>
7. değişkenin veri tipi:  <class 'tuple'>
8. değişkenin veri tipi:  <class 'dict'>
```

Tam sayılarla işlem yapıldığında "integer", kesirli sayılarla işlem yapıldığında "float" veri tipi kullanılmaktadır. Değerler üzerinde işlem yaparken (örneğin "input" ile kullanıcıdan veri alırken) içinde sadece rakamlar bulunan "string" ifadeyi sayısal veri tipine dönüştürmek, bazen de bunun tersini yapmak gerekebilir. Bunun için bazı fonksiyonlar bulunmaktadır. Veri tipini dönüştürmek için kullanılan temel fonksiyonlar şunlardır:

int( ) : Veri tipini integer'a çevirir.

float() : Veri tipini float'a çevirir.

str() : Veri tipini karakter dizisine çevirir.

"integer" tipinde bir sayıyla "float" tipinde bir sayı çarpıldığında sonuç "float" tipinde olacağı için Python bu veri tipini otomatik olarak belirler.

TASARIM BECERİ

Milli Eğitim Bakanı Sayın Prof. Dr. Ziya Selçuk'un açıkladığı güçlü yarınlar için 2023 Eğitim Vizyonu kapsamında akademik bilgiyi beceriye dönüştürmeyi hedefleyen, çocuklara ilgi, yetenek ve mizaçları doğrultusunda pratikler kazandırmayı amaçlayan, ortaöğretim kurumlarında Tasarım Beceri Atölyeleri için gerekli adımlar geçen yıl atılmıştı. Bu kapsamda pilot okullar içinde yer alan okulumuz, Göztepe İhsan Kurşunoğlu Anadolu Lisesi'nde Tasarım Beceri Atölyeleri açılışı 1.11.2019 tarihinde gerçekleştirildi.

Açılışa Orta Öğretim Genel Müdürü Sayın Yusuf BÜYÜK ,İstanbul İl Milli Eğitim Müdürü Sayın Levent Yazıcı, Kadıköy Kaymakamı Sayın Dr. Mustafa Özarslan, İlçe Milli Eğitim Müdürü Sayın Sadık Aslan ile çok sayıda okul müdürü, okulumuz öğretmen ve öğrencileri katıldı. Orta öğretim genel müdürümüz Sayın Yusuf Büyük'ün kısa bir konuşma yaptığı açılışta misafirler tarafından atölyeler gezildi ve atölye çalışmaları incelendi. Atölyelerin , okulun öğretmen ve öğrencilerine hayırlı olmasına yönelik dilek ve temennilerle açılış sona erdi.

Bu atölyelerdeki etkinlikler bilim, sanat, spor ve kültür odaklı yapılandırıldı. Tasarım-Beceri Atölyeleri ilkokul, ortaokul ve lise düzeyinde ortak bir amaç doğrultusunda tasarlandı. Çocuğun özellikle elini kullanmasını önemseyen, mesleklerle ilişkilendirilmiş işlikler olması planlandı. Bununla beraber bu atölyeler yeniçağın gerektirdiği problem çözme, eleştirel düşünme, üretkenlik, takım çalışması ve çoklu okuryazarlık becerilerinin kazandırılması için somut mekânlar olarak düzenlendi. Türk eğitim sisteminde ilk olacak Tasarım Beceri Atölyeleri ile öğrencilerin okulda düşünmeye, tasarlamaya ve üretmeye zaman ayırabileceği ortamlar yaratıldı. Böylece öğrenilen, ölçülen ve pratik arasındaki uyumun artması istendi. Tasarım Beceri Atölyeleri sayesinde öğrenci gelişimine olabildiğince fazla katkı sağlamak ve işbirliği, yaratıcılık, eleştirel düşünme, iletişim, sosyal ve kültürel farkındalık, sebat, uyumluluk, liderlik, girişimcilik, merak, empati gibi kişisel özellikler ve yeterliliklerin art tırılması amaçlandı. Disiplinler ötesi bir disiplin oluşturuldu.

Öğrenciler soru çözme, konu anlatımı gibi bir eğitim anlayışından üretimi, yapmayı, etkileşimi, derinleşmeyi öne çıkaran bir müfredat anlayışına yönelecek. Tasarım-Beceri Atölyeleri böyle bir müfredat yaklaşımının aracı işlevini görecek. Bilmekten çok tasarlamanın, yapmanın, üretmenin ön plana çıkacağı bu atölyeler çocuğun kendisini, meslekleri, çevresini tanımasına yardımcı olacak. Bu çalışmaların "E-portfolyo", "okul profili değerlendirmesi" gibi başka projelerle de etkileşim halinde olacak. Atölyeler aslında çok uzun vadeli bir gelecek yatırımı, eğitim sistemimizdeki anlayış değişikliğinin ve dönüşümün temeli

ATÖLYELERİ'MİZ AÇILDI

Okul Müdürümüz Sayın Ramazan Çoban'ın katkı ve destekleriyle okulumuzda bilişim teknolojilerine yönelik uygulamalar doğrultusunda öğrencilerimizin yeni bilgi ve teknolojileri üreten, bilişim araçlarını doğru, etkin ve güvenli kullanmalarını sağlayan becerilerin geliştirilmesi amacıyla Robotik Kodlama, Üç Boyutlu Tasarım Atölyesi; çizgi, boya ve değişik maddeler kullanılarak çeşitli teknikler ile temel tasarım ilkelerine uygun, estetik değeri olan çalışmalara yönelik olarak Ebru Atölyesi, Linol Atölyesi, Kaligrafi Atölyesi; ahşap ürünlerin tasarlanması ve üretilmesine yönelik Ahşap Atölyesi; sese biçim veren ,malzemesi ses ve söz olan sanatsal alanlara ilişkin Müzik Atölyesi açılmıştır. Tasarım-Beceri Atölyeleri Öğretmen ve Yönetici Eğitimi'ne katılan öğretmenlerimiz, her öğrencimizin ilgi ve yeteneği doğrultusunda gönüllü olarak yer aldığı atölye çalışmalarına başladı, atölye çalışmalarından oluşacak bir serginin 2020 Mayıs ayında açılması planlanıyor.

Milli Eğitim Bakanı Sayın Prof. Dr. Ziya Selçuk "Açılmamış kanatların genişliği bilinmez , açalım kanatlarını çocuklarımızın vakit gelmiştir." demişti. Bizler de Tasarım Beceri Atölyeleri ile öğrencilerimizin yeteneklerini fark etmelerine rehberlik ediyoruz.

GIKAL MUN CLUP

Let's start with what MUN is. So, what exactly is MUN? Model United Nations, also known as Model UN or MUN, is an extra- curricular activity in which students typically roleplay delegates to the United Nations and simulate UN committees. This activity takes place at MUN conferences, which are usually organized by a high school or a college MUN club.

What do we do as the GIKAL MUN CLUB? Most recently some of our club members joined the HAYDARPASAMUN conference. One of our members, Ece Çakır received the "Outstanding Delegate" award.

The main things we do as the MUN club are; meet up to discuss possible MUN conferences we can attend, do MUN conference simulations to practice for upcoming conferences and do mock debates to practice our English speaking and debating skills.

Who is in the MUN club? We have about 40-50 student members ranging from 9 th to 11 th grade. The presidents of the club are Ece Çakır and Melis Ada Onur. The club was created by Tuana Yetimoğlu last year and the president position was handed down from her to our current presidents.

Under the supervision of Aslı Özen, we have done two MUN simulations this year. The first one was done on November 18 th . It was about the "Yemen Crisis" and the delegates Haktan Oral and İrem Ayber shared the first-place honor for "Best Delegate".

Our second simulation was held on January 15 th and the topic was "Gender Equality in the Workplace".

This year we are planning on attending MUNSA, TAHRANMUN and MALMUN. We are also planning on doing another MUN simulation after our first exams of the semester end. And of course, we will continue to hold meetings and mock debates in between.

MUN has helped us expand our English vocabulary, learn vital communication and debating skills, which we can use in the future. It has helped us to have more confidence while giving public speeches. It has also helped us to learn about relevant world issues and politics.

We encourage any and all students to come join our club.

Ece Çakır 10-C/463
MUN CLUP Co-President


Let's make ideas happen


Funded by the
Erasmus+ Programme
of the European Union
Erasmus+
FONDACIJA
TEMPUS
START UP!
GROWTH
YEAR
IDEA
GROWTH!
INCREASE
YOUR MONEY!
STATISTICS
TEAM
BUSINESS
THINKING!
HOW TO START
YOUR OWN COMPANY
Let's Make
Ideas Happen
2020-1-RS01-KA229-065357_5