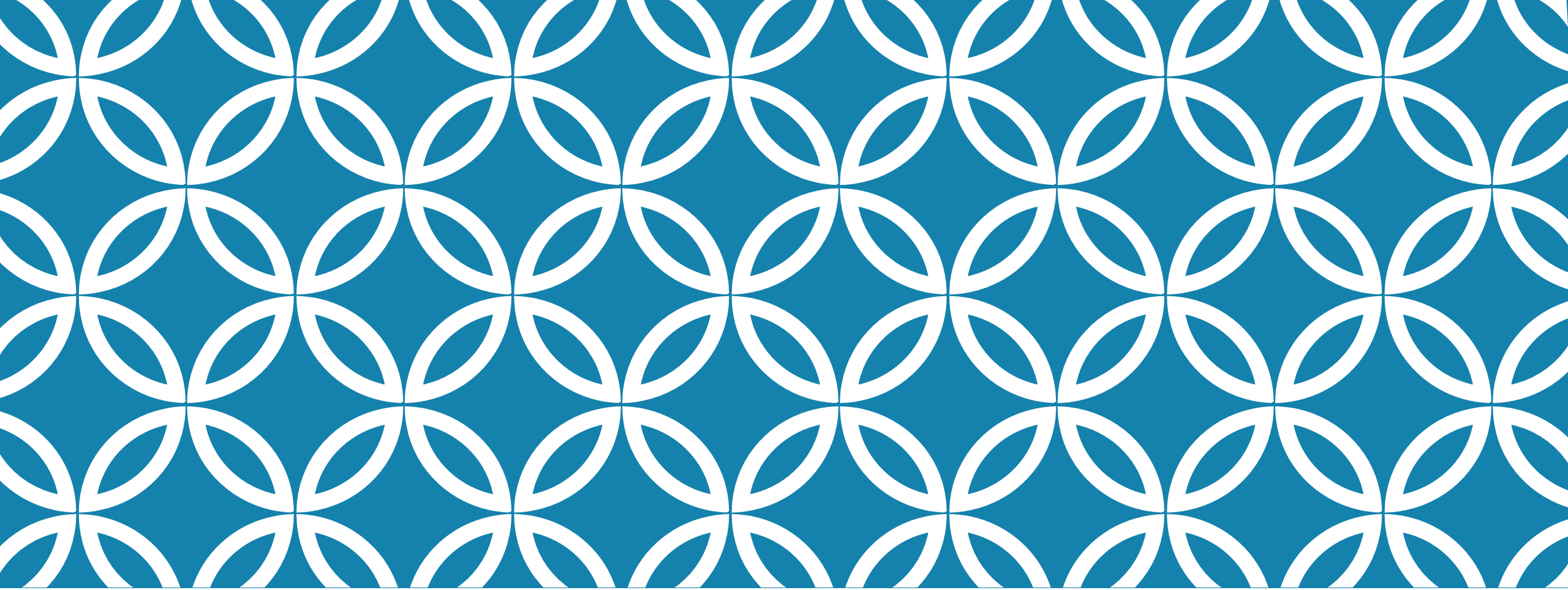

PROGRAMLAMA TEMELLERİ

Öğr. Gör. Erhan AKAGÜNDÜZ

TUPLE (DEMET) VERİ TİPİ

- Listeler konusunda oluşturulan donanim listesi üzerinden daha sonra değişiklikler yapılabildiğini gördünüz.
- Tuple veri tipi de listelere oldukça benzemektedir.
- Aralarındaki temel fark ise tuple veri tipinin tanımlandıktan sonra değişikliğe yani eleman ekleme ya da silmeye izin vermemesidir.
- Tuple veri tipi ile yapılabilecek işlemler şu şekildedir:

TUPLE (DEMET) VERİ TİPİ

- **Tuple oluşturma:** Tuple tanımlaması yapılırken listelerden farklı olarak parantezler kullanılır.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler)
```

```
('bit', 'inç', 'byte', 'hertz', 'piksel')
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple elemanlarına ulaşma:** Listelerdeki gibi indeks kullanılır.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler[3])
```

```
hertz
```

- Listelerde olduğu gibi negatif indekslerde kullanılabilir.

- -1 en sondaki eleman anlamına gelirken -2 sondan iki önceki elemanı temsil eder.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler[-3])
```

```
byte
```

TUPLE (DEMET) VERİ TİPİ

- **Indeks aralıklarına göre yazdırma:** Listelerde olduğu gibi başlangıç ve bitiş indeksleri verilerek istenilen aralık yazdırılabilir.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler[1:3])
```

```
('inç', 'byte')
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple elemanlarını değiştirme:** Tuple veri tipi tanımlanırken elemanların **değiştirilemeyeceğinden** bahsettik.
- Eğer tuple veri tipi listeye çevrilirse elemanlar değiştirilebilir.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
birimler_liste=list(birimler) #burada tuple listeye çevrildi.
birimler_liste[2]="mega byte" #listenin indeksi 2 olan elemanı değiştirildi.
print(birimler_liste)
```

```
['bit', 'inç', 'mega byte', 'hertz', 'piksel']
```

TUPLE (DEMET) VERİ TİPİ

- **Elemanın olup olmadığını sorgulama:** Tuple veri tipinde de listelerde olduğu gibi **in** operatörü ile bir elemanın listede olup olmadığı kontrol edilebilir.
- Eleman tuple'daysa **True**; yoksa **False** değerleri üretilir.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print("bit" in birimler)
```

```
True
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple uzunluğunu bulma:** len fonksiyonu ile tuple'ın eleman sayısı bulunur.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(len(birimler))
```

```
5
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple içinde bir elemanın sayısını bulma:**
  - Bu işlem için listelerde olduğu gibi count fonksiyonu kullanılır.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
say = birimler.count("piksel")
print(say)
```

```
1
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple içindeki elemanın indeksini bulma:**
  - Listelerde olduğu gibi index fonksiyonu kullanılır.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
print(birimler.index("byte"))
```

```
2
```

TUPLE (DEMET) VERİ TİPİ

- **Tuple birleştirme:**
  - Birden fazla tuple birleştirilerek tek bir tuple'da toplanabilir.

```
birimler = ("bit", "inç", "byte", "hertz", "piksel")
degerler = (8, 256, 1024)
birlestir = birimler + degerler
print(birlestir)
```

```
('bit', 'inç', 'byte', 'hertz', 'piksel', 8, 256, 1024)
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Python programlama dilinde sırasız, değiştirilebilir ve belirli bir konuma sahip koleksiyonlar sözlük olarak adlandırılır.
- Sözlükler süslü (ya da kırlangıç{ }) parantezler arasına yazılır.
- Sözlük veri tipinde anahtarlar ve bu anahtarların değerleri vardır.
- Her anahtardan sonra iki nokta (:) kullanılır ve değer yazılır.
- Anahtar:değer (key:value) ikilileri virgülle birbirinden ayrılır.

DICTIONARY (SÖZLÜK) VERİ TİPİ

- **Hatırlatma:**
  - **Liste** veri tipinde **köşeli parantez [ ],**
  - **tuple** veri tipinde **normal parantez ( ),**
  - **sözlük** veri tipinde ise **süslü parantez { }** kullanılır.
- Farklı şekillerde tanımlanabilen sözlük veri tipinin genel kullanımı şu şekildedir:
- sozluk_adi = { anahtar : deger }

DICTIONARY (SÖZLÜK) VERİ TİPİ

- **Örnek:**

```
sozluk = {"Mesleğiniz":"Öğrenci", "Alanınız":"Bilişim", "Yaşadığınız Yer":"Van"}
print(sozluk)
```

```
{'Mesleğiniz': 'Öğrenci', 'Alanınız': 'Bilişim', 'Yaşadığınız Yer': 'Van'}
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlükte sadece anahtarları göstermek için **key** ve **values** fonksiyonları kullanılır.
- **Örnek:**

```
sozluk = {"Mesleğiniz":"Öğrenci", "Alanınız":"Bilişim", "Yaşadığınız Yer":"Van"}
print(sozluk.keys())
print(sozluk.values())
```

```
dict_keys(['Mesleğiniz', 'Alanınız', 'Yaşadığınız Yer'])
dict_values(['Öğrenci', 'Bilişim', 'Van'])
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlük elemanlarına erişim aşağıdaki şekilde yapılmaktadır.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print(donanim["Türü"])
```

```
RAM
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlük içindeki değerleri değiştirebilirsiniz.
- Aşağıda değer değişimine yönelik bir örnek bulunmaktadır.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim["Kapasitesi"]="16 GB" #burada 8 GB değeri, 16 GB değeri ile değişti.
print(donanim)
```

```
{'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '16 GB'}
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Diğer veri tiplerinde olduğu gibi sözlüklerde de bir değerin olup olmadığına **in operatörü** ile bakılabilir.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print("Türü" in donanim) #Sözlükte Türü anahtarının olup olmadığı kontrol edilmiştir.
```

```
True
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlüklerde de uzunluk **len** fonksiyonu ile bulunur.
- Burada eleman sayısının **anahtar-değer** ikilileri olarak hesaplanacağını unutmayınız.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
print(len(donanim))
```

```
3
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlüğe daha sonra anahtar-değer ikilileri eklenebilir.
- Aşağıdaki örnekte ikinci satıra dikkat ediniz.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim["Hızı"]="2400 MHz" #burada ekleme işlemi yapılmıştır.
print(donanim)
```

```
{'Türü': 'RAM', 'Tipi': 'DDR4', 'Kapasitesi': '8 GB', 'Hızı': '2400 MHz'}
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- Sözlük veri tipinde silme işlemi yapmak için **pop** fonksiyonu kullanılır.
- **Örnek:**

```
donanim = {"Türü":"RAM", "Tipi":"DDR4", "Kapasitesi":"8 GB"}
donanim.pop("Kapasitesi") #burada silme işlemi yapılmıştır.
print(donanim)
```

```
{'Türü': 'RAM', 'Tipi': 'DDR4'}
```

DICTIONARY (SÖZLÜK) VERİ TİPİ

- **del** fonksiyonu ile sözlük tamamen silinebilir.
- Sözlüğü silmek yerine içini boşaltmak için **clear()** fonksiyonu kullanılır.
- Sözlüğü kopyalamak için listelerde olduğu gibi **copy()** fonksiyonu kullanılır.

SET (KÜME) VERİ TİPİ

- Python programlama dilinde kullanılan veri tiplerinden biri de **set (küme) veri** tipidir.
- Sözlükler gibi süslü parantezlerin içine yazılan set veri tipi, sözlüklerden farklı olarak ikili anahtar yapısında değildir.
- Set veri tipinde elemanlar sırasızdır ve tekrar etmez.
- Türkçeye küme olarak çevrilen bu veri tipi bir dizi matematiksel işlemin kolaylaştırılmasını sağlar.

SET (KÜME) VERİ TİPİ

- Set veri tipinin basit kullanımı şu şekildedir:

```
sayilar = {1, 2, 3, 4, 5} #integer veri tipi tırnak içinde yazılmaz.
print(sayilar)
```

```
{1, 2, 3, 4, 5}
```

SET (KÜME) VERİ TİPİ

- Set veri tipinde de fonksiyonlar kullanılarak bir dizi işlem yapılabilir.
- Bu fonksiyonlar genel olarak **liste**, **sözlük** ve **demet** veri tipindeki fonksiyonlarla benzerdir.
- Aşağıda bu fonksiyonlara bazı örnekler verilmiştir:
  1. Bir elemanın küme içinde olup olmadığı **in fonksiyonu** ile kontrol edilir.

```
sayilar = {1, 2, 3, 4, 5}
print(6 in sayilar)
```

```
False
```

SET (KÜME) VERİ TİPİ

2. Küme veri tipinde eleman eklemek için **add()** fonksiyonu kullanılır.

```
sayilar = {1, 2, 3, 4, 5}
sayilar.add(6)
print(sayilar)
```

```
{1, 2, 3, 4, 5, 6}
```

3. Tek bir eleman yerine birden fazla eleman eklemek için **update()** fonksiyonu kullanılır.

```
sayilar = {1, 2, 3, 4, 5}
sayilar.update([6, 7, 8])
print(sayilar)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

SET (KÜME) VERİ TİPİ

4. Set içindeki bir elemanı silmek için **remove()** ya da **discard()** fonksiyonları kullanılır. Her iki fonksiyonunun kullanımı aynıdır.

```
sayilar = {1, 2, 3, 4, 5}
sayilar.discard(3)
print(sayilar)
```

```
{1, 2, 4, 5}
```

SET (KÜME) VERİ TİPİ

- **Önemli Not:**
- Verilen örneklerde sadece integer tipi kullanılmış olsa da set veri tipinde farklı veri tiplerini (aynı kümede integer, string veya float gibi) aynı anda kullanabilirsiniz.