2024-2025 EĞİTİM VE ÖĞRETİM YILI

PROGRAMLAMA DİLLERİ MODÜLÜ (PYTHON)

2. ÜNİTE

PYTHON PROGRAMLAMA DİLİ İLE YAZILIM GELİŞTİRME

Kaynak: onuraltuntas.net

1. Karar Yapıları Nedir?

- Karar yapıları, programların belirli koşullara göre farklı sonuçlar üretmesini sağlar.

- Gerçek hayat örneği: Eğer hava yağmurluysa şemsiye alırım, yoksa almam.

Kaynak: onuraltuntas.net

2. If-Else Yapısı

- If-else koşulu, bir durum doğruysa bir kod bloğu çalıştırır, aksi halde başka bir kod bloğunu çalıştırır.

- Örnek Kod:

```
hava_durumu = "yağmurlu"
if hava_durumu == "yağmurlu":
    print("Şemsiyeni al!")
else:
    print("Şemsiyeye gerek yok.")
```

Kaynak: onuraltuntas.net

3. If-Elif-Else Yapısı

- If-elif-else, birden fazla durumu kontrol etmek için kullanılır.

- Örnek Kod:

```
saat = 15
if saat < 12:
    print("Günaydın!")
elif saat < 18:
    print("İyi günler!")
else:
    print("İyi akşamlar!")
```

4. İç İçe If Yapıları

- Karar yapılarının içinde başka bir karar yapısı olabilir.
- Örnek Kod:

```
hava = "güneşli"
sıcaklık = 30
if hava == "güneşli":
    if sıcaklık > 25:
        print("Hadi denize gidelim!")
    else:
        print("Parkta yürüyüş yapabiliriz.")
```

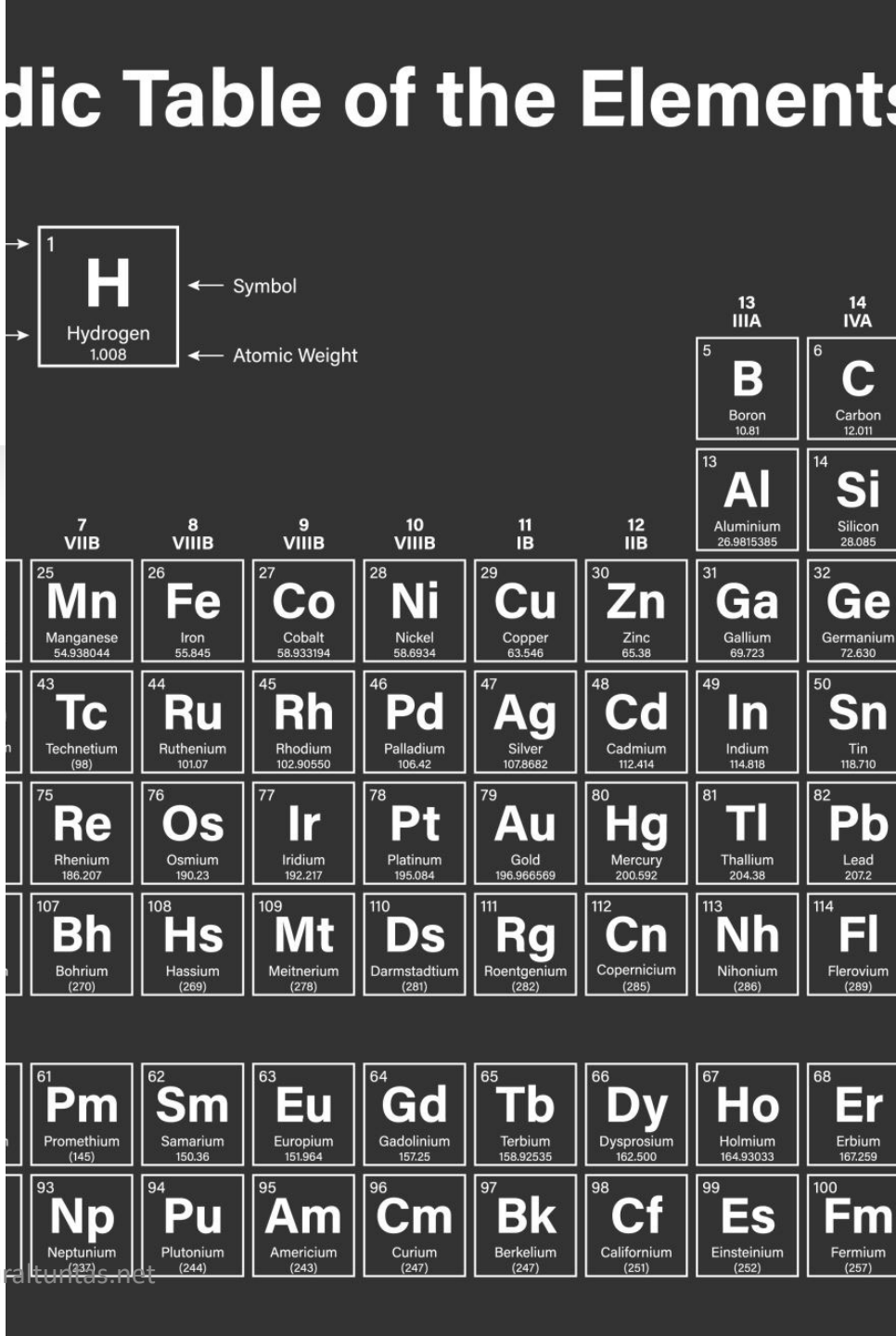
Kaynak: onuraltuntas.net

5. Boolean (Mantıksal) Yapılar

- Boolean yapılar, koşulların sonucunu doğru veya yanlış olarak döner.

- Örnek Kod:

```
yağmur_yağıyor = True
if yağmur_yağıyor:
    print("Şemsiyeni al.")
```



Kaynak: onuralkuntas.net

6. If-Else ve If-Elif-Else Karşılaştırması

If-else: İki durum kontrol edilir.

If-elif-else: Birden fazla durumu kontrol etme imkânı sağlar.

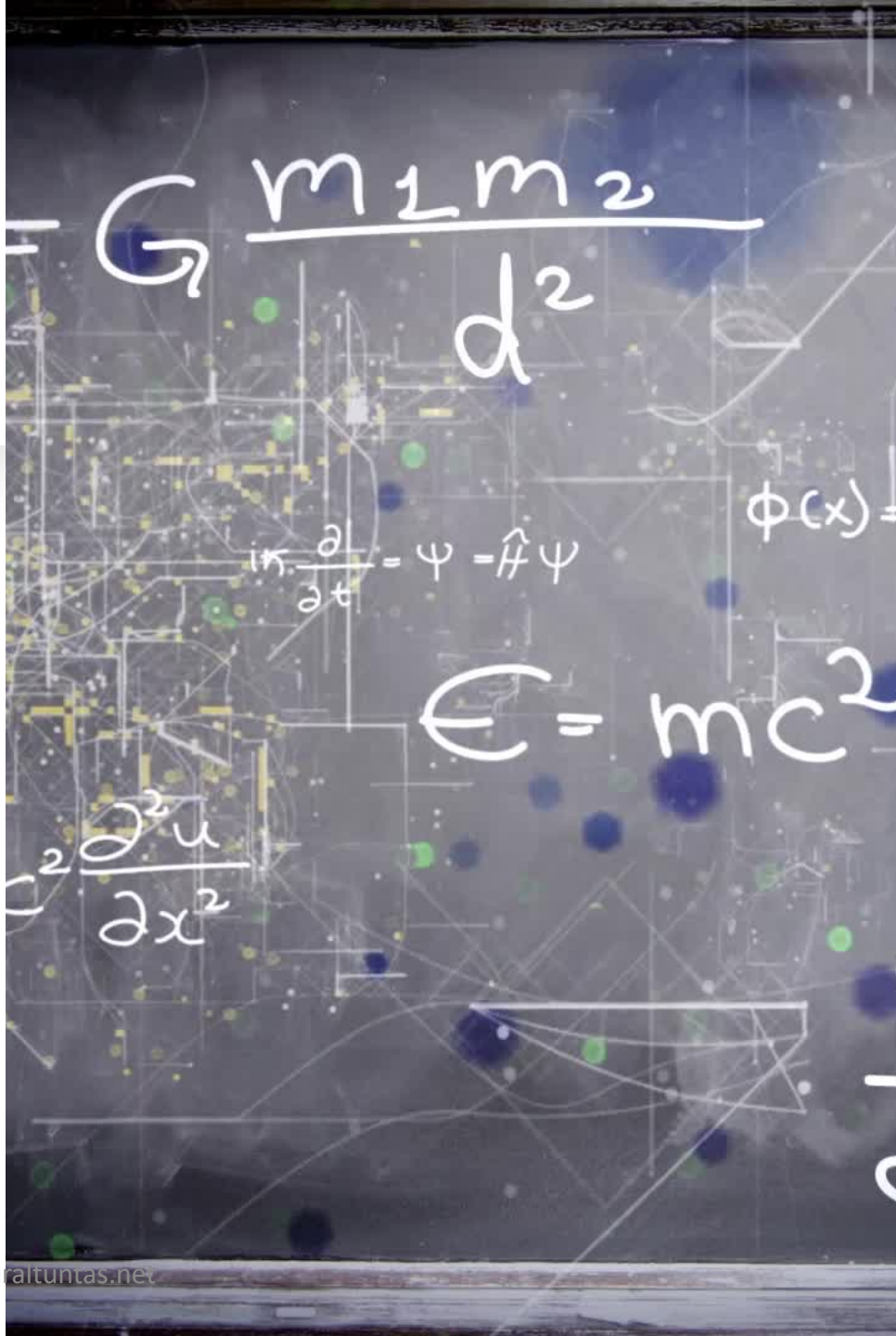
Kaynak: onuraltuntas.net

7. Farklı Problemler

- Problem: Bir öğrencinin notuna göre başarı durumunu belirleyelim.

- Örnek Kod:

```
notu = 85
if notu >= 90:
    print("Pekiyi")
elif notu >= 70:
    print("İyi")
else:
    print("Daha fazla çalışmalısın.")
```



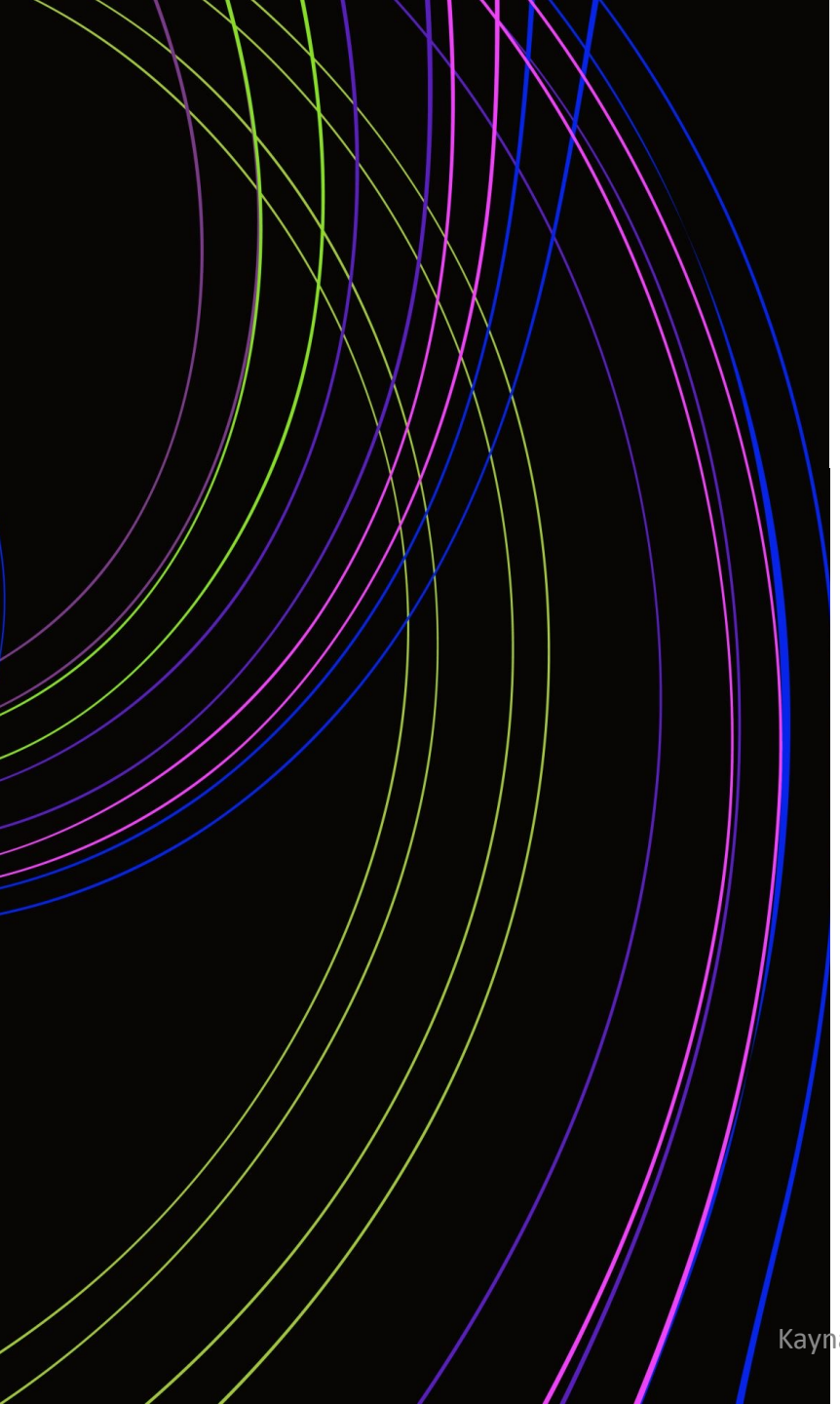
Kaynak: onuraltuntas.net

1. Döngü Yapıları Nedir?

- Döngü yapıları, belirli koşullar sağlandığı sürece aynı işlemi tekrar tekrar gerçekleştiren yapılar.

- Gerçek hayat örneği: Bir arkadaşına 5 kez 'Merhaba' demek istediğinde aynı kodu tekrar yazmak yerine bir döngü kullanabilirsin.

Kaynak: onuraltuntas.net



2. For Döngüsü

- For döngüsü, bir veri kümesi veya belirli bir aralık üzerinden tekrar eder.

- Örnek Kod:

```
for i in range(5):
    print("Merhaba")
```

3. While Döngüsü

- While döngüsü, koşul doğru olduğu sürece çalışmaya devam eder.

- Örnek Kod:

```
sayı = 0
while sayı < 5:
    print("Merhaba")
    sayı += 1
```

4. İç İçe Döngüler

- Bir döngünün içinde başka bir döngü çalıştırılabilir. Genellikle çok boyutlu veri yapılarında kullanılır.

- Örnek Kod:
- for i in range(3):
- for j in range(2):
- print(f"Dış döngü: {i}, İç döngü: {j}")

5. For ve While Döngüsü Arasındaki Fark

- For döngüsü genellikle bilinen bir aralıkta kullanılır.
- While döngüsü ise koşul sağlandığı sürece çalışır.

- Örnek Kod:

For: for i in range(3): print(i)

While: i = 0

```
while i < 3:
    print(i)
    i += 1
```

6. Range Kavramı

- Range, belirli bir aralıkta döngünün çalışmasını sağlar.

- Örnek Kod:

```
for i in range(1, 10, 2):
    print(i)  # 1'den 9'a kadar 2 artırarak ilerler
```

7. Break ve Continue

- Break, döngüyü tamamen sonlandırır.
- Continue, döngünün o adımını atlayarak bir sonrakine geçer.

- Örnek Kod:

```
for i in range(5):
    if i == 3:
        break
    print(i)

for i in range(5):
    if i == 3:
        continue
    print(i)
```

1. Çoklu Koşullar ve Mantıksal Operatörler

- Çoklu koşullar, birden fazla durumun değerlendirilmesi gereken yerlerde kullanılır.
- Mantıksal operatörler: 'and', 'or', 'not'.
- Pozitif Mantık: Tüm koşullar doğru olmalı (and).
- Negatif Mantık: Tek bir koşul doğruysa yeterli (or).

2. Pozitif ve Negatif Mantık

- Pozitif mantıkta tüm koşulların doğru olması gerekir (and).
- Negatif mantıkta tek bir koşulun doğru olması yeterlidir (or).

- Örnek Kod:

```
x = 5
y = 10
if x > 0 and y > 0:
    print("Her iki sayı da pozitif")
if x < 0 or y > 0:
    print("Bir koşul doğru")
```

Kaynak: onuraltuntas.net

3. İç İçe Döngüler

- Bir döngü içinde başka bir döngü olabilir. Özellikle çok boyutlu yapılar ve tekrar eden hesaplamalar için kullanılır.

- Örnek: Çarpım tablosu
- Örnek Kod:

```
for i in range(1, 11):
    for j in range(1, 11):
        print(f"{i} x {j} = {i*j}")
```

4. Kod Tekrarını Azaltma

- Döngüler sayesinde aynı işlemi tekrar tekrar yazmak zorunda kalmazsın.
- Öğrencilere içinde tekrar eden komutların bulunduğu kod parçaları verilerek, bunların döngülerle nasıl sadeleştirileceği tartışılır.

Kaynak: onuraltuntas.net

5. For ve While Döngüleri

- For döngüsü: Belirli bir aralık ya da veri kümesi üzerinde döner.
- While döngüsü: Koşul sağlandığı sürece döner.
- Örnek Kod:

For: for i in range(5): print(i)

While: i = 0

```
while i < 5:
    print(i)
    i += 1
```

Kaynak: onuraltuntas.net

6. Döngülerin Programı Sadeleştirmesi

Döngü kullanarak aynı işlemi tekrar tekrar yazmaktan kaçınabiliriz.

Bu, kodu daha okunabilir ve bakımı daha kolay hale getirir.

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.

Kaynak: onuraltuntas.net

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):
    print(f"Merhaba, {isim}!")

selam_ver("Ali")
```

Kaynak: onuraltuntas.net

4. Fonksiyon Kullanımının Avantajları

- Fonksiyonlar, kod tekrarını azaltır ve programı daha düzenli hale getirir.
- Aynı işlemi birçok yerde yapmak gerektiğinde fonksiyonları kullanarak kodları tekrar yazmak yerine fonksiyon çağrısı yapabiliriz.

Kaynak: onuraltuntas.net

5. Öğrencilerden İsim Yazdıran Fonksiyon

- Bir fonksiyon yazın ve bu fonksiyon, kullanıcının girdiği ismi ekrana yazdırsın.

- Örnek Kod:

```
def isim_yazdir(isim):
    print(f"İsim: {isim}")

isim = input("Adınızı girin: ")
isim_yazdir(isim)
```

6. Parametreli ve Parametresiz Fonksiyonlar Arasındaki Fark

- 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)
```

1. Ön Tanımlı Fonksiyonlar

- Python'da birçok fonksiyon önceden tanımlanmıştır ve kullanıma hazırdır.
- Örnek: len(), count(), insert() gibi fonksiyonlar.
- Örnek Kod:

```
liste = [1, 2, 3, 4]
print(len(liste))  # 4
print(liste.count(2))  # 1
liste.insert(2, 5)
print(liste)  # [1, 2, 5, 3, 4]
```

2. Math Kütüphanesi

- Math kütüphanesi, matematiksel işlemler için kullanılır.

- Örnek: math.factorial(8)
- Örnek Kod:

```
import math
print(math.factorial(8))  # 40320
```

Kaynak: onuraltuntas.net

3. Hazır Kütüphaneler

Python'da yerleşik birçok kütüphane vardır, bu kütüphaneler belirli işlevleri hızlıca gerçekleştirmeye yarar.

Örnek: math, random, datetime, os

Kaynak: onuraltuntas.net

4. Gömülü Fonksiyonlar ve Kütüphaneler

Gömülü fonksiyonlar: Python'da önceden tanımlanmış ve hemen kullanılabilecek fonksiyonlardır.

Kütüphaneler: Belirli bir işlevi gerçekleştiren fonksiyonları barındıran koleksiyonlardır.

Örnek Kütüphaneler: NumPy, Pandas, Matplotlib

Kaynak: onuraltuntas.net

5. Üst Düzey Fonksiyonlar

Üst düzey fonksiyonlar, fonksiyonları argüman olarak alabilen ve/veya fonksiyonları döndüren fonksiyonlardır.

Örnek: map(), filter(), reduce()

Kaynak: onuraltuntas.net

6. Kendini Çağıran Fonksiyonlar (Recursion)

- Bir fonksiyon, kendi içinde kendini çağırabilir. Bu teknik 'recursion' olarak bilinir.

- Örnek Kod:

```
def faktoriyel(n):
    if n == 1:
        return 1
    else:
        return n * faktoriyel(n-1)
```

- print(faktoriyel(5)) # 120

7. Kütüphanelerin Önemi

- Python kütüphaneleri, kodu daha verimli hale getirir ve belirli işlemleri kolaylaştırır.

- Örnek:
- NumPy ile bilimsel hesaplamalar, Pandas ile veri analizi.

Kaynak: onuraltuntas.net