CENG 415 Evrimsel Hesaplama

Bölüm 7: Python Dili

Şevket Umut Çakır

Pamukkale Üniversitesi

13 Aralık 2021

Anahat I

Anahat II

- Lambda İfadeler ve Yüksek Dereceli Fonksiyonlar
- pass İfadesi

9 Hata Yakalama

10 Nesneye Yönelik Programlama
- Sınıf Oluşturma ve Nesne Tanımlama
- Yapıcı Metot(Constructor)
- Kalıtım
- Büyülü Metotlar

Giriş

- Öğrenmesi kolay, güçlü bir programlama dilidir
- Hızlı program/prototip geliştirmeye olanak sağlar
- Bloklar girintilerle oluşturulur
- Değişkenler için tür bildirimine gerek yoktur
- Bir çok paradigmayı destekler(nesne yönelimli, fonksiyonel)
- Okunabilirlik ve üretkenliğe odaklanılmıştır
- Etkileşimli kabuğa sahiptir

Değişkenler

- Python dinamik tür yapısına sahiptir
- Değişkenin türü atama esnasında belirlenir
- Tür denetimi çalışma anında gerçekleşir

Değişkenler

Örnekler

```
metin = "Bu bir string"
metin = """Çok satırlı
string"""
sayi = 2021 # int
sayi = int("2021")
pi = 3.14159265 # float
bos = None # Boş
```

İşleçler(Operatörler)

Tür	**İşleçler**	**Örnek**
Aritmetik	+, -, *, /, %, **, //	`x + y`
Karşılaştırma	==, !=, <>, >, <, >=, <=	`a > b`
Atama	=, +=, -=, *=, /=, %=, **=, //=	`x += 2`
Bit tabanlı	&, |, ^, ~, <<, >>	`a & b`
Mantıksal	and, or, not	`True and False`
Üyelik	in, not in	`x in liste`
Kimlik	is, is not	`a is None`

Karar ve Kontrol Yapıları

- Bloklar : sembolü ile başlar
- Blok içindeki kısım girinti ile tanımlanır
- İç içe bloklar için daha fazla girinti vardır

if Yapısı

Örnek

```
sayi = int(input('Bir sayı girin: '))
if sayi < 0:
    print('Negatif bir sayı girdiniz.')
elif sayi > 0:
    print('Pozitif bir sayı girdiniz.')
else:
    print('Sıfır girdiniz.')
    print('Tebrikler!') # else bloğundaki başka bir satır
```

for döngüsü

```
for i in range(10): # 0-9 arası değerler
    print(i)
```

```
metin = input('Bir metin girin: ')
for harf in metin:
    print(harf)
```

Şekil: Metnin karakterlerini yazdırma

for döngüsü

```
sayi = int(input('Bir sayı girin: '))
for i in range(2, sayi):
    asal = True
    for k in range(2, i):
        if i % k == 0:
            asal = False
            break
    if asal:
        print(i)
```

Şekil: Girilen sayıdan küçük asal sayılar

while döngüsü

```
sayi = int(input('Toplanacak sayıları girin(-1 çıkış): '))
toplam=0
while sayi != -1:
    toplam += sayi
    sayi = int(input('Toplanacak sayıları girin(-1 çıkış): '))
print('Toplam: '+str(toplam))
```

Şekil: -1 girilene kadar sayı alan döngü

String Veri Türü

```
print('merhaba') # merhaba
print("dünya") # dünya
print('Bana "merhaba" dedi!') # Bana "merhaba" dedi!
print("Bana \"merhaba\" dedi!") # Bana "merhaba" dedi!
print(r"c:\users\pau") # c:\users\pau
print('Çok\nsatırlı\nmetin') # 3 satır içerir
print("""Çok
satırlı
metin""")
print(f"3+5={3+5}") # Python 3.5 ve sonrasında
```

Şekil: String tanımlama

String İşleçleri

```
print(3*'ab'+'cd') # abababcd yazdırır. * tekrarlar, + birleştirir
print('merhaba ' 'dünya') # yan yana stringler birleştirilir
s='Merhaba dünya!' # s metnini tanımlar
print(s[2]) # r yazar
print(s[3:7]) # haba yazar
print(s[-1]) # ! yazar
print(s[5:]) # ba dünya! yazar
print(s[:]) # Merhaba dünya! yazar
m='Python' # indislere göre konumlar aşağıdadır.
# +---+---+---+---+---+---+
# | P | y | t | h | o | n |
# +---+---+---+---+---+---+
#   0   1   2   3   4   5
#  -6  -5  -4  -3  -2  -1
```

Şekil: String işleçleri

String Metotları

- **lower(), upper():** Metnin sırasıyla küçük ve büyük harfli hallerini verir.
- **strip():** Metni başındaki ve sonundaki boşluklar olmadan verir.
- **startswith(str), endswith(str):** Metnin başka bir metinle başladığını ve bittiğini True ve False olarak verir.
- **find(str):** Metnin içinde başka bir metni arar; bulursa konumunu, bulamazsa -1 döndürür.
- **count(str):** Metnin içinde başka bir metnin kaç defa geçtiğini verir.
- **replace(eski, yeni):** Metnin içinde eşleşen değerleri başka bir metinle değiştirir.
- **split(ayraç):** Metni ayraca göre parçalara ayırır ve sonucu liste olarak döndürür.
- **join(liste):** Bir listenin elemanlarını aralarına metni koyarak birleştirir.

String Metotları

```
s='Türkçe karakter uyuşmazlığı'
print(s.upper()) # TÜRKÇE KARAKTER UYUŞMAZLIĞI
print(s.upper().lower()) # türkçe karakter uyuşmazliği
print(s.startswith('Türkçe')) # True
print(s.replace('karakter', 'harf')) # Türkçe harf uyuşmazlığı
l=s.split(' ') # ['Türkçe', 'karakter', 'uyuşmazlığı']
print('- . -'.join(l)) # Türkçe- . -karakter- . -uyuşmazlığı
print(s.count('a')) # 3
print(len(s)) # 27 (uzunluğu verir)
```

Şekil: String metotları

Listeler

- Çok kullanılan bir veri türüdür
- Kullanımı dizilere benzer
- `[]` veya `list()` ile başlatılabilir
- Eleman eklemek ve silmek mümkündür
- Farklı türlerden değerler aynı listede bulunabilir

Listeler

```
l = ['kırmızı', 'yeşil', 'mavi', 'sarı']
print(l) # ['kırmızı', 'yeşil', 'mavi', 'sarı']
print(l[0]) # kırmızı
print(l[1:3]) # ['yeşil', 'mavi']
print(len(l)) # 4
l2 = l # yeni liste oluşturulmaz, her ikisi de aynı listeyi göster
bos_liste = [] # boş liste oluşturur
bos_liste = list() # boş liste oluşturur
l3 = ['siyah', 'mor']
print(l[2:] + l3) # ['mavi', 'sarı', 'siyah', 'mor']
```

Şekil: Liste kullanımı

Listeler

for Döngüsü Kullanımı

```
renkler = ['kırmızı', 'yeşil', 'mavi']
for renk in renkler:
    print(renk)
print('yeşil' in renkler) # True
print('sarı' in renkler) # False
```

Şekil: Liste for ve in kullanımı

Liste Metotları

- **append(eleman):** Listenin sonuna eleman ekler.
- **insert(indis, eleman):** Listenin belirtilen indisine araya eleman ekler.
- **extend(list):** Listenin sonuna başka bir listeyi ekler. Orjinal liste değişir.
- **index(eleman):** Elemanı listede arar, varsa konumunu döndürür. Yoksa ValueError hatası verir.
- **remove(eleman):** Verilen elemanı listeden siler. Döngü içinde yapılmamalıdır.
- **sort():** Listeyi sıralar. Orjinal liste değişir. Liste değiştirilmeden sıralanmak isteniyorsa `sorted(liste)` kullanılmalıdır.
- **reverse():** Elemanların sırasını tersine çevirir.
- **pop(indis):** Verilen indisteki elemanı siler ve geri döndürür. İndis verilmemişse son eleman için bu işlemi yapar.

Liste Metotları

```
liste = ['Ali', 'Ahmet', 'Mehmet']
liste.append('Burak') # sona Burak eklenir
liste.insert(0, 'Hasan') # başa Hasan eklenir
liste.extend(['Ayse', 'Fatma']) # listenin sonuna Ayse ve Fatma
↪  eklenir.
print(liste) # ['Hasan', 'Ali', 'Ahmet', 'Mehmet', 'Burak',
↪  'Ayse', 'Fatma']
print(liste.index('Ahmet')) # 2
liste.remove('Burak')
liste.pop(1) # Ali silinir
print(sorted(liste)) # ['Ahmet', 'Ayse', 'Fatma', 'Hasan',
↪  'Mehmet']
print(liste) # ['Hasan', 'Ahmet', 'Mehmet', 'Ayse', 'Fatma']
liste.sort() # Liste sıralanmış hali ile değişir
```

Şekil: Liste metotları

Liste Üreteçleri

```
#pozitif olanları alma
print([a for a in [-8, -4, -2, 0, 2, 4, 8] if a>0])
vec = [[1,2,3], [4,5,6], [7,8,9]]
#listeleri düzleştirme(flatten)
print([num for elem in vec for num in elem])
# 100'den küçük asal sayılar
asal = [p for p in range(2,100) if all(p%y!=0 for y in range(2,p))]
print(asal)
# 1000'den küçük mükemmel sayılar
mukemmel = [p for p in range(2,1000) if sum([d for d in range(1,p) if
↪  p%d==0])==p]
print(mukemmel)
# liste içinde tuple üretme
print([(x, "2'nin katı" if x%2==0 else "2'nin katı değil") for x in
↪  range(20)])
t=(x*2 for x in range(10)) # üreteç nesnesi
print(tuple(t)) # tuple üretme, üretildikten sonra kaybolur
t=(x**2 for x in range(10))
print(list(t)) # liste üretme
```

Çokuzlular(Tuple)

- Listelere benzer
- Değiştirilemez
- Fonksiyonlardan çoklu değer dönüşü için de kullanılırlar
- `()` veya `tuple()` ile başlatılabilir

Çokuzlular(Tuple)

```
cu = (1, 3.14, 'merhaba')
for el in cu: # Tüm elemanları yazar
    print(el)
print(cu[1]) # 3.14
cu[2] = 0 # Hata: tuple değiştirilemez
```

Şekil: Çokuzlular

Sözlükler

- Anahtar değer çiftlerinden oluşur
- Çok kullanılan bir veri türüdür
- `{}` veya `dict()` ile başlatılabilir

Sözlükler

```
sozluk ={'a': 'alpha', 'o':'omega', 'g':'gamma'}
print(sozluk) # {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
sozluk=dict()
sozluk['a']='alpha'
sozluk['o']='omega'
sozluk['g']='gamma'
print(sozluk) # {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}
# for döngüsü varsayılan olarak anahtarlarda dolaşır
for anahtar in sozluk:
    print(anahtar + ':' + sozluk[anahtar])
# Aşağıdakiler listeye dönüştürülebilir.
print(sozluk.keys()) # dict_keys(['a', 'o', 'g'])
print(sozluk.values()) # dict_values(['alpha', 'omega', 'gamma'])
# anahtar değer çiftleri üzerinde dolaşma
for anahtar, deger in sozluk.items():
    print(anahtar,deger)

del sozluk['a'] # silmek için kullanılır
```

Fonksiyonlar

- `def` anahtar kelimesi ile tanımlanır
- `return` ifadesi ile değer döndürülür
- Parantez içinde virgülle parametreler ayrılır
- Değişken sayıda parametre alabilirler
- Anahtar kelime argümanları ile anahtar, değer çiftlerini parametre olarak alabilirler

Fonksiyonlar

```
def fib(n): # fonksiyon tanımı
    a, b = 0, 1
    while a < n:
        print(a)
        a, b = b, a+b

fib(100) # fonksiyon çağrısı
```

Şekil: Fonksiyon örneği

Fonksiyonlar

```
def fonksiyon(sayi): # tanım
    if sayi<1000:
        return True, 'Sayı 1000\'den küçük'
    else:
        return False, 'Sayı 1000\'den büyük'

sonuc, metin = fonksiyon(150) # fonksiyon çağrısı
```

Şekil: Fonksiyon örneği

Fonksiyonlar

Varsayılan Argüman Değerleri

```
def fonksiyon(a1, a2='merhaba', a3='dünya!'):
    print('{} {} {}'.format(a1, a2, a3))

fonksiyon('Python')
fonksiyon('Python', 'hello', 'world!')
fonksiyon('Python', a3='programlama')
```

Şekil: Fonksiyon örneği

Fonksiyonlar

Değişken Sayıda ve Anahtar Kelime Argümanları

```
def fonk(arg1, *args, **kwargs):
    print(arg1)
    for arg in args:
        print(arg)
    for k in kwargs:
        print(k, ':', kwargs[k])

fonk('zorunlu')
fonk('deneme', '1. parametre', '2.', 3.14, yaz='python',
↪  anahtar='AE12FF')
""" Çıktı aşağıdaki şekildedir:
zorunlu
deneme
1. parametre
2.
3.14
yaz : python
anahtar : AE12FF
"""
```

Fonksiyonlar

Lambda İfadeler ve Yüksek Dereceli Fonksiyonlar

- Lambda ifadeler isimsiz fonksiyonlar oluşturmak için kullanılır
- Fonksiyonel programlama paradigması kullanılabilir
- Fonksiyonel programlamadaki map, filter, reduce işlemleri için fonksiyonlar vardır
- map işlemi sonucunda bir map nesnesi döner(liste değil)

Fonksiyonlar

Lambda İfadeler ve Yüksek Dereceli Fonksiyonlar

```
(lambda n: print(n))(5) #isimsiz fonksiyon oluşturma ve değer gönderme
(lambda x: print('çift' if x%2==0 else 'tek'))(13) # tek yazar

def cift(x):
    return x%2 == 0

print(list(map(cift, range(20)))) # True ve False'lardan oluşan liste
print(list(filter(cift, range(20)))) # 20'den küçük çift sayı listesi

def fonk(f, sayi):# fonksiyona başka fonksiyonu parametre olarak
↪   gönderme
    print(f(sayi))

fonk(cift, 17) # False yazar
fonk(lambda n: 'çift' if n%2==0 else 'tek', 20) # çift yazar
```

Şekil: Lambda İfadeler ve Yüksek Dereceli Fonksiyonlar

Fonksiyonlar

pass ifadesi

- Bir işlevi yoktur
- Yazımın hata vermemesi için kullanılır
  - Örn: Boş fonksiyon
  - Örn: Sonsuz döngü

```
while True:
    pass #sonsuz döngü

def bos_metod():
    pass #bir iş yapmaz
```

Şekil: pass ifadesi

Hata Yakalama

```
while True print('sonsuz döngü') # yazım hatası
y = 0
x = 1 / y # olağan dışı durum

try:
    x=1 / y
except ZeroDivisionError: # sıfıra bölme hatası oluşursa
    print('Sıfıra bölme hatası')

while True:
    try:
        x = int(input('Bir sayı girin: '))
        break
    except (ValueError, KeyboardInterrupt): #birden fazla hata
    ↪  algılama
        print('Geçerli bir sayı girmediniz')
    except: # diğer tüm hatalar
        print('Beklenmedik hata')
        raise
```

Nesneye Yönelik Programlama

- Sınıf tanımlamak için `class` anahtar kelimesi kullanılır
- Yapıcı(constructor) için `__init__` metodu yazılır
- Devralınan sınıfın üyelerine `super()` ile erişilebilir

Sınıf Tanımlama

```
class Dog:
    pass
>>> obj=Dog()
>>> obj
<__main__.Dog object at 0x1094fe1d0>
```

Şekil: Sınıf Tanımlama

Sınıf Tanımlama

```
class Dog:
    # Sınıf üyesi
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age

# Örnek(nesne) oluşturma
Rodger = Dog("Rodger", 2)
Tommy = Dog("Tommy", 4)
# Sınıf üyelerine erişme
print("Rodger is a {}".format(Rodger.__class__.attr1))
# Nesne üyelerine erişme
print("My name is {}".format(Tommy.name))
k = Dog() # Hata verir
```

Sınıf Tanımlama

Metot Oluşturma

```
class Dog:
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age
    # Nesne metodu
    def description(self):
        return f"{self.name} is {self.age} years old"
    # Başka bir nesne metodu
    def speak(self, sound):
        return f"{self.name} says {sound}"

# Nesne oluşturma
Rodger = Dog("Rodger", 2)
Tommy = Dog("Tommy", 4)
# Sınıf metotlarına erişme
Rodger.speak() # Kabukta değeri gösterir
Tommy.speak() # Ekrana yazdırmaz
```

Sınıf Tanımlama

```
class Dog:
    # Dog sınıfının diğer kısımları aynı

    # .description() metodunu __str__() ile değiştirelim
    def __str__(self):
        return f"{self.name} is {self.age} years old"

miles = Dog("Miles", 4)
print(miles) # 'Miles is 4 years old' yazar
```

Şekil: Sınıf Tanımlama

Sınıf Tanımlama

Kalıtım

```
class JackRussellTerrier(Dog):
    def speak(self, sound="Arf"):
        return f"{self.name} says {sound}"

miles = JackRussellTerrier("Miles", 4)
miles.speak() # 'Miles says Arf' yazar
```

Şekil: Sınıf Tanımlama

Büyülü Metotlar

- `__init__` ve `__new__`: Başlatıcı görevi vardır
- `__str__` ve `__repr__`: Nesneyi temsil eden metotlar
- `__cmp__`, `__eq__`, `__ne__`, `__lt__`, `__gt__`, `__le__`, `__ge__`: Karşılaştırma metotları

Sınıflar

Büyülü Metotlar

```
class Area:
    def __init__(self, height, width):
        self.height = height
        self.width = width
    def __eq__(self, other):
        if isinstance(other, Area):
            return self.height * self.width == other.height *      other.width
        else:
            return False
    def __ne__(self, other):
        return not self == other
    def __lt__(self, other):
        if isinstance(other, Area):
            return self.height * self.width < other.height * other.width
        else:
            return False
    def __gt__(self, other):
        if isinstance(other, Area):
            return self.height * self.width > other.height * other.width
        else:
            return False
    def __le__(self, other):
        return self == other or self < other
    def __ge__(self, other):
        return self == other or self > other
```

Şekil: Büyülü Metotlar

Sınıflar

Büyülü Metotlar

```
a1 = Area(7, 10)
a2 = Area(35, 2)
a3 = Area(8, 9)
print('Testing ==')
print(a1 == 'hello')
print(a1 == a2)
print(a1 == a3)
print('Testing !=')
print(a1 != 'hello')
print(a1 != a2)
print(a1 != a3)
print('Testing <')
print(a1 < 'hello')
```

```
print(a1 < a2)
print(a1 < a3)
print('Testing >')
print(a1 > 'hello')
print(a1 > a2)
print(a1 > a3)
print('Testing <=')
print(a1 <= 'hello')
print(a1 <= a2)
print(a1 <= a3)
print('Testing >=')
print(a1 >= 'hello')
print(a1 >= a2)
print(a1 >= a3)
```