Python Programming

Diziler(Sequences):
Dizeler(Strings),
Listeler(Lists),
Dosyalar(Files)

Sözcük değişkenleri:

```
a = "Merhaba Python"
print( a)
b = " ile programlama"
print(a+b)
```

- 'Merhaba Python ile programlama'

sözcük olan değiskeni sayıya çeviren *“int()”*, sayıyı sözcüğe çeviren *“str()”*, sözcüğün uzunluğunu veren *“len()”* verilebilir.

```
a=29; b=" 15"; print( a+int(b))
44

print(str(a)+b)
29 15

print(len(str(a)+b))
5
```

The String (Dize) Data Type

- Metin, programlarda Dize Veri Türüne göre temsil edilir.
- Dize, tırnak işaretleri (") veya kesme (') içine alınan bir karakter dizisidir.

```
str1="Hello"
str2='spam'
print(str1, str2)
```

Hello spam

```
type(str1)
```

<class 'str'>

```
type(str2)
```

<class 'str'>

The String Data Type

- Girdi olarak bir dize alma

```
firstName = input("Please enter your name: ")
print("Hello", firstName)
```

Please enter your name: John
Hello John

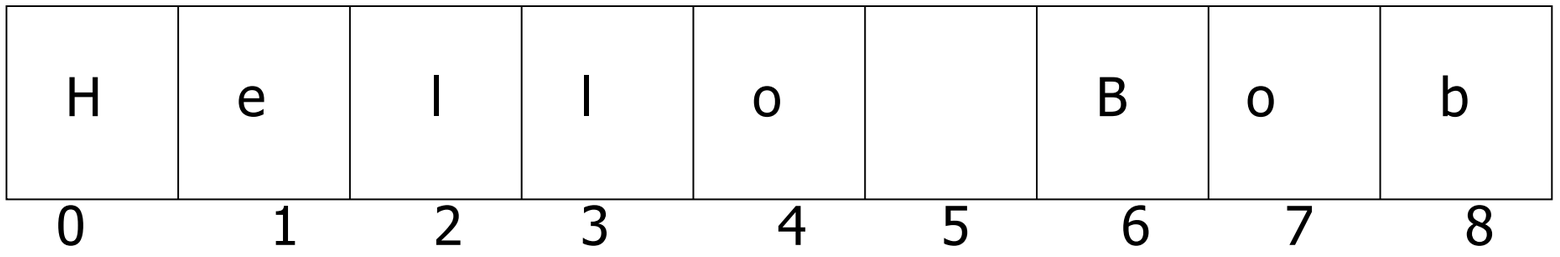
The String Data Type

- İndeksleme yoluyla bir dizideki tek tek karakterlere erişebiliriz.
- Bir dizideki konumlar, 0'dan başlayarak soldan numaralandırılır.
- Genel biçim <string>[<expr>] şeklindedir, burada expr değeri dizeden hangi karakterin seçileceğini belirler.
- **n karakterlik bir dizide saymaya 0 ile başladığımız için son karakter n-1 konumundadır.**
- Negatif indeksleri kullanarak sağ taraftan indeksleyebiliriz.

```
greet = "Cahit Karakuş"
print(greet)
print(greet[6])

print(greet[0], greet[2], greet[4])
C h t

x = 8
print(greet[x - 2])
K
print(greet[-1])
‘ş'
print(greet[-3])
‘k'
```



The String Data Type

- İndeksleme, daha büyük bir dizgeden tek bir karakter içeren bir dizge döndürür.
- Dilimleme (slicing) adı verilen bir işlemle alt dize adı verilen bitişik bir karakter dizisine de erişebiliriz.

- Slicing:
  <string>[<start>:<end>]
- Başlangıç ve bitiş indisleri olmalıdır.
- Dilim, konum başlangıcında başlayan alt dizeyi içerir ve konum sonuna kadar uzanır ancak konum sonunu içermez.

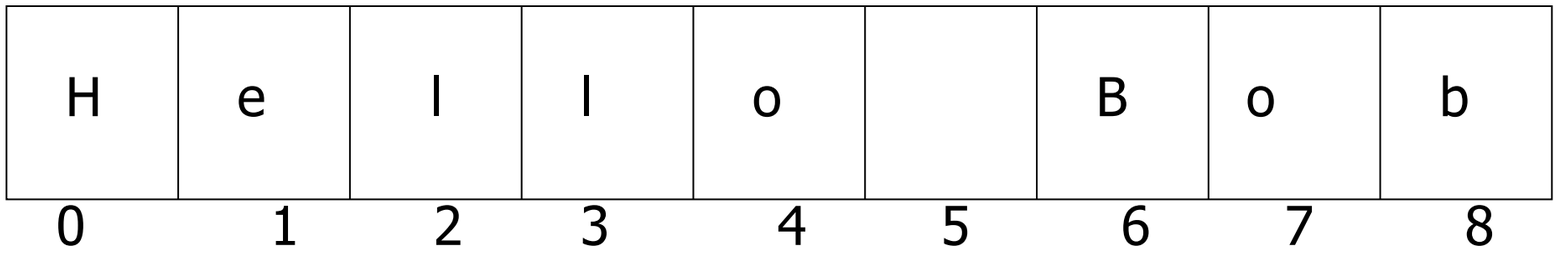
```
greet = “Hello Bob"
print(greet[0:6])
'Hello’

print(greet[5:9])
' Bob’

print(greet[:5])
'Hello’

print(greet[5:])
' Bob’

print(greet[:])
'Hello Bob‘
```



The String Data Type

- İfadelerden herhangi biri eksikse, dizenin başı veya sonu kullanılır.
- İki diziyi daha uzun bir dizide bir araya getirebilir miyiz?
- Birleştirme (Concatenation), iki diziyi birbirine “yapıştırır” (+)
- Tekrarlama(Repetition), bir dizenin kendisiyle birden fazla birleştirilmesiyle bir dizi oluşturur (*)
- *“len” işlevi bir dizenin uzunluğunu döndürür.*
- *"spam" + "eggs"*
- *'spameggs‘*

- *"Spam" + "And" + "Eggs"*
- *'SpamAndEggs‘*

- *3 * "spam"*
- *'spamspamspam‘*

*"spam" * 5*
'spamspamspamspamspam'

*(3 * "spam") + ("eggs" * 5)*
'spamspamspameggseggseggse ggseggs'

len("spam")
4

for ch in "Spam!":
*   print (ch, end=" ")*

S p a m !

The String Data Type

Operator	Meaning
+	Concatenation
*	Repetition
<string>[]	Indexing
<string>[:]	Slicing
len(<string>)	Length
for <var> in <string>	Iteration through characters

Basit Dize (String) İşleme

- Usernames on a computer system
  - First initial, first seven characters of last name

```
# get user's first and last names
first = input("Please enter your first name (all lowercase): ")
last = input("Please enter your last name (all lowercase): ")
# concatenate first initial with 7 chars of last name
print(first+last)
uname = first[0] + last[:]
print(uname)
oname = first[0] + last[:4]
print(oname)
```

CahitKarakuş
CKarakuş
CKara

Sequences

- Dizelerin gerçekten özel bir dizi olduğu ortaya çıktı, dolayısıyla bu işlemler diziler için de geçerli!

[1,2] + [3,4]

[1, 2, 3, 4]

[1,2]*3

[1, 2, 1, 2, 1, 2]

grades = ['A', 'B', 'C', 'D', 'F']

grades[0]

'A'

grades[2:4]

['C', 'D']

len(grades)

5

Listler

- Dizeler her zaman karakter dizileridir, ancak listeler keyfi değer dizileri olabilir. Listelerde sayılar, dizeler veya her ikisi birden olabilir!

  myList = [1, "Spam ", 4, "U"]

- Listeleri verileri tutan diziler seklinde düsünebiliriz. Olustururken köseli parantez kullanılır.
- list1 = ['java','c++','python']
- print(list1)
- ['java','c++','python']

- list1[1]
- 'c++‘

- list1 = list1 + [‘Matlab']
- print(list1)
- ['java','c++','python',‘Matlab’]

Strings, Lists, and Sequences

- Ayların bir listesini yapın:

months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]

- Diziden ayları çıkarmak için şu yapılır:

  monthAbbrev = months[n-1]

- Ayların satırının bir satırla örtüştüğüne dikkat edin. Python,] kapanışıyla karşılaşılıncaya kadar ifadenin tamamlanmadığını bilir.
- Liste 0'dan başlayarak indekslendiğinden, n-1 hesaplaması, ayrı bir adıma ihtiyaç duymadan print deyimine koymak için yeterince basittir.
- Programın bu sürümü, bir kısaltma yerine tüm ayın adını yazdıracak şekilde genişletilebilir!

```
# month2.py
#  A program to print the month name, given it's number.
#  This version uses a list as a lookup table.

def main():
    # months is a list used as a lookup table
    months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
    n = eval(input("Enter a month number (1-12): "))
    print ("The month abbreviation is", months[n-1] + ".")

main()
```

```
 months = ["January", "February", "March", "April", "May",
"June", "July", "August", "September", "October",
"November", "December"]
```

Strings, Lists, and Sequences

- Listeler değişkendir, yani değiştirilebilirler. Dizeler değiştirilemez.

```
myList = [34, 26, 15, 10]
myList[2]
15

myList[2] = 0
myList
[34, 26, 0, 10]

myString = "Hello World"
myString[2]
'l'
myString[2] = "p"

Traceback (most recent call last):
  File "<pyshell#16>", line 1, in -toplevel-  myString[2] = "p"
TypeError: object doesn't support item assignment
```

- Liste elemanları aralık vererek de kullanılabilir. Burada ilk eleman verilmezse 0 kabul edilir.
- ['java','c++','python','ruby']
- liste[2:3]
- ['c++','python']

- liste[:2]
- ['java','c++']

- liste[2:]
- ['python','ruby']

Liste

- "liste.count(eleman)",
- iki listeyi toplamak için "liste.extend(liste)",
- herhangi bir elemanın indisini döndürmek için "liste.index(eleman)",
- belli bir yere eleman eklemek için "liste.insert(indis,eleman)",
- listenin son elemanını çekmek için "liste.pop()",
- herhangi bir elemanı listeden çıkarmak için "liste.remove(eleman)",
- listeyi ters çevirmek için "liste.reverse()", listeyi sıralamak için "liste.sort()",
- listenin uzunluğunu döndürmek için "liste.len()" fonksiyonları verilebilir.
- Burada "liste" olusturulan listenin ismi, "eleman" ise listedeki elemanı temsil eder. Örneğin "liste" olarak yularıdaki örneklerde tanımladığımız liste değiskeni, eleman olarak da 'python' verilebilir.
- Örnek bir fonksiyon çağrısı:
- >>> liste.count('java')
- 1 olabilir.

Strings and Character Numbers

- Bilgisayarın içinde, dizeler tıpkı sayılar gibi 1'ler ve 0'lar dizisi olarak temsil edilir.
- Bir dizede, karakterlerin herbiri bir sayı olacak şekilde ikili sayılar dizisi olarak saklanır.
- Bilgisayarların ilk günlerinde, her üretici karakterler için kendi sayı kodlamasını kullandı.
- ASCII sistemi (Bilgi Değişimi için Amerikan Standart Kodu) 127 bitlik kodlar kullanır
- Python, Unicode'u destekler (100.000'den fazla karakter)

- “ord” fonksiyonu, tek bir karakterin sayısal (sıralı) kodunu döndürür. “chr” işlevi, sayısal bir kodu karşılık gelen karaktere dönüştürür

```
ord("A")
65
ord("a")
97
chr(97)
'a'
chr(65)
'A'
```

```
a=input(print ("Klavyeden bir tuşa basınız:"))
u1=ord(a)
print(u1)
```

```
b=input(print ("0 ile 255 arasında bir sayı giriniz:"))
u2=chr(int(b))
print(u2)
```

Other String Methods

- Bir dizi başka dize yöntemi vardır. Hepsini deneyiniz!
  - s.capitalize() – "s” 'nin yalnızca ilk karakteri büyük olan kopyası
  - s.title() – Copy of “s”; first character of each word capitalized
  - s.center(width) – Center “s” in a field of given width
  - s.count(sub) – Count the number of occurrences of sub in s
  - s.find(sub) – Find the first position where sub occurs in s
  - s.join(list) – Concatenate list of strings into one large string using s as separator.
  - s.ljust(width) – Like center, but s is left-justified
  - s.lower() – Copy of s in all lowercase letters
  - s.lstrip() – Copy of s with leading whitespace removed
  - s.replace(oldsub, newsub) – Replace occurrences of oldsub in s with newsub
  - s.rfind(sub) – Like find, but returns the right-most position
  - s.rjust(width) – Like center, but s is right-justified
  - s.rstrip() – Copy of s with trailing whitespace removed
  - s.split() – Split s into a list of substrings
  - s.upper() – Copy of s; all characters converted to uppercase

Input/Output as String Manipulation

- Dize verilerimizi çıktıya hazırlamak için genellikle bazı dize işlemleri yapmamız gerekir (“pretty it up”)
- Diyelim ki “05/24/2003” formatında bir tarih girmek ve “24 Mayıs 2003” çıktısını almak istiyoruz. Bunu nasıl yapabiliriz?
- Input the date in mm/dd/yyyy format (dateStr)
- Split dateStr into month, day, and year strings
- Ay dizesini bir ay numarasına dönüştürür
- Ay adını aramak için ay numarasını kullanır
- “Ay Gün, Yıl” biçiminde yeni bir tarih dizisi oluşturur.
- Yeni tarih dizesini çıkarır.

- İlk iki satır kolayca uygulanır!
-

```
dateStr = input("Enter a date (mm/dd/yyyy): ")
monthStr, dayStr, yearStr = dateStr.split("/")
```

- Tarih bir dizge olarak girilir ve ardından eğik çizgilerle bölünerek ve eş zamanlı atama kullanılarak üç değişkene "açılır".
- Sonraki adım: MonthStr'yi bir sayıya dönüştürür
- Örneğin "05"i 5 tamsayısına dönüştürmek için monthStr'deki “int” işlevini kullanabiliriz. (int("05") = 5)

Input/Output as String Manipulation

- Not: eval işe yarar, ancak baştaki 0’a dikkat!

```
>>> int("05")
5
>>> eval("05")
 Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>eval("05")
File "<string>", line 1
05
    ^
      SyntaxError: invalid token
```

- Python'da temel 8 (sekizlik) hazır değerler için önde gelen 0 kullanılırdı.

Input/Output as String Manipulation

months = [“January”, “February”, …, “December”]

monthStr = months[int(monthStr) – 1]

- Saymaya 0'dan başladığımız için, aydan bir çıkarmamız gerektiğini unutmayın.Şimdi çıktı dizesini birlikte birleştirelim!

```
print ("The converted date is:", monthStr, dayStr+",", yearStr)
```

- DayStr'ye bitiştirme ile virgülün nasıl eklendiğine dikkat edin!
- >>> main()
  Enter a date (mm/dd/yyyy): 01/23/2010
  The converted date is: January 23, 2010

Input/Output as String Manipulation

- Bazen bir sayıyı bir diziye dönüştürmek isteriz. “str” fonksiyonunu kullanabiliriz.

```
str(500)
'500'

value = 3.14
str(value)
'3.14'

print("The value is", str(value) + ".")
The value is 3.14.
```

- Değer bir dize ise, sonuna bir nokta ekleyebiliriz. Değer bir int ise ne olur?

```
>>> value = 3.14
>>> print("The value is", value + ".")
The value is

Traceback (most recent call last):
  File "<pyshell#10>", line 1, in -toplevel-
    print "The value is", value + "."
TypeError: unsupported operand type(s) for +: 'float' and
      'str'
```

Input/Output as String Manipulation

- Artık eksiksiz bir tip dönüştürme işlemleri setimiz var:

Function	Meaning
float(<expr>)	Convert expr to a floating point value
int(<expr>)	Convert expr to an integer value
str(<expr>)	Return a string representation of expr
eval(<string>)	Evaluate string as an expression

String Formatting

- String formatting is an easy way to get beautiful output!

Change Counter

Please enter the count of each coin type.

Quarters: 6

Dimes: 0

Nickels: 0

Pennies: 0

The total value of your change is 1.5

- Shouldn’t that be more like $1.50??

String Formatting

- “print” deyimini aşağıdaki gibi değiştirerek çıktımızı biçimlendirebiliriz:

  print("The total value of your change is ${0:0.2f}".format(total))

- Şimdi şöyle bir şey elde ederiz:
- The total value of your change is $1.50
- String, dize biçimi yöntemidir.

String Formatting

- <template-string>.format(<values>)
- {} within the template-string mark “slots” into which the values are inserted.
- Each slot has description that includes *format specifier* telling Python how the value for the slot should appear.

print("The total value of your change is ${0:0.2f}".format(total)

- The template contains a single slot with the description: 0:0.2f
- Form of description: <index>:<format-specifier>
- Index tells which parameter to insert into the slot. In this case, total.
- The formatting specifier has the form: <width>.<precision><type>
- f means "fixed point" number
- <width> tells us how many spaces to use to display the value. 0 means to use as much space as necessary.
- <precision> is the number of decimal places.

String Formatting

"Hello {0} {1}, you may have won ${2}" .format("Mr.", "Smith", 10000)
'Hello Mr. Smith, you may have won $10000'

'This int, {0:5}, was placed in a field of width 5'.format(7)
'This int,   7, was placed in a field of width 5'

'This int, {0:10}, was placed in a field of witdh 10'.format(10)
'This int,   10, was placed in a field of witdh 10'

'This float, {0:10.5}, has width 10 and precision 5.'.format(3.1415926)
'This float,   3.1416, has width 10 and precision 5.'

'This float, {0:10.5f}, is fixed at 5 decimal places.'.format(3.1415926)
'This float,   3.14159, has width 0 and precision 5.'

String Formatting

- Genişlik gerekenden daha genişse, sayısal değerler varsayılan olarak sağa dayalıdır ve dizeler sola dayalıdır.
- You can also specify a justification before the width.

```
"left justification: {0:<5}.format("Hi!")
'left justification: Hi!  '
"right justification: {0:>5}.format("Hi!")
'right justification:   Hi!'
"centered: {0:^5}".format("Hi!")
'centered:  Hi! '
```

Better Change Counter

- Kayan noktalı sayılar hakkında artık bildiklerimizle, onları parasal bir durumda kullanmaktan rahatsız olabiliriz.Bu sorunun üstesinden gelmenin bir yolu, int veya uzun int kullanarak sent cinsinden paranın izini sürmek ve çıktı alındığında onu dolar ve sente dönüştürmektir.

- If total is a value in cents (an int),
  dollars = total//100
  cents = total%100
- Cents is printed using width 0>2 to right-justify it with leading 0s (if necessary) into a field of width 2.
- Thus 5 cents becomes '05'

Better Change Counter

```
# change2.py
#   A program to calculate the value of some change in
      dollars.
#   This version represents the total cash in cents.

def main():
    print ("Change Counter\n")

    print ("Please enter the count of each coin type.")
    quarters = eval(input("Quarters: "))
    dimes = eval(input("Dimes: "))
    nickels = eval(input("Nickels: "))
    pennies = eval(input("Pennies: "))
    total = quarters * 25 + dimes * 10 + nickels * 5 +
       pennies

    print ("The total value of your change is ${0}.{1:0>2}"
           .format(total//100, total%100))
```

```
main()
Change Counter
Please enter the count of each coin type.
Quarters: 0
Dimes: 0
Nickels: 0
Pennies: 1
The total value of your change is $0.01
```

```
main()
Change Counter
Please enter the count of each coin type.
Quarters: 12
Dimes: 1
Nickels: 0
Pennies: 4
The total value of your change is $3.14
```

Files: Multi-line Strings

- Dosya, ikincil bellekte (disk sürücüsü) saklanan bir veri dizisidir.
- Dosyalar herhangi bir veri türünü içerebilir, ancak üzerinde çalışılması en kolay olanı metindir.
- Bir dosya genellikle birden fazla metin satırı içerir.
- Python, satır sonlarını işaretlemek için standart yeni satır karakterini (\n) kullanır.

- This is exactly the same thing as embedding \n in print statements.
- Remember, these special characters only affect things when printed. They don’t do anything during evaluation.

- Hello
  World
  Goodbye 32
- When stored in a file:
  Hello\nWorld\n\nGoodbye 32\n