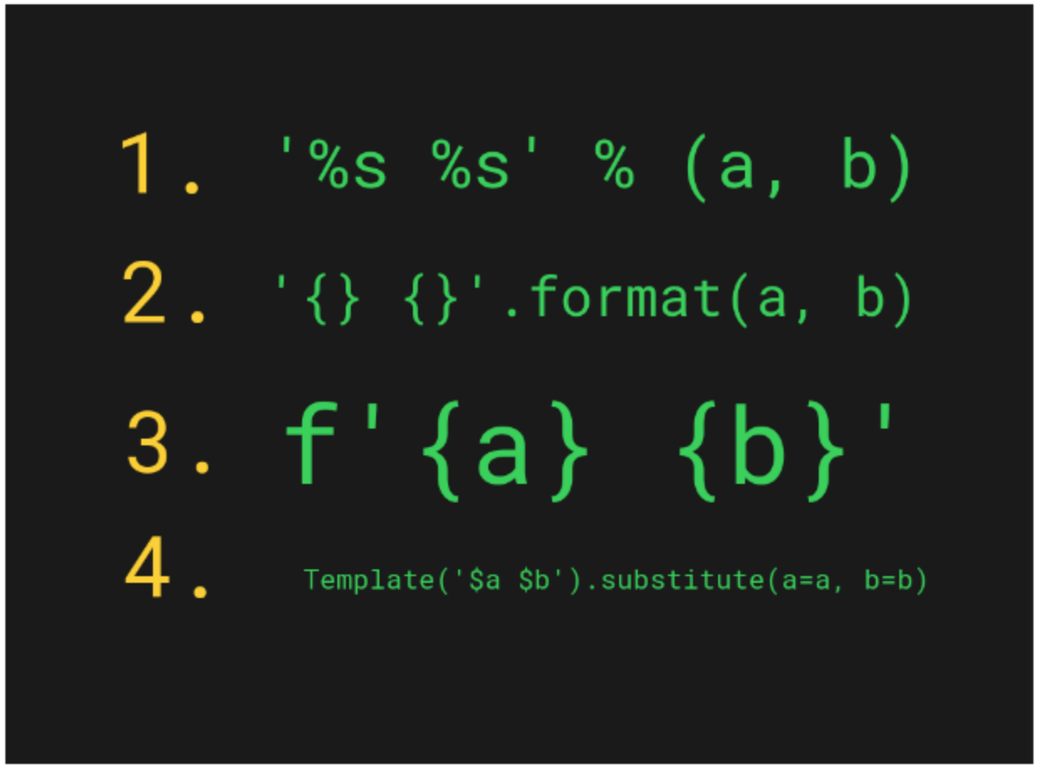
Python Programming

Giriş / Çıkış

Veri Tipleri Biçimlendirme

Format Strings in Python



4 ways to do the same thing!

Veri Tipleri Biçimlendirme

- Python 3.6, biçimlendirilmiş dize hazır değerleri veya "f-dizeleri" adı verilen yeni bir dize biçimlendirme yaklaşımı ekledi. Dizeleri biçimlendirmenin bu yeni yolu, dize sabitlerinin içine katıştırılmış Python ifadelerini kullanmanıza izin verir.

```
a = 5
b = 10
print(f"Five plus ten is {a + b} and not {2 * (a + b)}.")
print(f'Five plus ten is {a + b} and not {2 * (a + b)}.')
```

Five plus ten is 15 and not 30.

Veri Tipleri Biçimlendirme

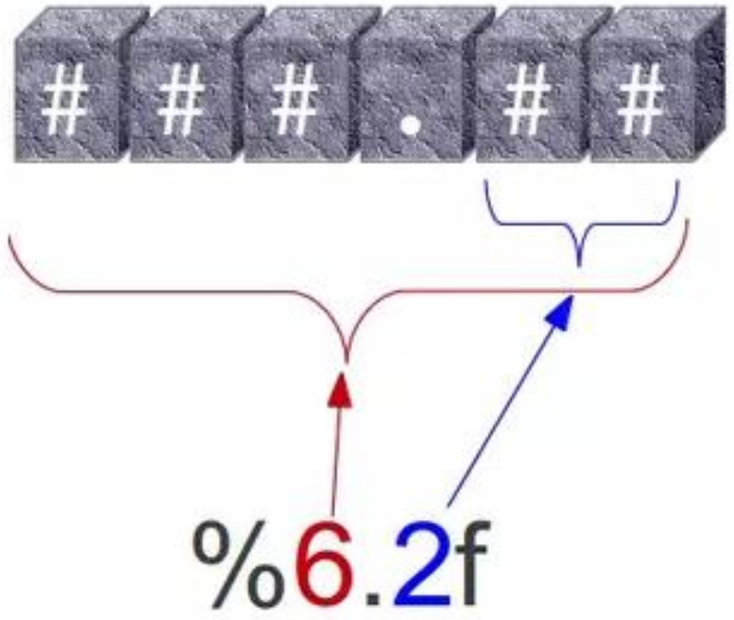
```
i = 3
num = 25
field_size = 7
print(f'Number {i}: {num:{field_size}.2f}'.format(i=i, num=num, field_size=field_size))
```

Değikenleri Biçimlendirme

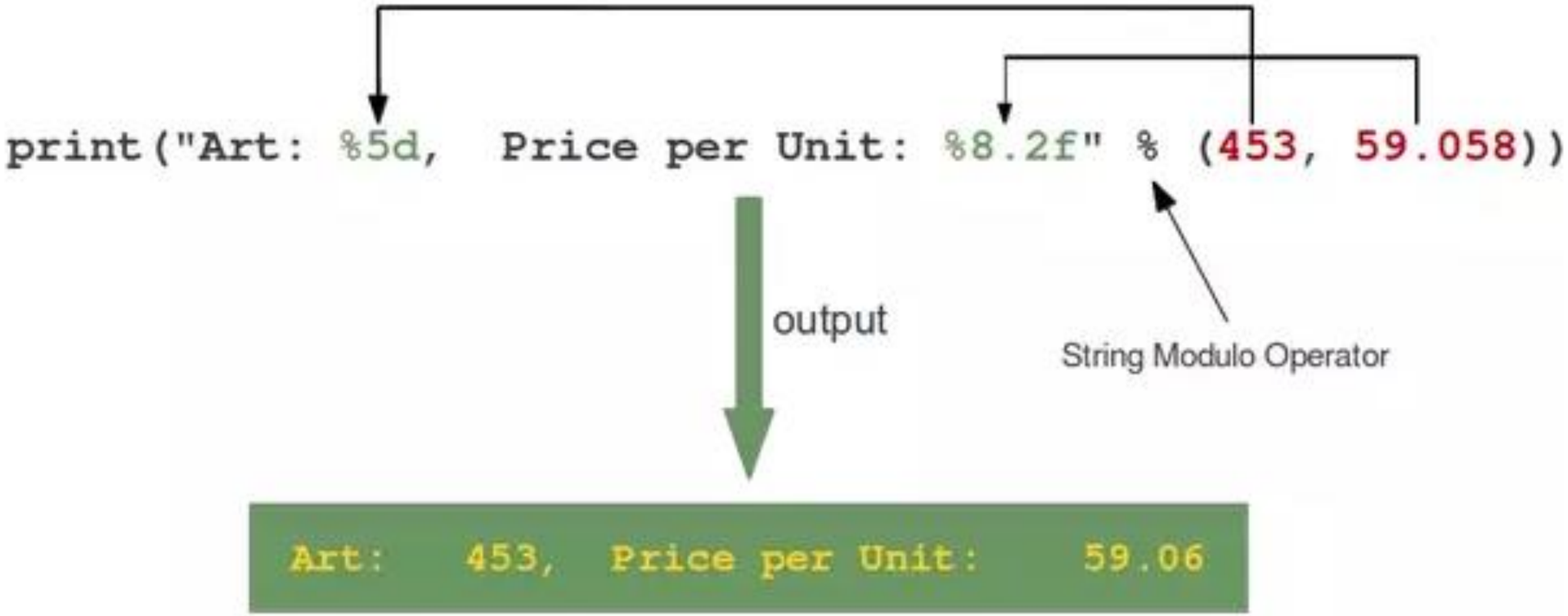
```
q = 459
p = 0.098
print(q, p, p * q)
```

Değikenleri Biçimlendirme

```
%[flags][width][.precision]type
```



Değikenleri Biçimlendirme



Değikenleri Biçimlendirme

print("%10.3e"% (356.08977))

3.561e+02

print("% 2d"% (42))

42

s = "Price: $ %8.2f"% (356.08977)

Print(s)

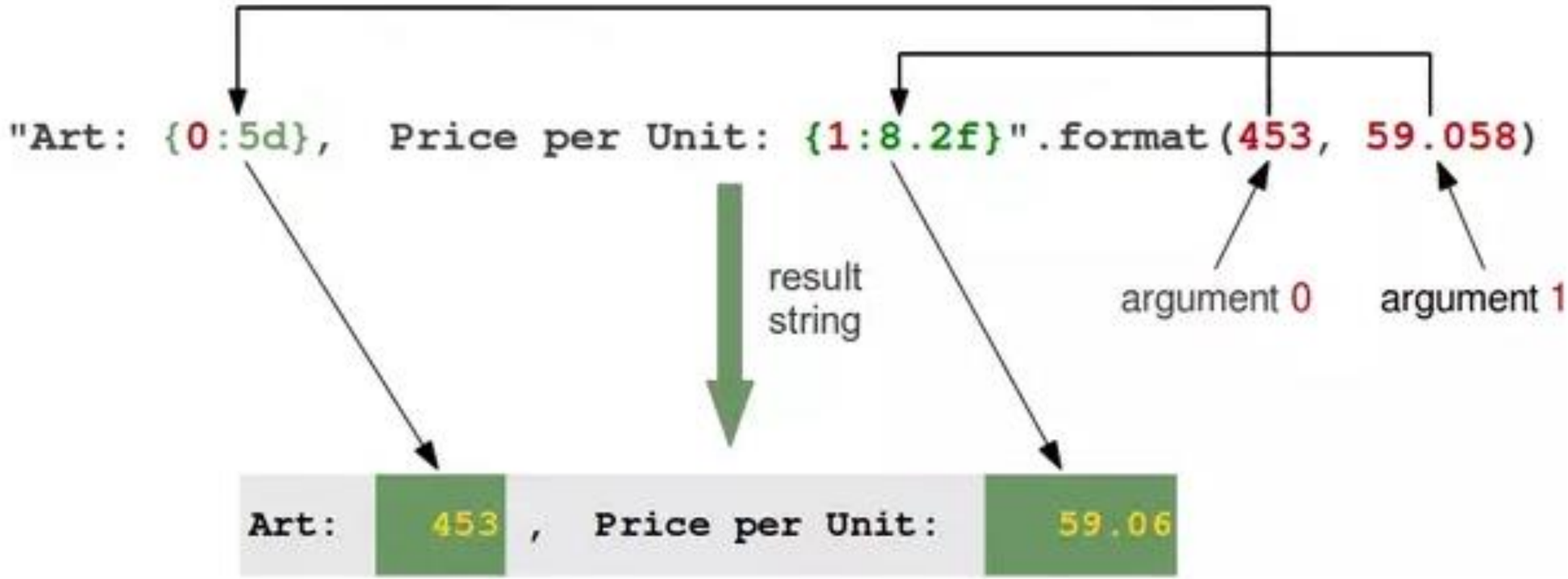
Price: $ 356.09

s = "Price: $ %10.5f"% (356.08977)

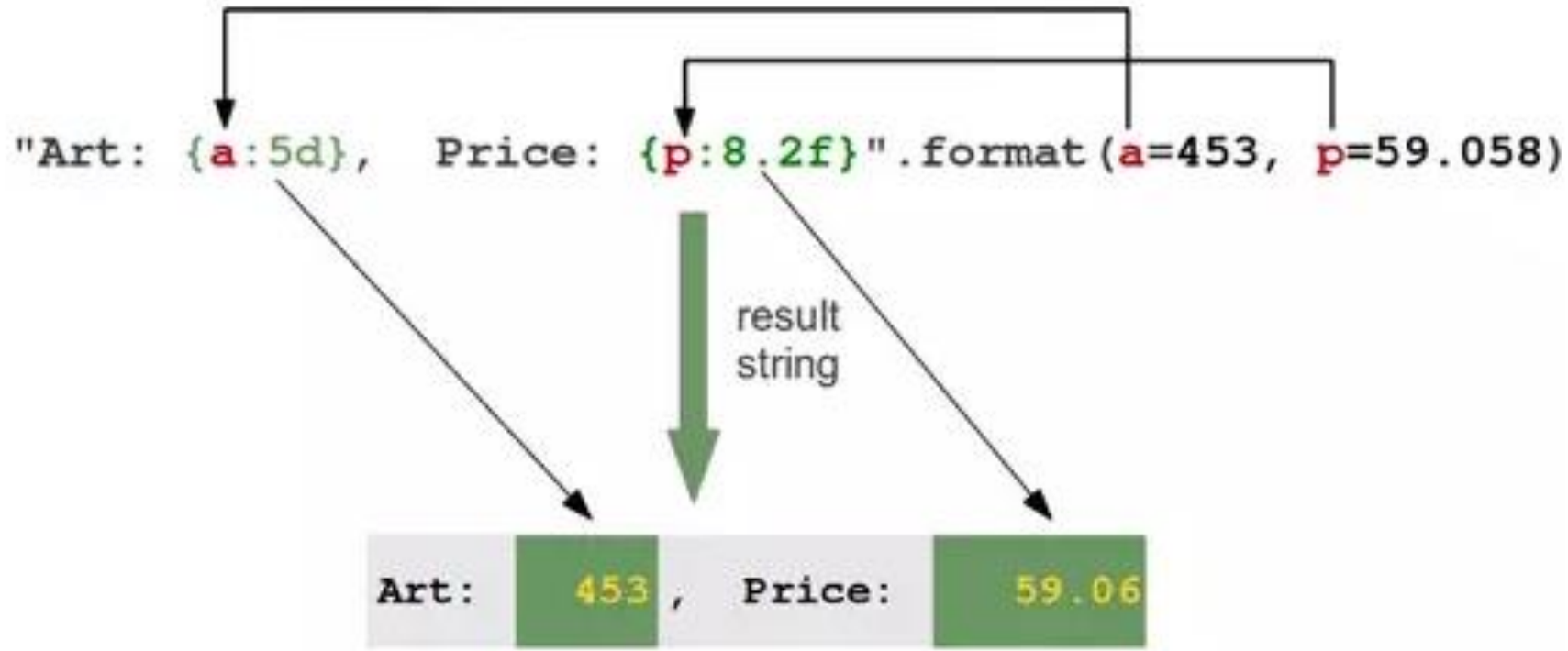
print(s)

Price: $ 356.08977

Değikenleri Biçimlendirme



Değikenleri Biçimlendirme



format()

```
for x in range(1, 5):
    print ('{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x))
```

```
Print ( ‘PI : {0:.3f}'.format(math.pi))
>> PI: 3.142
```

- **!s (str()) and !r (repr())**

```
 print ‘PI : {!r}'.format(math.pi)
>> PI: 3.141592653589793
 print ‘PI : {!s}'.format(math.pi)
>> PI: 3.14159265359
```

Print set of variables

print)'{1} and {0}'.format('spam', 'eggs'))

>> eggs and spam

print '{dollar} dollars and {cent} cents'.format(dollar='ten', cent='forty')

>> ten dollars and forty cents

- Print table

```
table = {'Sjoerd': 4127, 'Jack': 4098, 'Dcab': 7678}
    for name, phone in table.items():
            print '{0:10} ==> {1:10d}'.format(name, phone)
```

```
Jack       ==>       4098
Dcab       ==>       7678
Sjoerd     ==>       4127
```

```
table = {'Sjoerd': 4127, 'Dcab': 8637678}
print ('Sjoerd: {0[Sjoerd]:d}; Dcab: {0[Dcab]:d}'.format(table))
```

>> Jack: 4098; Sjoerd: 4127; Dcab: 8637678

```
print 'Sjoerd: {Sjoerd:d}; Dcab: {Dcab:d}'.format(**table)
```

- String Formatting

  veg = “apple”
  color = “red”
  print (veg + “ is “ + color) >> apple is red
  print (“%s is %s” % (veg, color)) >> apple is red

  count= 6
  print (“There are %d %s” % (count, veg)) >> There are 6 apple
  print (“There are “ + count + veg) ERROR!!

- Numbers Formatting

  print ("Today's stock price: %f" % 50.4625)
  >> Today's stock price: 50.462500

  print ("Today's stock price: %.2f" % 50.4625)
  >> Today's stock price: 50.46

  print ("Change since yesterday: %+.2f" % 1.5)
  >> Change since yesterday: +1.50

Örnek-1

```
name = input("Enter your name: ")
age = input("Enter your age: ")

print(f"Hello {name}")
print(f"You are {age} years old")
```

f-string, string içindeki ifadeleri değerlendirmemizi sağlar. Değerlendirmek için ifadeyi { } içine koymanız yeterlidir. f-string, programın çalışma zamanında değerlendirir. Size zaman ve kod kazandıran mükemmel bir özelliktir.

Giriş string yani karekter dizisi olarak alınır.
Tanımlama yapılması gerekiyor.

```
## arithmetic expression
print(f"{5 * 5}")
```

```
name = input("Enter your name: ")
age = input("Enter your age: ")

age = age + 1
print(f"Hello {name}")
print(f"You are {age} years old")
```

```
name = input("Enter your name: ")
age = input("Enter your age: ")
age = int(age)
age = age + 1
print(f"Hello {name}")
print(f"You are {age} years old")
```

```
name = input("Enter your name: ")
age = int(input("Enter your age: "))
age = age + 1
print(f"Hello {name}")
print(f"You are {age} years old")
```

```
name = input("Enter your name: ")
age = float(input("Enter your age: "))
age = age + 1
print(f"Hello {name}")
print(f"You are {age} years old")
```

Örnek-2

```python
adjective1 = input("Enter an adjective: ")
noun = input("Enter a noun: ")
adjective2 = input("Enter an adjective: ")
verb = input("Enter a verb: ")
adjective3 = input("Enter an adjective: ")

print(f"Today I went to a {adjective1} zoo.")
print(f"In an exhibit, I saw {noun}")
print(f"{noun} was {adjective2} and {verb}ing")
print(f"I was {adjective3}")
```

```
Enter an adjective: sus
Enter a noun: Mark Zuckerburg
Enter an adjective: berserk
Enter a verb: screech
Enter an adjective: amazed
```

```
Today I went to a sus zoo.
In an exhibit, I saw Mark Zuckerburg
Mark Zuckerburg was berserk and screeching
I was amazed
```

Berserk: çılgın
Screech: çılgına dönmüş
Amaze: hayrete düşmek

Örnek-3

```
length = input("Enter the length of a rectangle: ")
width = input("Enter the width of a rectangle: ")

area = length * width

print(f"The area is: {area}cm^2")
```

```
Traceback (most recent call last):
  File "C:\Users\HP\PycharmProjects\practice\main.py", line 5, in <module>
    area = length * width
TypeError: can't multiply sequence by non-int of type 'str'
```

```
length = float(input("Enter the length of a rectangle: "))
width = float(input("Enter the width of a rectangle: "))

area = length * width

print(f"The area is: {area}cm^2")
```

```
length = float(input("Enter the length of a rectangle: "))
width = float(input("Enter the width of a rectangle: "))
height = float(input("Enter the height of a rectangle: "))

volume = length * width * height

print(f"The volume is: {volume}cm^3")
```

Örnek-4

```
item = input("What item would you like to buy?: ")
price = float(input("What is the price?: "))
quantity = int(input("How many would you like?: "))

total = price * quantity

print(f"You have bought {quantity} x {item}/s")
print(f"Your total is: ${total}")
```

```
What item would you like to buy?: pizza
What is the price?: 4.99
How many would you like?: 9
```

```
You have bought 9 x pizza/s
Your total is: $44.910000000000004
```

```
item = input("What item would you like to buy?: ")
price = float(input("What is the price?: "))
quantity = int(input("How many would you like?: "))

total = price * quantity

print(f"You have bought {quantity} x {item}/s")
print(f"Your total is: ${round(total, 2)}")
```

```
What item would you like to buy?: pizza
What is the price?: 4.99
How many would you like?: 9
```

```
What is the price?: 4.99
How many would you like?: 9
You have bought 9 x pizza/s
Your total is: $44.91
```

Output Format

str( )

same representation but String and Floating point number

•(aynı gösterim ancak Dize ve Kayan nokta sayısı)

```
a=0.24
str(a)                  ‘0.24’
repr (a)                ‘0.23999999999999999’

s = 'Hello, world.'
str(s)                  'Hello, world.'
repr(s)                 "'Hello, world.'"
print  ( repr(s) )      ‘Hello, word.’
```

rjust() , ljust(), center()

- Verilen uzunluktaki bir dizide iki yana dizilmiş dizeyi döndürür

```
str = “hello world”
print str.rjust (20)

>>          hello world

print str.rjust(20, ‘!’)
>> !!!!!!!!!hello world
```

zfill()

- Soldaki sayısal bir dizeyi sıfırlarla doldurur

```
'-3.14'.zfill(7)                    '-003.14'
```

repr( )

```
for x in range(1, 5):
    print repr(x).rjust(2), repr(x*x).rjust(3), print repr(x*x*x).rjust(4)
```

```
1   1   1
2   4   8
3   9  27
4  16  64
5  25 125
```

Reading and Writing Files

Dosya Okuma - Yazma

- Dosya okuma(r), yazma(w), ekleme(a), okuma ve yazma(rw) olmak üzere 4 modda açılabilir.
- Bazı dosya islemleri sunlardır:
- * write(x) : x sözcüğü dosyaya yazılır.
- * readline() : Dosyadan bir satır okunur.
- * readlines() : Dosyanın tamamı bir listeye okunur.
- * read(x) : Dosyadan x byte okuma yapılır.
- * read() : Dosyanın tamamı bir sözcüğe okunur.
- * close() : Dosya kapatılır.

Reading and Writing Files

open()

- open(filename, mode)

  f = open('/cs/PA', 'w')

  mode:

  ‘r’ read

  ‘w’ write *file with same name will be erased*

  ‘a’ append(eklemek)

  ‘r+’ read and write

  ‘b’ open file in binary mode (on Windows)

- dosyam = open('./dosya.txt', 'w')
- dosyam.write("ilk satır\n")
- dosyam.write("ikinci satır\n")
- dosyam.close()

Methods of File Objects

- **f.read(size)**

  f.read() **read the entire file and return string**
  f.read(1) **read one character at a time**

- **f.readline()**

  reads a single line from the file until it reaches ‘\n’

- **f.readlines()**

  Read all lines and returns a list of all lines

  >>['This is the first line of the file.\n', 'Second line of the file\n']

- **f.write(string)**
  write the contents of string to the file
  f.write(‘first line\n’)

  make numbers to string before writing
  s = str (0.24)
  f.write(s)

- **f.tell()**

  dosyanın başından itibaren bayt cinsinden ölçülen dosyadaki geçerli konumu döndürür

- **f.close()**

  dosya ve dosya nesnesi yeniden açılmadıkça kullanılamaz

- **f.seek(offset, from)**

  from: 0: beginning of file

  1: current position

  2: end of file

```
f = open('/tmp/workfile', 'r+')
f.write('0123456789abcdef')
f.seek(5)        # Go to the 6th byte in the file
f.read(1)        '5'
f.seek(-3, 2)     # Go to the 3rd byte before the end
f.read(1)        'd'
```

Pickle Module

```
selfref_list = [1, 2, 3]
output = open('data.pkl', 'wb')
pickle.dump(selfref_list, output)
data1 = pickle.load(output)
```

Pickle içeren bir dosyayı okurken, dosyayı ikili modda açmalısınız çünkü ASCII veya ikili formatın kullanıldığından emin olamazsınız.

Varolan bir dosyaya yazma

- Var olan bir dosyaya yazmak için open() fonksiyonuna bir parametre eklenir.

- "a" - Append - Ekle - dosyanın sonuna eklenir
- "w" - Write - Yaz - mevcut içeriğin üzerine yazar

- Örnek: “Demofile.txt” dosyasını açın ve dosyaya içerik ekleyin:

```
f = open("demofile.txt", "a")
f.write("Now the file has one more line!")
```

- Örnek: “Demofile.txt” dosyasını açın ve içeriğin üzerine yazın:

```
f = open("demofile.txt", "w")
f.write("Woops! I have deleted the content!")
```

- Not: “w” yöntemi tüm dosyanın üzerine yazacaktır.

Yeni Dosya Oluşturma

- Python’da yeni bir dosya oluşturmak için, aşağıdaki parametrelerden biriyle open() fonksiyonunu kullanıılır:
- "x" - Create - Oluştur - bir dosya oluşturur, dosya mevcutsa bir hata döndürür.
- "a" - Append - Ek - belirtilen dosya mevcut değilse bir dosya oluşturur.
- "w" - Write - Yaz - belirtilen dosya mevcut değilse bir dosya oluşturur.

- Örnek:
- “myfile.txt” adlı bir dosya oluşturun:

  f = open("myfile.txt", "x")
- Sonuç: yeni boş bir dosya oluşturuldu!

- Örnek:
- Mevcut değilse yeni bir dosya oluşturun:

  f = open("myfile.txt", "w")

Python Dosya Silme

- Bir dosyayı silmek için, OS modülünün içe aktarılması ve os.remove() komutunun çalıştırılması gerekir:

- Örnek:
- “demofile.txt” dosyasını kaldırın:

```
import os
os.remove("demofile.txt")
```

Dosya var mı kontrol edilmesi

- Bir hata oluşmasını önlemek için, dosyayı silmeye çalışmadan önce mevcut olup olmadığını kontrol etmek istenebilir:

- Örnek:
- Dosyanın var olup olmadığını kontrol edin, ardından silin:

```
import os
if os.path.exists("demofile.txt"):
    os.remove("demofile.txt")
    else
    print("Dosya mevcut değil")
```

Klasör Silme

- Tüm bir klasörü silmek için os.rmdir() metodunu kullanın:

- Örnek:
- “myfolder” klasörünü kaldırın:

```
import os
os.rmdir("myfolder")
```

Örnek-5

- Çoktan Seçmeli test sınavı hazırlansın. Sorular toplam 5 soru ve 4şık seçmeli olacak.
- Python’da yazılım yazılmaya başlandığında soru kümesi, şıklar, cevaplar, skor gibi başlıklar olacaktır.

```
# Python quiz game

questions = ()

options = ((),(),(),(),())

answers = ()
guesses = []
score = 0
question_num = 0
```

```
# Python quiz game

questions = ("How many elements are in the periodic table?: ",
             "Which animal lays the largest eggs?: ",
             "What is the most abundant gas in Earth's atmosphere?: ",
             "How many bones are in the human body?: ",
             "Which planet in the solar system is the hottest?: ")

options = ((),(),(),(),())

answers = ()
guesses = []
score = 0
question_num = 0
```

```python
# Python quiz game

questions = ("How many elements are in the periodic table?: ",
             "Which animal lays the largest eggs?: ",
             "What is the most abundant gas in Earth's atmosphere?: ",
             "How many bones are in the human body?: ",
             "Which planet in the solar system is the hottest?: ")

options = (("A. 116", "B. 117", "C. 118", "D. 119"),
           ("A. Whale", "B. Crocodile", "C. Elephant", "D. Ostrich"),
           ("A. Nitrogen", "B. Oxygen", "C. Carbon-Dioxide", "D. Hydrogen"),
           ("A. 206", "B. 207", "C. 208", "D. 209"),
           ("A. Mercury", "B. Venus", "C. Earth", "D. Mars"))

answers = ("C", "D", "A", "A", "B")
guesses = []
```

```
for question in questions:
    print("----------------------")
    print(question)
    for option in options[question_num]:
        print(option)

    |
    question_num += 1
```

Her for çevriminden sonra çalıştırınız ve sonucu görünüz.

```
guess = input("Enter (A, B, C, D): ").upper()
guesses.append(guess)
if guess == answers[question_num]:
    score += 1
    print("CORRECT!")
else:
    print("INCORRECT!")
    print(f"{answers[question_num]} is the correct answer")
question_num += 1
```

```
print("----------------------")
print("       RESULTS        ")
print("----------------------")
```

```
print("answers: ", end="")
for answer in answers:
    print(answer, end=" ")
print()

print("guesses: ", end="")
for guess in guesses:
    print(guess, end=" ")
print()
```

```
score = int(score / len(questions) * 100)
print(f"Your score is: {score}%")
```

File Processing

Dosya İşleme

- Bir dosyayı açma işlemi, diskteki bir dosyayı bellekteki bir nesneyle ilişkilendirmeyi içerir.Bu nesneyi manipüle ederek dosyayı manipüle edebiliriz.
  - Read from the file
  - Write to the file
- Dosya ile işiniz bittiğinde, kapatılması gerekir.
- Dosyanın kapatılması, dosya için bekleyen tüm işlemlerin ve diğer muhasebe işlemlerinin tamamlanmasına neden olur.
- Bazı durumlarda, bir dosyanın düzgün şekilde kapatılmaması veri kaybına neden olabilir.
  - Bir dosyayı kelime işlemciye okuma
  - dosya açıldı
  - İçerik RAM'e okunur
  - dosya kapatıldı
- Dosyadaki değişiklikler diskte değil bellekte saklanan kopyada yapılır.

File Processing

- Bir kelime işlem dosyasını kaydetme
  - Diskteki orijinal dosya, yazmaya izin verecek bir modda yeniden açılır (bu aslında eski içeriği siler)
  - Dosya yazma işlemleri, belgenin bellekteki sürümünü diske kopyalar
  - dosya kapatılır.

- Python'da metin dosyalarıyla çalışma
  - <filevar> = open(<name>, <mode>) açık işlevini kullanarak bir disk dosyasını bir dosya nesnesiyle ilişkilendirilir.
  - Ad, diskteki gerçek dosya adına sahip bir dizedir. Dosyayı okuyup yazmadığımıza bağlı olarak mod ya 'r' ya da 'w'dir.
  - Dosya = open("numbers.dat", "r")

File Methods

- <file>.read() – returns the entire remaining contents of the file as a single (possibly large, multi-line) string
- <file>.readline() – returns the next line of the file. This is all text up to *and including* the next newline character
- <file>.readlines() – returns a list of the remaining lines in the file. Each list item is a single line including the newline characters.

File Processing

```
# printfile.py
#     Prints a file to the screen.

def main():
    fname = input("Enter filename: ")
    infile = open(fname,'r')
    data = infile.read()
    print(data)

main()
```

- First, prompt the user for a file name
- Open the file for reading
- The file is read as one string and stored in the variable data

File Processing

- readline can be used to read the next line from a file, including the trailing newline character
- ```
  infile = open(someFile, "r")
  for i in range(5):
      line = infile.readline()
      print line[:-1]
  ```
- This reads the first 5 lines of a file
- Slicing is used to strip out the newline characters at the ends of the lines
  ```

File Processing

- Another way to loop through the contents of a file is to read it in with readlines and then loop through the resulting list.
- infile = open(someFile, "r")
  for line in infile.readlines():
  	# Line processing here
  infile.close()

- Python treats the file itself as a sequence of lines!
- Infile = open(someFile, "r")
  for line in infile:
  	# process the line here
  infile.close()

File Processing

- Opening a file for writing prepares the file to receive data
- If you open an existing file for writing, you wipe out the file's contents. If the named file does not exist, a new one is created.
- Outfile = open("mydata.out", "w")
- print(<expressions>, file=Outfile)

Example Program: Batch Usernames

- *Batch* mode processing is where program input and output are done through files (the program is not designed to be interactive)
- Let's create usernames for a computer system where the first and last names come from an input file.

Example Program: Batch Usernames

```
# userfile.py
#    Program to create a file of usernames in batch mode.

def main():
    print ("This program creates a file of usernames from a")
    print ("file of names.")

    # get the file names
    infileName = input("What file are the names in? ")
    outfileName = input("What file should the usernames go in? ")

    # open the files
    infile = open(infileName, 'r')
    outfile = open(outfileName, 'w')
```

Example Program: Batch Usernames

```
# process each line of the input file
for line in infile:
    # get the first and last names from line
    first, last = line.split()
    # create a username
    uname = (first[0]+last[:7]).lower()
    # write it to the output file
    print(uname, file=outfile)

# close both files
infile.close()
outfile.close()

print("Usernames have been written to", outfileName)
```

Example Program: Batch Usernames

- Things to note:
  - It's not unusual for programs to have multiple files open for reading and writing at the same time.
  - The lower method is used to convert the names into all lower case, in the event the names are mixed upper and lower case.

Tüp(tuple) Değişkenleri

- Tüpleri içeriği değistirilemeyen listeler olarak düsünebilirsiniz. Parantez () kullanarak tanımlanırlar.
- >>> diller = ('c','python')
- >>> diller[1]
- 'python'