برنامه سازی پیشرفته
(کالکشن ها)

صادق اسکندری – دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

به انواع اطلاعاتی که قابل پردازش توسط زبان برنامه نویسی باشند، نوع داده گفته میشود.

یک نوع داده عبارت است از مجموعه ای از مقادیر به همراه مجموعه ای از عملگرها بر روی آن مقادیر

Data type = set of values (domain) + set of operators

Integer = Z + {+,*,/, ...}

انواع داده در پایتون

- انواع داده درون ساخت (Built-in data type)

Numbers, Strings, Lists, Dictionaries, Tuples, Files, Sets,

- انواع داده کلاسی

Student, Teacher, Car, TV, ...

کالکشن ها: رشته (String)

یک کالکشن مرتب از کاراکتر ها که به منظور ذخیره و پردازش داده های متنی مورد استفاده قرار می گیرد.

تفاوت رشته ها در پایتون با زبان C++:

C++:

کاراکتر (char) یک نوع داده اولیه ..... رشته یک آرایه از کاراکتر ها

پایتون:

رشته یک نوع داده اولیه ..... کاراکتر یک رشته با طول ۱ (نوع داده کاراکتر نداریم)

لیترال های رشته ای (نحوه نمایش مقادیر رشته ای در کد)

Single quotes: `'spam'`

Double quotes: `"spam"`

Escape sequences: `"s\tp\na\\m"`

Raw sequences: `r"s\tp\na\\m"`

Triple quotes: `'''... spam ...''', """... spam ..."""`

Single quotes و Double quotes معادل یکدیگر هستند.

‘spam’ ≡ “spam”

در صورتی که تعدادی رشته کنار یکدیگر بیایند، پایتون آنها را با یکدیگر ترکیب می کند.

```
1  title = "This " 'is ' "Python Programming"
2  print(title)
```

```
This is Python Programming
```

برای نمایش برخی کاراکترهای خاص، از دنباله فرار (Escape Sequence) استفاده می کنیم.

```
1 title = '\a this is a python course:)'
2 print(title)
```

```
this is a python course:)
```

```
1 title = 'this\n \t\tis a python course:)'
2 print(title)
```

```
this
                is a python course:)
```

```
1  title = 'this is a\b python \bcourse:)'
2  print(title)
```

```
this is  pythoncourse:)
```

```
1  title = 'this is a python \rcourse:)'
2  print(title)
```

```
course:)a python
```

برای بی تأثیر کردن کارکتر فرار (backslash)، در ابتدای رشته از کاراکتر r استفاده می کنیم.

```
1  title = r'this\n is\t a python course:)'
2  print(title)
```

```
this\n is\t a python course:)
```

سوال: بی تأثیر کردن کاراکتر فرار چه سودی دارد؟

C:(newline)ew(tab)ext.dat

```
1  with open('C:\new\text.dat', 'w') as myfile:
2      myfile.write("Hello!")
```

صحیح: r'C:\new\text.dat'

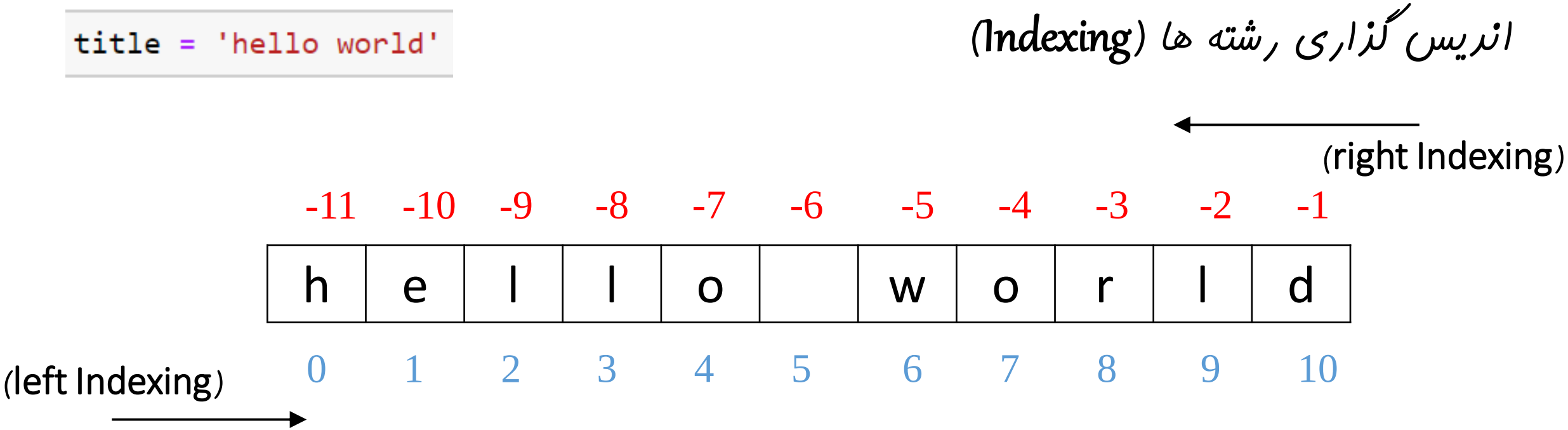
برای تعریف رشته های چندخطی، از مد Triple quotes استفاده می کنیم.

```
1  title = '''hello world
2  this is a python course :)
3  and you should code!!'''
4  print(title)
```

```
hello world
this is a python course :)
and you should code!!
```

```
title = 'hello world'
```

اندیس گذاری رشته ها (Indexing)




```
1 print(title[4])
```

```
o
```

```
1 print(title[-5])
```

```
W
```

برش رشته ها (Slicing): می توان یک زیردنباله از یک رشته را استخراج کرد.

شکل کلی اول:

```
string[start:end]
```

```
1  title = 'Hello World'
2
3  print(title[0:3])        Hel
4  print(title[:3])         Hel
5  print(title[3:])         lo World
6  print(title[3:-1])       lo Worl
7  print(title[:])          Hello World
```

برش رشته ها (Slicing): می توان یک زیردنباله از یک رشته را استخراج کرد.

```
string[start:end:step]
```

شکل کلی دوم:

```
1  title = 'Hello World'
2
3  print(title[0:3:1])      Hel
4  print(title[0:5:2])      Hlo
5  print(title[:8:3])       HlW
6  print(title[3::2])       l ol
7  print(title[9:2:-1])     lroW ol
```

عملیات پایه ای پایتون بر روی رشته ها

عملگر	توضیحات
+	الحاق رشته ها
*	تکرار رشته ها
len	طول رشته ها

```
1  str1 = 'hello'
2  str2 = 'world'
3  str3 = str1+str2
4  str4 = str1*5
5
6  print(str3)
7  print(str4)
8  print(len(str3))
```

```
helloworld
hellohellohellohellohello
10
```

- علاوه بر عملیات پایه ای، تعداد فراوانی عملیات غیراولیه (متد) نیز برای رشته ها تعریف شده است. بحث بیشتر در باره این نوع عملیات به بعد از مباحث برنامه نویسی شیءگرا موکول می گردد.

یادآوری

نکته مهم: رشته ها نیز نوعی دنباله (sequence) هستند و در نتیجه می توان از آنها در for استفاده کرد

```
for var in sequence:
    body
```

```
1 title = 'hello'
2 for i in title:
3     print(i)
```

```
h
e
l
l
o
```

کالکشن ها: لیست

یک *کالکشن مرتب از اشیاء* که دارای خصوصیات زیر است: (یادآوری: رشته یک کالکشن مرتب از کاراکترها است)

۱- می تواند شامل انواع داده غیر همگن باشد.

برخلاف آرایه ها در C++ و رشته ها در پایتون، مقادیری که در لیست نگهداری می شوند، لزوماً همنوع نیستند.

```
l = [1,'a',2,2.75]
```

۲- لیست ها مرتب هستند و ترتیب در آنها توسط اندیس ها مشخص می شود.

اندیس گذاری در لیست ها مشابه اندیس گذاری در رشته ها است. (اندیس گذاری راست، اندیس گذاری چپ، برش (slicing)

-4	-3	-2	-1
1	‘a’	2	2.75
0	1	2	3

۳- طول لیست ها قابل تغییر است.

برخلاف آرایه ها در C++ و رشته ها در پایتون

۴- لیست ها می توانند به شکل تودرتو نیز تعریف شوند

```
l1 = [1,'a',[1,2,’x’,[1,2,3],4],2.75]
l2 = [[1,2,3],[4,5,6],[7,8,9]]
```

۵- لیست ها یک نوع داده تغییرپذیر (mutable) هستند.

برخلاف رشته ها

```
1 title = 'hello'
2 title[1] = 'a'
3 print(title)
```

TypeError: 'str' object does not support item assignment

```
1 L = [1,2,3]
2 L[1] = 'a'
3 print(L)
```

OK

```
[1, 'a', 3]
```

لیست ها نیز نوعی دنباله (sequence) هستند و در نتیجه می توان از آنها در for استفاده کرد ☺

```
1 l = [-1,4,3,2,9]
2 for i in l:
3     print(i**2)
```

```
1
16
9
4
81
```

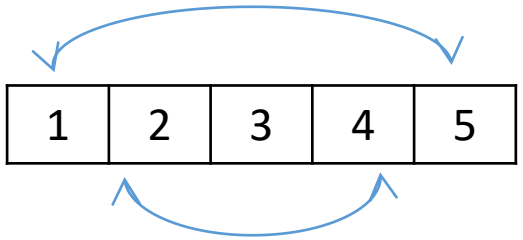
لیست

مثال: برنامه ای بنویسید که یک لیست از مقادیر صحیح را از کاربر گرفته و آن را معکوس کند.

```
1  lst = []
2  n = int(input('Enter the number of elements:'))
3  for i in range(0,n):
4      element = int(input('enter item %d: '%(i)))
5      lst = lst + [element]
6  print(lst)
7
8  for i in range(0,n//2):
9      temp = lst[i]
10     lst[i] = lst[-1-i]
11     lst[-1-i] = temp
12 print(lst)
```

لیست خالی

```
Enter the number of elements:5
enter item 0: 1
enter item 1: 2
enter item 2: 3
enter item 3: 4
enter item 4: 5
[1, 2, 3, 4, 5]
[5, 4, 3, 2, 1]
```



می توان از متد استفاده کرد

```
lst.append(element)
```

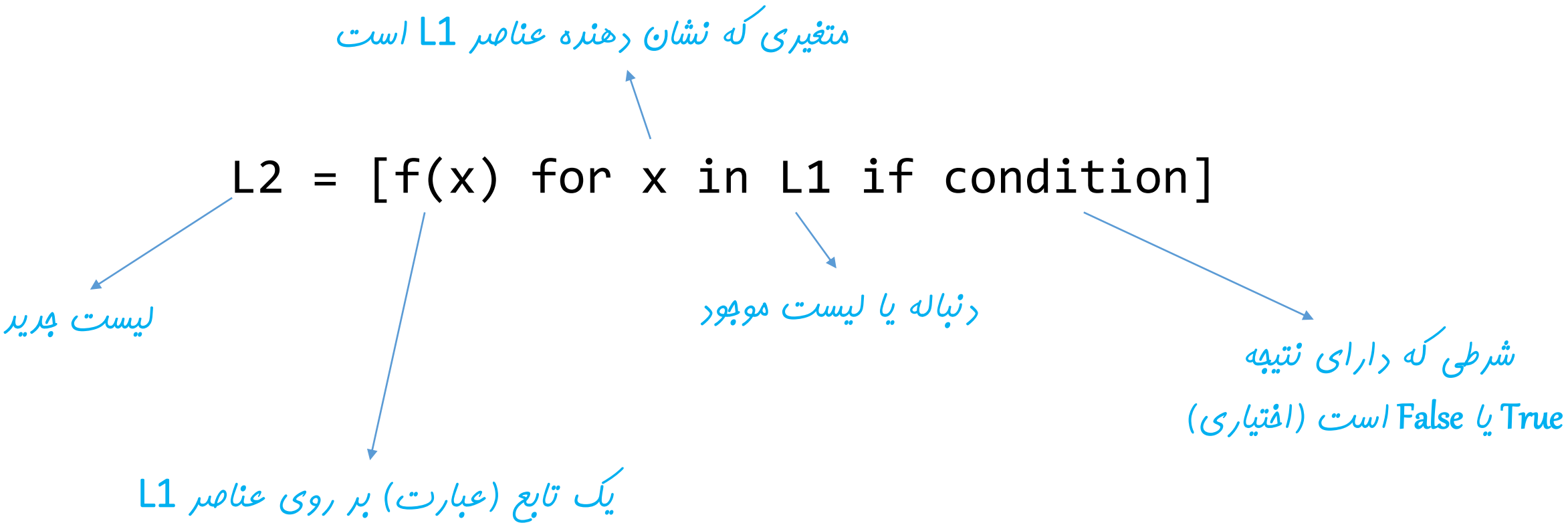
```
Lst = lst + element
```

باعث خطا می شود

مثال: برنامه ای بنویسید که یک عدد صحیح را از کاربر گرفته و لیستی از اعداد اول کوچکتر از آن را ایجاد کند.

لیست: Comprehension

شکل کلی:



لیست: Comprehension

شکل کلی:

```
L2 = [f(x) for x in L1 if condition]
```

تفسیر: از لیست L1، هر عنصری که در شرط condition صدق می کند را انتخاب کرده و تابع f(x) را بر روی آن اعمال کرده و نتیجه را به لیست L2 اضافه کن.

دستور فوق معادل کد زیر است:

```
for x in L1:
    if condition:
        L2 = L2 + [f(x)]
```

لیست: Comprehension

مثال: برنامه ای بنویسید که یک لیست صحیح را از کاربر گرفته و عناصر فرد آن را مشخص کند.

```
1  lst = []
2  n = int(input('Enter the number of elements:'))
3  for i in range(0,n):
4      element = int(input('enter item %d: '%(i)))
5      lst = lst + [element]
6  print(lst)
7
8  new_lst = [i for i in lst if i%2!=0]
9  print(new_lst)
```

```
Enter the number of elements:6
enter item 0: 1
enter item 1: 4
enter item 2: 3
enter item 3: 6
enter item 4: 4
enter item 5: 3
[1, 4, 3, 6, 4, 3]
[1, 3, 3]
```

مثال: برنامه ای بنویسید که لیست توان های سوم اعداد کوچکتر از ۱۰۰ که هم بر ۳ و ۵ یا بر ۱۱ بخش پذیر هستند را تولید کند.

```
1  lst = [x**3 for x in range(0,100) if (x%3==0 and x%5==0) or x%11==0 ]
2  print(lst)
```

```
[0, 1331, 3375, 10648, 27000, 35937, 85184, 91125, 166375, 216000, 287496, 4218
75, 456533, 681472, 729000, 970299]
```

لیست: Comprehension

```
lst = [8,1,10,21,10,13,2,10,8,34,11,23,1,4,2]
center = lst[len(lst)//2]
print(center)

l1 = [i for i in lst if i<center]
l2 = [i for i in lst if i==center]
l3 = [i for i in lst if i>center]

lst = l1+l2+l3
print(lst)
```

```
10
[8, 1, 2, 8, 1, 4, 2, 10, 10, 10, 21, 13, 34, 11, 23]
```

مثال: برنامه ای بنویسید که یک لیست از اعداد صحیح را از کاربر گرفته و عنصر وسط آن را استخراج کند (فرض کنید عنصر وسط **X** باشد)، سپس عناصر لیست را به گونه ای جابجا کند که تمامی عناصری که بعد از **X** قرار می گیرند، از **X** بزرگتر و تمامی عناصری که قبل از **X** قرار می گیرند، از آن کوچکتر باشند.

نکته: عمل فوق پایه و اساس یکی از سریعترین الگوریتم های مرتب سازی، تحت عنوان **quicksort** است. ☺

کالکشن ها: دیکشنری

یک *کالکشن نامرتب از اشیاء* که در آن دسترسی به عناصر از طریق *کلید ها* (نه اندیس ها) انجام می گیرد:

```
{key1 : val1, key2 : val2, … , keyn : valn}
```

کلید ها از نوع string هستند ولی *مقادیر می توانند از هر نوعی باشند.*

```
D ={‘name': 'John', 'major': 'Computer Science', 'age': 25,
'weight': 77.4, 'family': {'father': 'Peter', 'mother': 'Sara'},
'Grades': {'AP': 18.75, 'Math': 12.5}, ‘phone':[0911..8 , 0911..3]}
```

مثالی از تعریف و دسترسی به عناصر دیکشنری

```
D = {'name': 'John', 'major': 'Computer Science',
     'age': 25, 'weight': 77.4,
     'family':{'father':'Peter', 'mother': 'Sara'} ,
     'Grades': {'AP': 18.75, 'Math': 12.5} ,
    'phone':[91198438, 912838484]}


print(D['name'])
print(D['family']['mother'])
print(D['phone'][-1])

```

```
John
Sara
912838484
```

مثالی از عملیات بر روی یک دیکشنری

```
1  D = {'name': 'John', 'major': 'Computer Science',
2       'age': 25, 'weight': 77.4,
3       'family':{'father':'Peter', 'mother': 'Sara'} ,
4       'Grades': {'AP': 18.75, 'Math': 12.5} ,
5      'phone':[91198438, 912838484]}
6
7  D['name'] = 'Bob'
8
9  D['friends'] = ['Ali', 'Ahmad']
10
11 del D['weight']
12
13 print(D)
14 print(len(D))
```

تغییر عناصر دیکشنری

افزودن یک عنصر جدید

حذف یک عنصر از دیکشنری

```
{'name': 'Bob', 'major': 'Computer Science', 'age': 25, 'family': {'father': 'Peter', 'mother': 'Sar
a'}, 'Grades': {'AP': 18.75, 'Math': 12.5}, 'phone': [91198438, 912838484], 'friends': ['Ali', 'Ahma
d']}
7
```

کالکشن ها: خلاصه

نام کلکسیون	نوع داده	تغییرپذیر؟	دسترسی به عناصر
string	متنی	✘	اندیس (راست و چپ)
list	هر نوع داده ای	✓	اندیس (راست و چپ)
Dictionary	هر نوع داده ای	✓	کلید