

Lecture 6

2.3 Other Sequence Classes: `str` and `tuple` classes

2.4 Numeric Types: `int`, `long`, and `float`

4.1 For Loops (review)

Other Sequence Classes:

`str` class

- a built-in class
- immutable objects
(once constructed, can no longer be modified)
- used to represent ordered sequences

Construction of lists:

`str()` - invocation (call) of strings' constructor
(returns empty string "")

or

`phrase = "Hello! Long time no see."`
(using literal form)

See pages 56-57 for the list of behaviors of strings:

- return the information about existing string
- generate a new string as a result
- convert between strings and lists of strings

Examples:

```
string=raw_input("Please, input any sentence:")
```

```
string.count('a') - counts the number of occurrences of  
letter 'a' in the string
```

```
len(string) - finds the length of the string
```

```
print string.upper() - converts all letters to upper case, prints  
print string - the value of "string" is unchanged
```

```
position= string.find(' ') - finds the first occurrence of  
whitespace
```

```
str2=string[:position] - assigns the head of the string  
(before the whitesace) to "str2"
```

```
pos2=string.find(' ',position+1) - finds the second  
occurrence of whitespace
```

```
str3=string[position+1:pos2] - assigns the content of the  
string in between the first and the second whitespaces to "str3"
```

```
print str2 + " ho-ho-ho " + str3 - prints the new strings with  
"ho-ho-ho" in between
```

```
print string * 3 - prints three times concatenated string
```

Examples:

See `string_example.py`

Did you notice that the method `len` is supported in both `list` and `string` classes?

Examples:

See `string_example.py`

Did you notice that the method `len` is supported in both `list` and `string` classes?

- *Polymorphism*

Examples:

More uses:

```
string="Hello, Buy, 123, Thank you"
```

```
print string, type(string)
```

```
l = string.split(',')
```

```
print l, type(l)
```

See `string_example2.py`

Other Sequence Classes:

tuple class

- a built-in class
- immutable objects
(once constructed, can no longer be modified)
- used to encapsulate multiple pieces of information into a single composite object that can be stored or transmitted.

Construction of lists (using literal form):

```
contact_info = (718-987-2345, 917-765-2314, AAA@mail.com)
```

or

```
white = (255, 255, 255)
```

Tuples support all the nonmutating behaviors of lists
(see page 42)

Numeric Types:

int, long, and float

- a built-in class
- immutable objects

See the list of **operators** (along with the syntax) on page 59.

Recall tricky cases:

`result = 5/2` - 'result' will have value 2 assigned to it

Type conversions (casting):

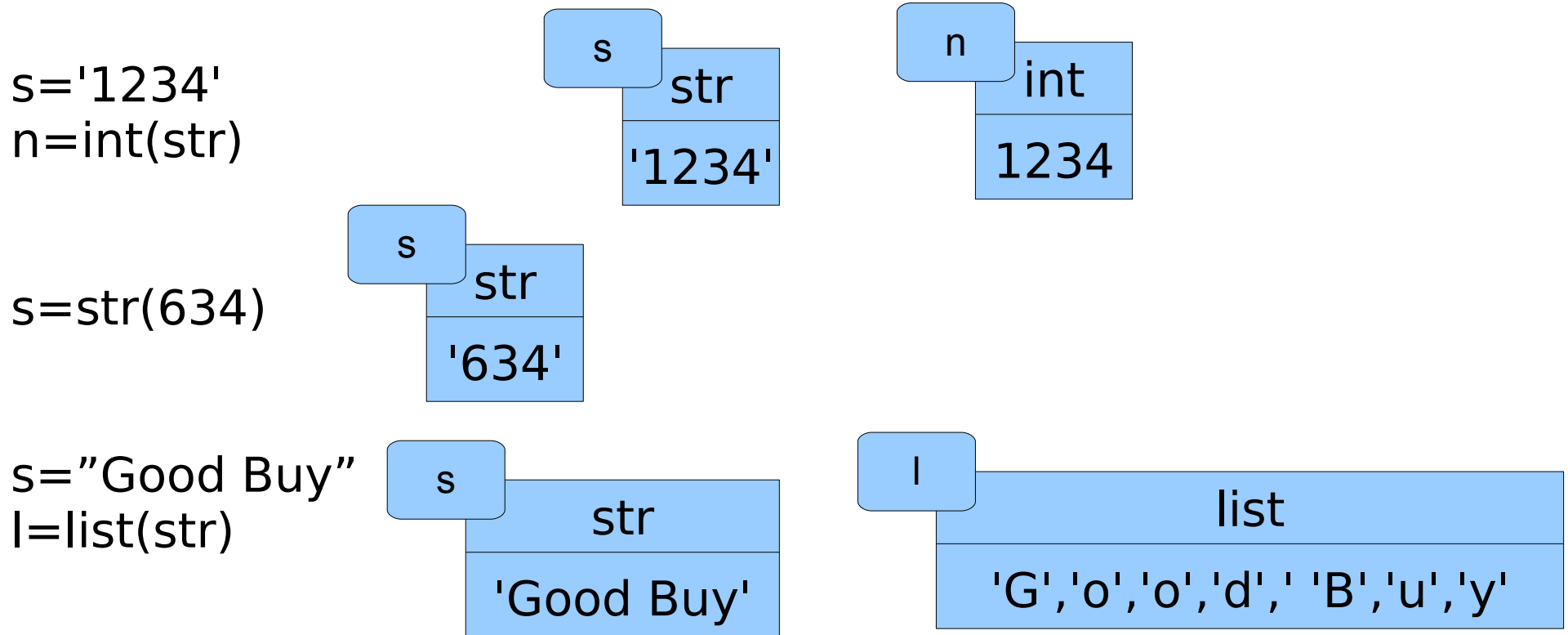
`n=int(2.8)` - 'n' will have value 2 assigned to it (truncation)

`n=round(2.8)` - 'n' will have value 3 assigned to it

`n=round(3.141592654,4)` - 'n' will be 3.1416

Numeric Types: int, long, and float

Type conversions (casting) continues:



Low-level encoding of characters (ASCII, or Unicode):

`ord('a')` `chr(97)`

4.1 For Loops

- use when we need to repeat a series of steps [for each item in a sequence]

Syntax of the *for loop*:

```
for identifier in sequence:  
    body
```

example:

```
for i in range(10):  
    t = i*10  
    print "iteration",i,":",  
    print t
```

4.1 For Loops

One more example (see *for-list.py* for full version):

```
groceries = ["Milk", "Sugar", "Bread", "Honey"]
```

```
enum=1  
for item in groceries:  
    item=str(enum) + '. ' + item  
    enum +=1  
    print item
```

Can you predict what the program will do?

Types of loops:

definite loops, index-based loops (counted loops), nested loops

Homework Assignment

HW4:

1. page 84 / 2.21

2. Write a program that takes a list of strings from the user, and then if an element of the list is a string of digits, the program converts all such elements to whole numbers (positive integers), and replaces the corresponding elements of the list with these numbers.

Example of input-output:

Input: ['Hello', 'buy', 'mind', '5', 'abc', '12', '0', '1243', 'thank you', 'sorry', '432', 'mind', '5', 'Hello', 'George']

Output: ['Hello', 'buy', 'mind', 5, 'abc', 12, 0, 1243, 'thank you', 'sorry', 432, 'mind', 5, 'Hello', 'George']

Don't forget to comment your program

Exercises for self-practice (not for grade): 2.17, 2.18
(answers are given at the end of the book)