Lecture 11

Topics: *Chapter 5. Computing with strings*

5.1 The String Data Type
5.2 Simple String Processing
5.3 Lists as Sequences

5.1 String data type

Text is represented in programs by the *string* data type.

Think of string as a sequence of characters (symbols).

```
>>> string1=''Hello''
>>> string2='Hello'
>>> type(string1)
<class 'str'>
>>> type(string2)
<class 'str'>
```

5.1 String data type

Inputting strings:

use input function: `input`

– doesn't evaluate the expression the user enters.
(use raw_input in Python 2)

Example:

```
def main()
   f_name = input(''Enter your name, please:'')
   print(''Good day, '', f_name)

main()
```

If we run the program and type in name Caroline, then we'll get:

```
Good day, Caroline
```

5.1 String data type

String is a sequence of characters/symbols, thus we can access the individual characters/symbols through *indexing*:

I	T		I	S		S	U	N	N	Y		T	O	D	A	Y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

indexing is used in string expressions to access a specific character position in the string.

Syntax:

```
<string>[<expression>]
```

5.1 String data type

Type the following in the interactive window:

```
>>> phrase = ''It is sunny today''
>>> phrase[0]
'I'
>>> phrase[2]
' '
>>> phrase[10]
'y'
```

5.1 String data type

positive indexing: from the left end to the right end
negative indexing: from the right end to the left end

```
>>> phrase[-1]

>>> phrase[-5]
```

5.1 String data type

It is also possible to access a contiguous sequence of characters (substring):

syntax:

```
<string>[<start>:<end>]
```

```
>>> phrase = ''It is sunny today''
>>> phrase[3:10]
'is sunn'
>>> phrase[:3]
'It '
>>> phrase[11:]
' today'
```

5.1 String data type

Operations with strings:

concatenation:	`+`	example: `string1+string2`
repetition:	`*`	example: `string1*string2`
length of a string:	`len`	example: `len(string1)`

iteration through characters: `for <var> in <string>`

example: for i in 'Hello John'

see example program: string_operations.py

5.2 Simple string processing

Let's write a program that will be printing out the months name given the number of the month.
In other words, the user will enter the number of the month, and our program will output the months name.

Recall the months names and their numbers:

1	January	7	July
2	February	8	August
3	March	9	September
4	April	10	October
5	May	11	November
6	June	12	December

5.2 Simple string processing

Let's write a program that will be printing out the months name given the number of the month.
In other words, the user will enter the number of the month, and our program will output the months name.

Recall the months names and their numbers:

1	January	7	July
2	February	8	August
3	March	9	September
4	April	10	October
5	May	11	November
6	June	12	December

What Python's tools to use? **1.** Use if-elif-else or **2.** Use strings

5.2 Simple string processing

1. Use if-elif-else statements.

Algorithm:

```
input the number if the month (n)
if n=1: output January
elif n=2: output February
elif n=3: output March
...
elif n=12:  output December
else :  wrong number of the month
```

5.2 Simple string processing

2. Use strings

the longest months name is September: 9 characters
Store the names of months in one string, each months name should occupy 9 slots:

```
months='January  February March    April    May
    June     July     August   SeptemberOctober
November December '
```

Algorithm:

```
input the number of a month (n)
output the slice/piece of the months string:
  9 characters long,
  starting from ((n-1)*9)th position
```

see program months_string.py

5.3 Lists as Sequences

List data type in Python represents a sequence of elements, which are values of any Python data type.

We can work with *lists of strings*, *list of integers*, *list of values of mixed data type*, etc.

Syntax: [list of elements separated by commas]

Examples: [1, 2, 3, 4, 5, 6]
[1, ’a’, “Hi, how are you?”, 132, 5.6]

All the string operations listed before are applicable to sequences (lists).

5.3 Lists as Sequences

Type the following in the interactive window:

```
>>> [1,5] + [2,8]
[1, 5, 2, 8]

>>>[1,5]*4
[1, 5, 1, 5, 1, 5, 1, 5]

>>> list_of_grades=['A','B','C','D','F']
>>> list_of_grades[0]
'A'
>>> list_of_grades[3]
'D'
>>> list_of_grades[2:4]
['C', 'D']
```

```
>>> len(list_of_grades)
5
```

5.3 Lists as Sequences

Lists are more general than strings: they can be sequences of arbitrary values, not just characters

```
myList=[1,''January'',2,''February'',3,''Hello'']
```

Using lists we can re-write our program for months and make it easier to retrieve months names by their number:

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

see program months_lists.py

5.3 Lists as Sequences

! Lists are mutable, i.e. the value of an item in a list can be modified with an assignment statement.

Example:

```
myList=[1,''Thank you'',5]
myList[2]=''Hello''
```

the resulting list: [1,"Thank you","Hello"]

5.3 Lists as Sequences

Matrices

[**Def**] A matrix is a rectangular array of numbers.
A matrix with m rows and n columns is called m × n matrix.

plural form: matrices

denotaion: $\mathbf{A}_{m \times n}$

matrix with m = n is called square matrix

$$A = \begin{bmatrix} 1 & 4 & -9 \\ 0 & 1 & 3 \\ 7 & -2 & 8 \\ 0 & -1 & 5 \end{bmatrix}$$

matrix with 4 rows and 3 columns

5.3 Lists as Sequences

Matrices

a_{ij} – element of matrix in row i and column j

$$A = \begin{array}{cccc} 1\text{st} & 2\text{nd} & 3\text{rd} & \\ \left[\begin{matrix} 1 & 4 & -9 \\ 0 & 1 & 3 \\ 7 & -2 & 8 \\ 0 & -1 & 5 \end{matrix}\right. & & \left.\right] & \begin{matrix} 1\text{st} \\ 2\text{nd} \\ 3\text{rd} \\ 4\text{th} \end{matrix} \end{array}$$

5.3 Lists as Sequences

Matrices

a_{ij} – element of matrix in row i and column j

$$\begin{array}{cccc|c} & \text{1st} & \text{2nd} & \text{3rd} & \\ & 1 & 4 & -9 & \text{1st} \\ A = & 0 & 1 & 3 & \text{2nd} \\ & 7 & -2 & 8 & \text{3rd} \\ & 0 & -1 & 5 & \text{4th} \end{array}$$

$a_{12} = 4$

$a_{21} = 0$

$a_{42} = -1$

5.3 Lists as Sequences

Matrices in Python

$$A = \begin{array}{cccc} 0\text{th} & 1\text{st} & 2\text{nd} & \\ \hline 1 & 4 & -9 & 0\text{th} \\ 0 & 1 & 3 & 1\text{st} \\ 7 & -2 & 8 & 2\text{nd} \\ 0 & -1 & 5 & 3\text{rd} \end{array}$$

$a_{12} = 4$ $\rightarrow$ $a_{01} = 4$

$a_{21} = 0$ $\rightarrow$ $a_{10} = 0$

$a_{42} = -1$ $\rightarrow$ $a_{31} = -1$

In Python interactive window:

```
>>> A = [ [1,4,-9], [0,1,3], [7,-2,8], [0,-1,5]]

>>> A[0][1]
4
          >>> A[1][0]
          0
```

5.3 Lists as Sequences

Matrices in Python

$$A = \begin{array}{c|ccc|c} & 0\text{th} & 1\text{st} & 2\text{nd} & \\ & 1 & 4 & -9 & 0\text{th} \\ & 0 & 1 & 3 & 1\text{st} \\ & 7 & -2 & 8 & 2\text{nd} \\ & 0 & -1 & 5 & 3\text{rd} \end{array}$$

$$\begin{array}{l} a_{12} = 4 \\ a_{21} = 0 \\ a_{42} = -1 \end{array} \longrightarrow \begin{array}{l} a_{01} = 4 \\ a_{10} = 0 \\ a_{31} = -1 \end{array}$$

In Python interactive window:

```
>>> A = [1, 4, -9, 0, 1, 3, 7, -2, 8, 0, -1, 5]

>>> A[0*3+1]
4
        >>> A[1*3+0]
        0
```

$\boxed{a_{ij} \rightarrow \text{a[i*columns+j]}}$

5.3 Lists as Sequences

Matrices in Python

Example:
Let’s write a program that asks the user for the matrix size (number of rows and number of columns) and then creates a matrix filled with random numbers from [-5,5]

```
from random import randint

for i in range(rows * columns):
    get a random number
    append it to the list
```

5.3 Lists as Sequences

```
from random import randint
```

random.randrange([start], stop[, step])
returns a randomly selected element from range(start, stop, step).

random.randint(a, b)
returns a random integer N such that a <= N <= b.
Alias for randrange(a, b+1).

see matrix_intro.py