PYTHON: SEQUENCES: STRINGS, AND LISTS

Objectives

- To understand the string data type and how strings are represented in the computer.
- To be familiar with various operations that can be performed on strings through built-in functions and the string library.
- To understand the basic idea of sequences and indexing as they apply to Python strings and lists.
- To understand the concept of objects, an active data type.

The String Data Type

- Text is represented in programs by the *string* data type.
- A string is a **sequence of** characters enclosed within quotation marks (") or apostrophes (').

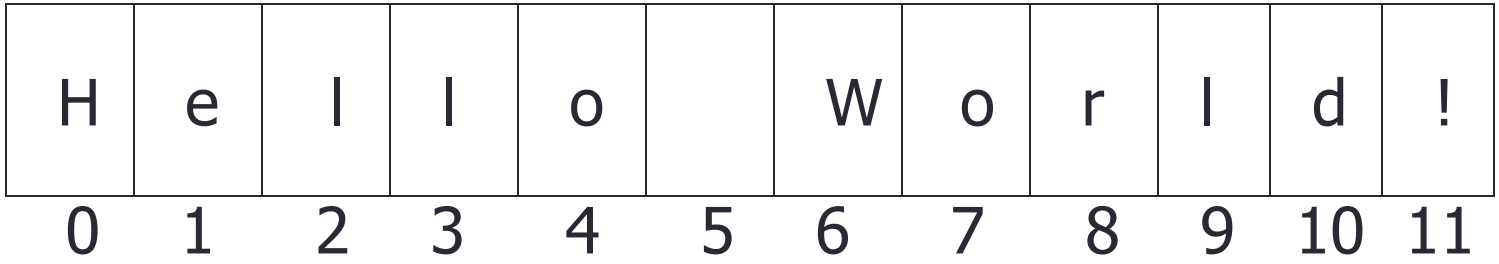
```
>>> str1="Hello"
>>> str2='spam'
>>> print(str1, str2)
Hello spam
>>> type(str1)
<type 'str'>
>>> type(str2)
<type 'str'>
```

EXERCISE 5.1

- Create a string str = "Hello World!".
- Enter str[0], str[5], str[11], str[12], and find out what you are doing.

Indexing a string

- A string is a sequence of characters.
- The first character is indexed by 0, the second by 1, and so on.



- The general form is `<string>[<expr>]`, where the value of `expr` determines which character is selected from the string.

EXERCISE 5.2

- Create a string str = "Hello World!".
- Enter str[-1], str[-7], str[-12], str[-13] and find out what you are doing.

String slicing

- We can also access a contiguous sequence of characters, called a *substring*, through a process called *slicing*.
- Slicing: `<string>[<start>:<end>]`
  - `start` and `end` should both be integers.
  - The slice contains the substring beginning at position `start` and runs up to **but doesn't include** the position `end`.
  - The defaults are the beginning and the end.

  - Inclusive <start>
  - Exclusive <end>
  - [1,10) = {1,2,…9}

EXERCISE 5.3

- Create a string str = "Hello World!".
- Enter str[0:3], str[5:9], str[:5], str[5:], str[:] and find out what you are doing.
- Try negative indices.

EXERCISE 5.4

- Set

```
>> str1 = "sausage"
>> str2 = "egg"
```

- Try

```
>> str1 + "and" + str2
>> 3 * str1
>> (3 * str1) + (2 * str2)
```

  to find out what you are doing.
- Try `len(str1 + "and" + str2)`.

EXERCISE 5.5

Try

```
>>> for ch in "Sausage":
      print(ch, end=" ")
```

What does this for-loop do?

Fancy string operations

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

EXERCISE 5.6

Write a program that will ask for a user's first name and last name in lower cases and will output a username that consists of the first letter in the first name and the first seven letters in the last name.

Sample input and output

```
Please enter your first name: hong-va
Please enter your last name: leong
Your username is: hleong
>>> main()
Please enter your first name: dennis
Please enter your last name: liu
Your username is: dliu
>>> main()
Please enter your first name: great
Please enter your last name: alexandar
Your username is: galexand
>>>
```

Lists

- Recall that a string is a sequence of characters.
- A more general object is a list which is a sequence of arbitrary objects.
- For example,
  - myList = [1, 2, 3, 4]
  - myGrades = ["A+", "A", "B+", "B"]
  - myMenu = ["Sausage", "egg", "bread", "potato"]
  - myMix = [1, "Spam ", 4, "U"]

EXERCISE 5.7

- Enter myList1 = [1, 2, 3, 4] and myList2 = [5, 6, 7, 8].
- Try myList1 + myList2, myList1[0], myList1[2:4], and len(myList1)

The **string** operations apply to **list** as well.

Operator	Meaning
+	Concatenation
*	Repetition
<list>[]	Indexing
<list>[:]	Slicing
len(<list>)	Length
For <var> in <list>	Iteration through list items

EXERCISE 5.8

- Enter myGrades = ["A+", "A", "B+", "B"] and myName = "Dennis Liu"
- Try myGrades[0] = "C" and myName[0] = "C".
  - What do you see? Why?

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Native data types

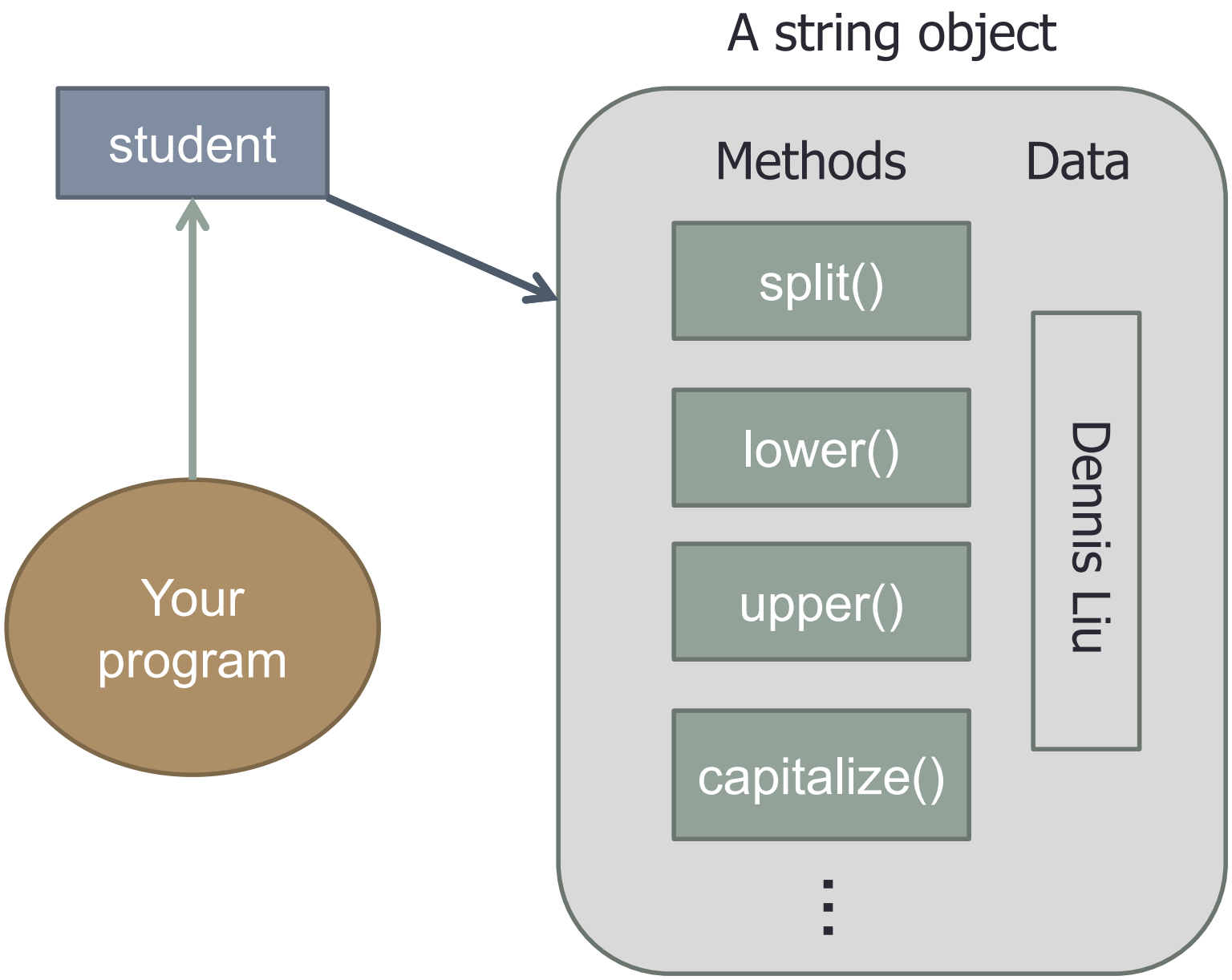
- So far, we have learned some native data types in Python.
- These data types are “passive” in the sense that they are just data and they cannot compute.
- Some problems:
  - We cannot easily use these data types to model real-life objects in our problem solving.
  - E.g., a circle (object) needs three `float` data, and a student record (object) needs numbers and strings.

The concept of objects

- Languages that support object-oriented features provide object data types, such as strings and lists.
- Example:
  - myName = "Dennis Liu"
  - myGrades = ["A+", "A", "B+", "B"]
- The difference with the object data types?
  - Each object contains data (which are generally more complex).
  - Each object also has methods operated in the data.
- In a program, we could request an object to perform operation for us.

EXERCISE 5.9

- Create a string object, such as `student = "first_name last_names student_ID"`.
- Invoke some methods on the object, such as `student.split()`, `student.lower()`, `student.upper()`, `student.capitalize()`.


A string object
student
Methods
Data
split()
lower()
upper()
capitalize()
...
Dennis Liu
Your program

String methods

- For Python version 3 or above:
  - https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str
    - `str.capitalize()`
    - `str.casefold()`
    - `str.center(width[, fillchar])`
    - `str.count(sub[, start[, end]])`
    - `...`
    - `str.title()`
    - `str.translate(map)`
    - `str.upper()`
    - `str.zfill(width)`

Lists are also objects

- One of the methods is `append()`.
- What do these codes give us?

```
squares = []
for x in range(1, 10):
    squares.append(x * x)
print(squares)
```

- Other methods in
  - https://docs.python.org/3/library/stdtypes.html#common-sequence-operations.

Turning a list into a string

- `s.join(list)`: concatenate `list` into a string, using `s` as a **separator**.

```
>> aList = ["Hong-Va", "Leong"]
>> "Dennis Liu".join(aList)
>> " ".join(aList)
```

- `s.join(str)`: concatenate `str` into a string, using `s` as a separator.

```
>> "Dennis Liu".join("Hong-Va Leong")
>> " ".join("Hong-Va Leong")
```

EXERCISE 5.10

Create a list of "A", "B", "C", "D".

How do you use the `join()` method for lists to return "ABCD"?

END