Introduction to Strings
Part II

Course Title: Programming Language II
Course Code: CSE 111
Semester: Summer 2020
Lecture - 8

Last Lecture

- Introduction
- Indexing
- Mutability of String
- Basic String operations
  - Concatenation
  - Deletion
  - Repetition
  - Slicing

Today’s Lecture

- Escape sequence
- Formatting
    - Strings
    - Numbers
- ASCII code
- String functions

BRAC UNIVERSITY
Inspiring Excellence

Escape Sequence

- Strings are represented using single quotes(ex. 'Hello') or double quotes(ex. "Hello")
- What if we a single quote or double quote inside a String?
  For example,

```
s = "I loved the movie "Big Hero 6""
print(s)
```

Output:

SyntaxError: invalid syntax

Did the String end here?

- Interpreter gets confused.
- If you have a substring inside a String that is surrounded by double quotes(""), then use single quotes('') to represent String and vice versa. For example,

```
s = "I loved the movie 'Big Hero 6'" or s = 'I loved the movie "Big Hero 6"'
print(s)
```

Output:

```
I loved the movie 'Big Hero 6' or I loved the movie "Big Hero 6"
```

Escape Sequence

- Better solution to the previously mentioned problem is using Escape sequence**(\ )**.
- If you have some special character inside your String that might confuse the interpreter, you can use escape sequence to clarify. For example,

```
s = "I loved the movie \"Big Hero 6\""
print(s)
```

Output:

```
I loved the movie "Big Hero 6"
```

- Go to this link to see more Escape sequences.
- Another solution to the previously mentioned problem is using triple quotes ("""").

Formatting String

- Use **format()** function to format Strings.
- format() is a powerful and versatile function.
- Basic formatting example:

```
s = "Hello {}, I am {}.".format("Bob", "Alice")
print(s)
s = "Hello {0}, I am {1}.".format("Bob", "Alice")
print(s)
s = "Hello {1}, I am {0}.".format("Bob", "Alice")
print(s)
s = "Hello {speaker}, I am {ThirdPerson}.".format(speaker="Bob", ThirdPerson ="Alice")
print(s)
```

0	1

```
Output:
Hello Bob, I am Alice.
Hello Bob, I am Alice.
Hello Alice, I am Bob.
Hello Bob, I am Alice.
```

Number formatting

- **format( )** function can also used for formatting numbers.
  For example,

```
s = "Hello {0}, can you lend me {1:d}$?".format("Bob", 100)
print(s)
s = "I am {0}, {1}. I have only {2:4.2f}$.".format("sorry", "Alice",50.95876)
print(s)
s = "Its ok, {}.".format("Bob")
print(s)
```

Output:

```
Hello  Bob, can you lend me 100$?
I am sorry, Alice. I have only 50.96$.
Its ok, Bob.
```

- More number formats: b, o, x, e, % etc.

Number formatting with alignment

- **format( )** function can also used for formatting numbers with alignment.
  - < for left alignment
  - ^ for center alignment
  - > for right alignment

For example,

```
print("{:<d}".format(20))
print("{:^14.3f}".format(20.1235))
print("{:>9d}".format(30))
print("{:>09d}".format(30))
print("{0:<d}|{1:^14.3f}|{2:>9d}".format(20,20.1235,30))
```

Output:

```
20
    20.123
       30
000000030
20|    20.123    |       30
```

Follow this link to know more about formatting.

BRAC UNIVERSITY
Inspiring Excellence

ASCII

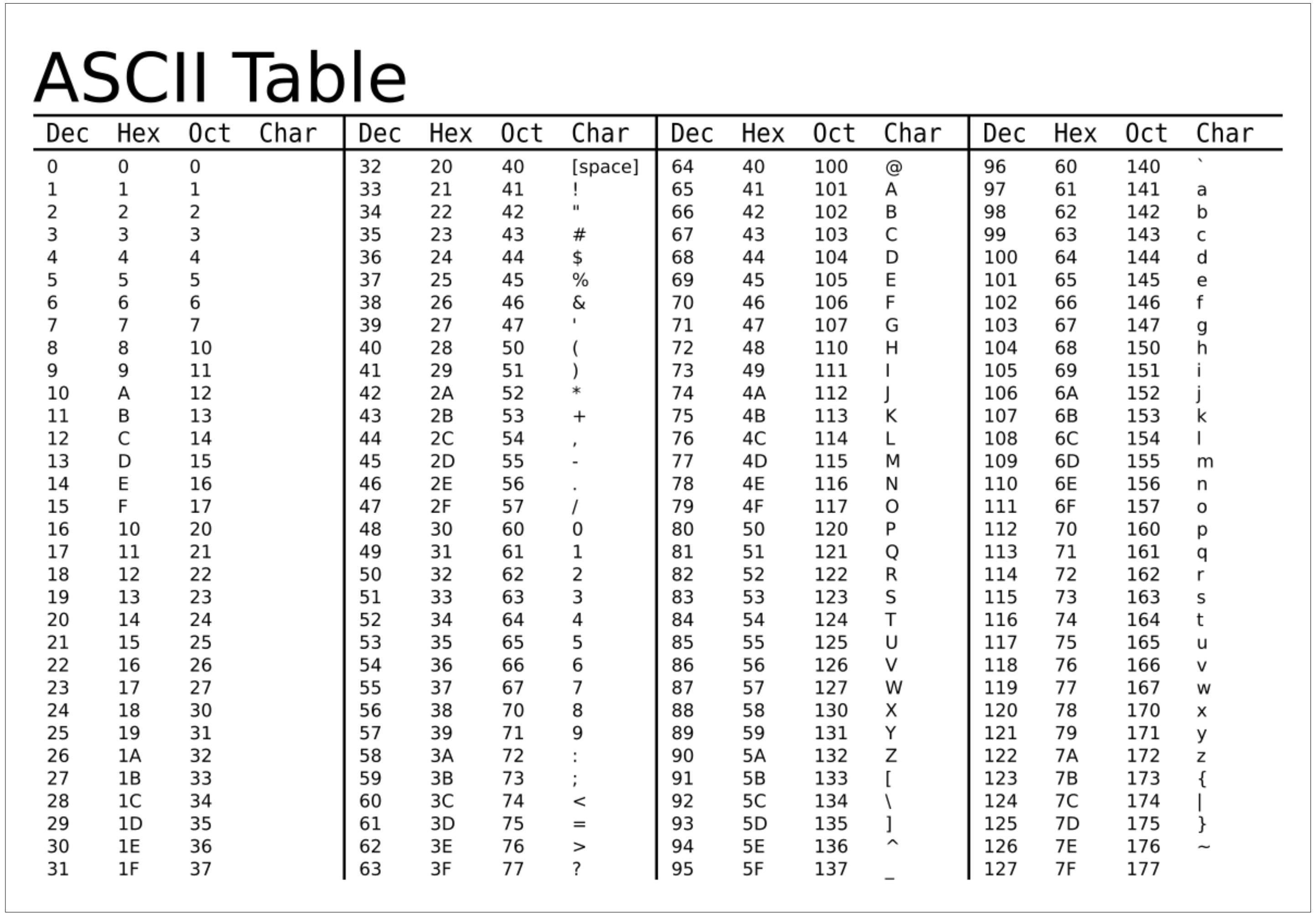
- Machines do not understand characters or decimals.
- Only binary numbers (0s and 1s)
- Every character is converted to an integer number called the **"ASCII code".**
- The ASCII code is converted to binary numbers.
- **ASCII** (American Standard Code for Information Interchange) is a code for representing any character typed using keyboards as numbers, assigned from 0 to 127.
- You can convert a character to its corresponding ASCII value using **ord(character)** function and convert number to its corresponding character using **chr(int)** function.

  For example,

Instructions	Output
`ord('a')`	97
`ord('A')`	65
`ord('Z')`	90
`chr(97)`	'a'
`chr(65)`	'A'

ASCII Table

- You can see the ASCII table to find the ASCII value of the corresponding characters.



ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

BRAC UNIVERSITY
Inspiring Excellence

String Functions

- **lower()** returns a copy of the string with all lower case letters.
- **upper()** returns a copy of the string with all upper case letters.
- **strip()** returns a copy of the string with all whitespace removed before and after the string.
- **count(substring)** returns total occurrence of substring in the main string.
- **startswith(substring)** returns True if the string starts with given substring; Otherwise returns False.
- **endswith(substring)** returns True if the string starts with given substring; Otherwise returns False.
- **find(substring)** returns the index of first occurrence of substring in the main string.
- **replace(oldstring, newstring)** replaces every instance of the oldstring with newstring.

String Functions(Example)

Instructions	Output
s = 'Hello World'	
tmp = s.lower()	
print(s)	Hello World
print(tmp)	hello world
' Hello World '.strip()	'Hello World'
' Hello World '.rstrip()	' Hello World'
' Hello World '.lstrip()	'Hello World '
'Hello World'.count('l')	3
'Hello World'.find('l')	2
'Hello World'.replace('l', 'x')	'Hexxo Worxd'
'Hello World'.startswith('He')	True

BRAC UNIVERSITY
Inspiring Excellence

String Functions

- Use **dir(String)** to see all the methods in String class.
- Run this instruction: **dir('Hello')**

Output:

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower',
'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition',
'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip',
'swapcase', 'title', 'translate', 'upper', 'zfill']
```

- Follow this link to know about all the functions.

BRAC UNIVERSITY
Inspiring Excellence

Summary

- If we have a substring inside a String that is surrounded by double quotes(""), we can use Escape sequence to treat the quotes as part of our String.
- Strings and numbers can be formatted using format() function.
- Every character can be represented using ASCII code.
- String class a lot of methods to make our work easy. We can see them using **dir()** function.

BRAC UNIVERSITY
Inspiring Excellence

Next Lecture

- Introduction to Lists
- List Manipulation
    - Creation
    - Indexing
    - Adding, accessing and removing elements
    - Mutability
    - Slicing
    - Concatenation

BRAC UNIVERSITY
Inspiring Excellence