Programming Tutorial

07 Input and Output

Console

A **command-line interface** (**CLI**) processes commands to a computer program in the form of lines of text. Operating systems implement a command-line interface in a shell (**Console**)for interactive access to operating system functions or services. Users can comunicate with software programs by the console. User can input numbers, words or a sentence into and the program can show the output.

```
# io_input.py

def reverse(text):
    return text[::-1]

def is_palindrome(text):
    return text == reverse(text)

something = input("Enter text: ")

if is_palindrome(something):
    print("Yes, it is a palindrome")
else:
    print("No, it is not a palindrome")
```

```
// 用fmt包从控制台读取输入:
package main
import "fmt"

var (
   firstName, lastName, s string
   i int
   f float32
   input = "56.12 / 5212 / Go"
   format = "%f / %d / %s"
)

func main() {
   fmt.Println("Please enter your full name: ")
   fmt.Scanln(&firstName, &lastName)
   // fmt.Scanf("%s %s", &firstName, &lastName)
   fmt.Printf("Hi %s %s!\n", firstName, lastName) // Hi Chris Naegels
```

```
    fmt.Sscanf(input, format, &f, &i, &s)
    fmt.Println("From the string we read: ", f, i, s)
     // 输出结果: From the string we read: 56.12 5212 Go
}
```

File

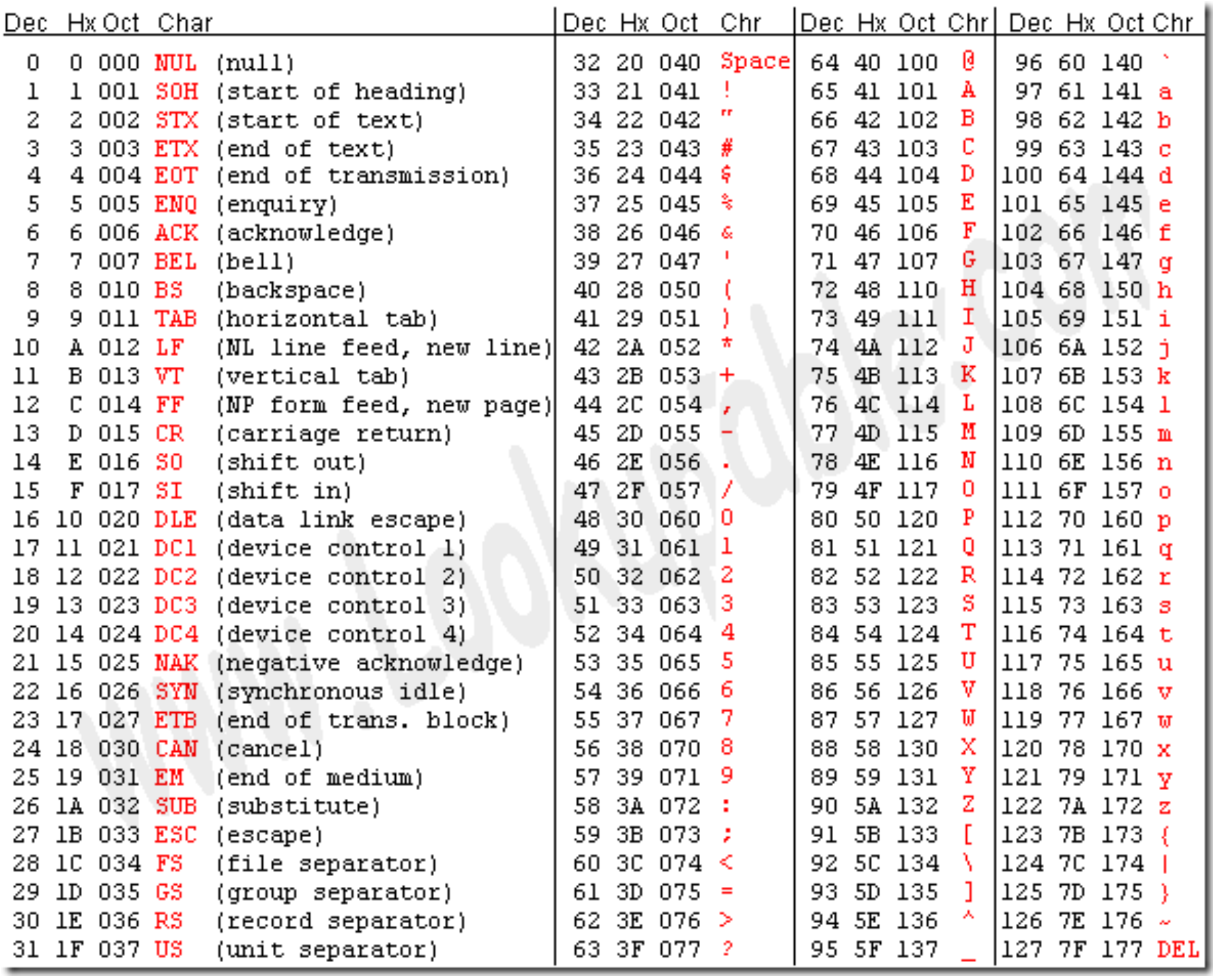
File Format

A **comma-separated values** (**CSV**)) is a delimited text file that uses a comma to separate values. Each line of the file is a data record. Each record consists of one or more fields, separated by commas.

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

Encoding

ASCII (/ˈæskiː/ ASS-kee),abbreviated from American Standard Code for Information Interchange, is a character encoding standard for electronic communication. ASCII codes represent text in computers, telecommunications equipment, and other devices. Most modern character-encoding schemes are based on ASCII, although they support many additional characters.



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

UTF-8 (8-bit Unicode Transformation Format) is a variable width character encoding capable of encoding all valid code points in Unicode using one to four one-byte (8-bit) code units. The encoding is defined by the Unicode Standard, and was originally designed by Ken Thompson and Rob Pike.The name is derived from Unicode (or Universal Coded Character Set) Transformation Format – 8-bit.

It was designed for backward compatibility with ASCII. Code points with lower numerical values, which tend to occur more frequently, are encoded using fewer bytes. The first 128 characters of Unicode, which correspond one-to-one with ASCII, are encoded using a single byte with the same binary value as ASCII, so that valid ASCII text is valid UTF-8-encoded Unicode as well.

Read and Write a File

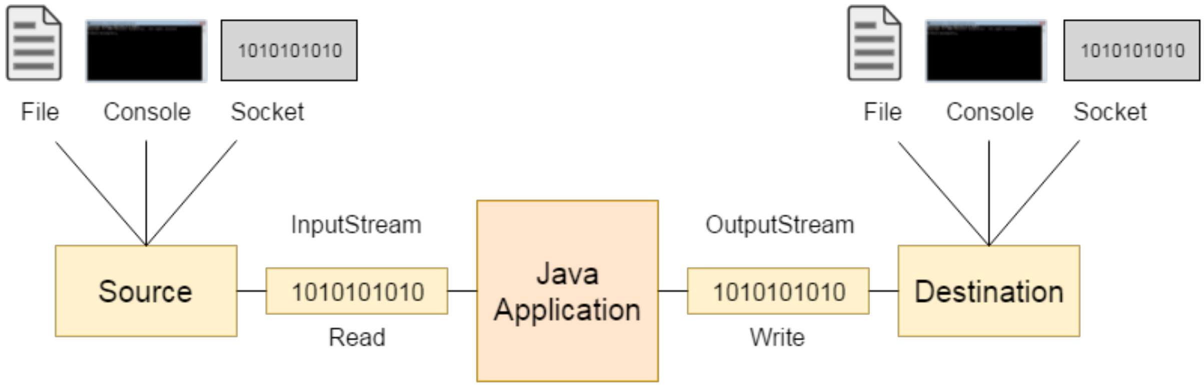
```
# encoding = utf-8
import io

f = io.open("abc.txt", "wt", encoding ='utf-8')
f.write(u"Imagine non-English language here")
f.close()

text = io.open("abc.txt", encoding ="utf-8").read()
print(text)
```

Stream/Reader/Chain

Stream

Many languages perform I/O through Streams. A Stream is linked to a physical layer by I/O system to make input and output operation. In general, a stream means continuous flow of data. Streams are clean way to deal with input/output without having every part of your code understand the physical. These two abstract classes have several concrete classes that handle various devices such as console, disk files, network connection(Socket) etc. Two most important are:

- read() : reads byte of data.
- write() : Writes byte of data.



Reader

Reader is an abstract class for reading character streams. Most subclasses, however, will override some of the methods to provide higher efficiency, additional functionality, or both. The only methods that a subclass must implement are

- read(char[], int, int) : It reads characters into a portion of an array.
- close() : It closes the stream and releases any system resources associated with it.

```
// 用bufio包从控制台读取输入：
package main
import (
    "fmt"
    "bufio"
    "os"
)
var inputReader *bufio.Reader
var input string
var err error

func main() {
    inputReader = bufio.NewReader(os.Stdin)
    fmt.Println("Please enter some input: ")
    input, err = inputReader.ReadString('\n')
    if err == nil {
        fmt.Printf("The input was: %s\n", input)
    }
}
```

Stream Chain

For higher functionality, one stream can be linked or chained to another in Pipe and Filter Style. The output of one stream becomes input to the other. Or to say, we pass an object of one stream as parameter to another stream constructor.

```
import java.io.*;
public class BufferedReaderExample{
public static void main(String args[])throws Exception{
    //Constructor:   InputStreamReader(InputStream in)
    //System.in is the Console Stream.
    InputStreamReader r=new InputStreamReader(System.in);
    BufferedReader br=new BufferedReader(r);
    System.out.println("Enter your name");
    String name=br.readLine();
    System.out.println("Welcome "+name);
}
}
```

Persistance

pickle in Python

Python provides a standard module called pickle which you can use to store *any* plain Python **object** in a file and then get it back later. This is called storing the object **persistently**.

To store an object in a file, we have to first open the file in **write binary** mode and then call the dump function of the pickle module. This process is called *pickling*. Next, we retrieve the object using the load function of the pickle module which returns the object. This process is called *unpickling*.

```
# io_picke.py

import pickle

shoplistfile = 'shoplist.data'
shoplist = ['apple', 'mango', 'carrot']

f = open(shoplistfile, 'wb')

pickle.dump(shoplist, f)
f.close()

del shoplist
```

```

f = open(shoplistfile, 'rb')

storedlist = pickle.load(f)
print(storedlist)
```