

CC1612

Fundamentos de Algoritmos

Prof. Danilo H. Perico

Strings

Strings - Caracteres

- **Strings são cadeias de caracteres: sequência de caracteres**
- A representação de um **caractere** é dada por um **número** inteiro, hexadecimal ou binário
- Esse número segue um **padrão conhecido** entre diversos sistemas computacionais:
 - **ASCII** - *American Standard Code for Information Interchange* (lê-se ASC2)
 - **UTF** - *Unicode Transformation Format*

Tabela ASCII

- **Tabela ASCII:**

- 7 bits (números de 0 a 127)

- **Tabela ASCII Estendida**

- 8 bits

- Igual a tabela ASCII, porém contém mais caracteres:

- Além do 0 ao 127,
- Contém do 128 até o 255 (inclui os caracteres com acentos)

Tabela ASCII - 7 bits

Dec	Hex	Oct	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr	Dec	Hex	Oct	HTML	Chr
0	0	000	NULL	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	Start of Header	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	Start of Text	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	End of Text	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	End of Transmission	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	Enquiry	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	Acknowledgment	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	Bell	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	Backspace	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	Horizontal Tab	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	Line feed	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	Vertical Tab	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	Form feed	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	Carriage return	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	Shift Out	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	Shift In	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	Data Link Escape	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	Device Control 1	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	Device Control 2	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	Device Control 3	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	Device Control 4	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	Negative Ack.	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	Synchronous idle	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	End of Trans. Block	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	Cancel	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	End of Medium	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	Substitute	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	Escape	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	File Separator	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	Group Separator	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	Record Separator	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	Unit Separator	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		Del

UTF - Unicode

- Um dos padrões mais utilizados da atualidade é o **UTF-8** (8-bit *Unicode Transformation **F**ormat*)
- **UTF-8** pode representar qualquer caractere universal padrão do Unicode, sendo também compatível com o **ASCII**

Strings - Unicode

- Tabela Unicode:
 - <https://www.utf8-chartable.de/unicode-utf8-table.pl>
- Tabela ASCII:
 - <https://pt.wikipedia.org/wiki/ASCII>

Strings - Unicode

- *print()* com unicode:

<pre>print(u'\u00ae')</pre>
®
<pre>print(u'\u0061')</pre>
a
<pre>print(u'\u00c7')</pre>
Ç

- No Python, aspas simples ou duplas são intercambiáveis para representar strings

Strings - Unicode

- imprimir o valor **ASCII/Unicode** de um **caractere** - função **ord()**:

```
print( ord("a") )
```

97

```
print( ord("A") )
```

65

```
print( ord("á") )
```

225

Strings - Unicode

- Imprimir o **caractere** a partir de um valor **ASCII/Unicode** - função **chr()**:

```
print( chr(97) )
```

a

```
print( chr(65) )
```

A

```
print( chr(225) )
```

á

Exemplo: entrada em hexadecimal

```
print( chr(0xe1) )
```

á

Strings - aspas triplas

- O Python também tem a opção de aspas triplas: `""" texto """`
- Este comando pode ser chamado de **bloco de string**
- Aspas triplas são bastante úteis para textos com múltiplas linhas
- Exemplo:

```
print("""Este  
texto  
tem  
muitas  
linhas""")
```

```
Este  
texto  
tem  
muitas  
linhas
```

Strings - Índices

- De uma maneira geral, *strings* são **tuplas** de caracteres
- Podemos acessar cada caractere utilizando o índice de sua posição dentro da *String*
- Exemplos:

```
s = "spam"
print(s[0])
```

s

```
s = "spam"
print(s[2])
```

a

```
s = "spam"
print(s[-2])
```

a

Strings - fatiamento (*slicing*)

- Podemos também utilizar o fatiamento (*slicing*) nas strings
- O fatiamento serve para extrairmos uma seção específica da *string*
- Exemplos:

```
s = "spam"
print(s[1:3])
```

pa

```
s = "spam"
print(s[:3])
```

spa

```
s = "spam"
print(s[1:])
```

pam

Strings - Imutabilidade

- As strings seguem o conceito de imutabilidade
- Isso quer dizer que não é possível alterar um único caractere de uma string
- Por exemplo: Alterar o “s” por “z” na palavra “spam”

```
s = "spam"
```

```
s[0] = "z"
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-13-6ae494b2e9c0> in <module>  
      1 s = "spam"  
      2  
>>> 3 s[0] = "z"  
  
TypeError: 'str' object does not support item assignment
```

Strings - Imutabilidade

- Para alterarmos um ou mais caracteres de uma string, normalmente criamos outra variável
- Exemplo: alterar 's' por 'z' na palavra "spam"

```
s = "spam"
s_novo = "z" + s[1:]
print(s_novo)
```

zpam

Strings - Métodos

- Existem muitos métodos próprios para serem utilizados com strings
- Alguns exemplos:
 - `capitalize()`
 - `count()`
 - `find()`
 - `lower()`
 - `islower()`
 - `isdigit()`
 - `isalpha()`
 - `isupper()`
 - `split()`
 - `strip()`
 - `replace()`

Strings - Métodos

```
s = "spammy"  
# conta as ocorrências de "m"  
s.count("m")
```

2

```
# coloca a primeira letra em maiúscula  
s.capitalize()
```

'Spammy'

```
# transforma tudo em maiúsculo  
s.upper()
```

'SPAMMY'

```
# retorna o índice de "a"  
s.find("a")
```

2

```
# verifica se os caracteres são minúsculos  
s.islower()
```

True

```
# substitui "p" por "d"  
s.replace("p", "d")
```

'sdammy'

```
# divide uma frase no argumento utilizado  
s = "teste*de*spam"  
s.split("*")
```

['teste', 'de', 'spam']

```
# remove o espaço vazio no começo e no fim da string  
s = "teste de spam\n"  
s.strip()
```

'teste de spam'

Strings - Métodos

- Assim como nas listas ou tuplas, podemos utilizar também o método *len()* para obter o tamanho da string.
- Tamanho significa o número de caracteres
- Exemplo:

```
s = "olá mundo!"  
print(len(s))
```

10