

CMPSC 100 SPRING 2021

Look it up



“curly” braces



```
a_dictionary = {}
```



dictionaries
have indexes



dictionaries
have keys

Dictionaries:

- Are “associative”
 - Have `key:value` “pairs”
- Use “curly” braces
- Do not use “indexes”
 - All references are to `keys`
- Often mix data types

```
fruit = {  
    "name": "apple",  
    "cost": ".99",  
    "weight": 1, # That's a heavy apple?  
    "color": "red", # Red Delicious r best, no @ me  
    "has_worm": True  
}
```

“subscript” notation



```
print(fruit["name"])
```

```
> apple
```

```
print(fruit["weight"])
```

```
> 1
```

Fruit	Cost	Quantity
Apple	.99	10
Banana	1.10	5
Orange	.75	15

```
fruit_crate = {  
    "apple": {"cost":.99, "quantity": 10},  
    "banana": {"cost":1.10, "quantity": 5},  
    "orange": {"cost": .75, "quantity": 15}  
}
```

Three yellow arrows originate from the dictionary structure and point to the corresponding rows in the table above. The first arrow points from the 'apple' key to the 'Apple' row. The second arrow points from the 'cost' value of the 'apple' entry to the '.99' value in the 'Cost' column. The third arrow points from the 'quantity' value of the 'apple' entry to the '10' value in the 'Quantity' column.


```
fruit_crate = {  
    "apple": {"cost":.99, "quantity": 10},  
    "banana": {"cost":1.10, "quantity": 5},  
    "orange": {"cost": .75, "quantity": 15}  
}
```

```
print(fruit_crate["apple"])
```

```
> {"cost": .99, "quantity": 10}
```


Fruit	Cost	Quantity
Apple	.99	10
Banana	1.10	5
Orange	.75	15

```
print(fruit_crate["apple"]["cost"])
```

```
> .99
```

```
print(fruit_crate["apple"]["quantity"])
```

```
> 10
```



```
for fruit in fruit_crate:  
    data = fruit_crate[fruit]  
    print(data)
```

```
> {"cost": ..., "quantity": ...}
```

```
.
```

```
.
```

```
.
```

And now for something
(not so, but yet kinda)
completely different



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

`strings` are essentially groups
of characters - symbols interpretable
for human convenience.

integer



4 != "4"



string

Bottom line:

Computer don't care

^-_(ツ)_/^-

0	0	000	NULL
1	1	001	Start of Header
2	2	002	Start of Text
3	3	003	End of Text
4	4	004	End of Transmission
5	5	005	Enquiry
6	6	006	Acknowledgment
7	7	007	Bell
8	8	010	Backspace
9	9	011	Horizontal Tab
10	A	012	Line feed
11	B	013	Vertical Tab
12	C	014	Form feed
13	D	015	Carriage return
14	E	016	Shift Out
15	F	017	Shift In
16	10	020	Data Link Escape
17	11	021	Device Control 1
18	12	022	Device Control 2
19	13	023	Device Control 3
20	14	024	Device Control 4
21	15	025	Negative Ack.
22	16	026	Synchronous idle
23	17	027	End of Trans. Block
24	18	030	Cancel
25	19	031	End of Medium
26	1A	032	Substitute
27	1B	033	Escape
28	1C	034	File Separator
29	1D	035	Group Separator
30	1E	036	Record Separator
31	1F	037	Unit Separator

32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
33	21	041	!		65	41	101	A	A	97	61	141	a	a
34	22	042	"		66	42	102	B	B	98	62	142	b	b
35	23	043	#		67	43	103	C	C	99	63	143	c	c
36	24	044	$		68	44	104	D	D	100	64	144	d	d
37	25	045	%		69	45	105	E	E	101	65	145	e	e
38	26	046	&		70	46	106	F	F	102	66	146	f	f
39	27	047	'		71	47	107	G	G	103	67	147	g	g
40	28	048			72	48	108	H	H	104	68	148	h	h
41	29	049			73	49	109	I	I	105	69	149	i	i
42	2A	050	 		74	50	110	J	J	106	70	150	j	j
43	2B	051	!		75	51	111	K	K	107	71	151	k	k
44	2C	052	"		76	52	112	L	L	108	72	152	l	l
45	2D	053	#		77	53	113	M	M	109	73	153	m	m
46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
63	3F	077	?	?	95	5F	137	_	_	127	7F	177		Del

In this course, we really only care about “Horizontal Tab” and “Line Feed” (at least, in principle). Each of these are “nonprinting” characters:

- “\t” Horizontal Tab (“tab”)
- “\n” Line Feed (“new line”)

Because they're groups
of individual characters,
`strings` function as a kind
of data structure (particularly
`lists`)

```
name = "G. Wiz"
```

```
for letter in name:  
    print(letter)
```

```
> G.
```

```
W
```

```
i
```

```
z
```

Also like lists:

- Have known length
- Have methods
- Can be sliced

Methods are the “powers”
of a given object.

Because objects do different
things, methods are often
completely different.

Methods always follow
dot notation

variable

method name

`object.method()`

dot operator

arguments (always - even if none)

Various useful methods of `string`:

- `replace`
- `count`
- `upper`
- `lower`
- `startswith`
- `endswith`

I don't know them all, and it's actually somewhat pointless to Memorize all of them.

If we want to know what methods something has, we generally just look it up.




The interwebs


We can also format strings
using something called an
`f-string`

We can create “templates”
that we can use to print
Formatted statements more
easily.

Place the variable to print
in curly braces.



```
name = input("What's your name? ")  
print(f"Oh hello, {name}")
```



Prefix the string with an f

And, we can also do more than this with strings: we can load them from files using the open function:

```
open("file.txt", "r")
```



File path




File mode


We pull file contents in
by using the

`read` or
`readlines`

methods.



Reads all file contents
as a string



Reads all file contents
as a list, separating
all lines by `\n`