Strings

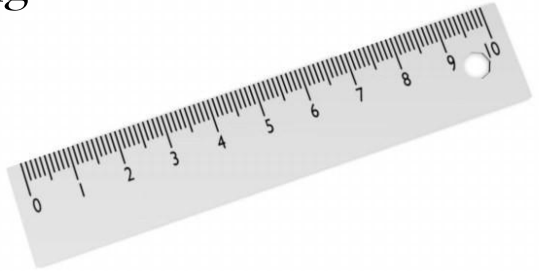
Chapter 8

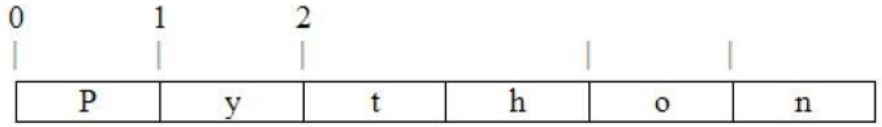
String

- Primitive types (simple values): integers, floats, and booleans
- String is a **sequence**, an ordered collection of other values

- Bracket operator `[]` to access the characters of a string

```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> i = 1
>>> fruit[i]
>>> fruit[i+1]
>>> letter = fruit[1.5]
```





`len` function

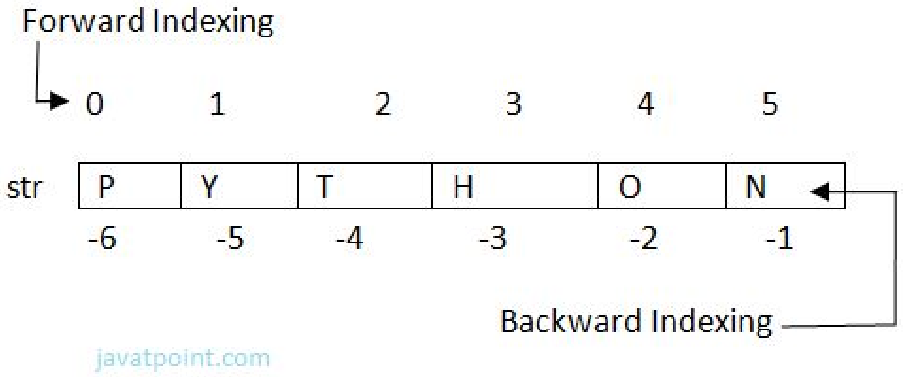
- Returns the number of characters in a string

```
>>> fruit = 'banana'
>>> len(fruit)
6
```

- To get the last letter of a string:

```
>>> length = len(fruit)
>>> last = fruit[length]

>>> fruit[-1]
>>> fruit[-2]
```

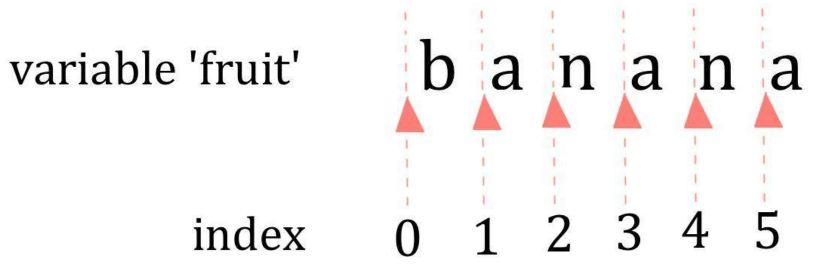


Traversal with a `for` loop

- Process each character of a string one at a time, from beginning till end.
- Using `while` loop:

```
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(letter)
    index = index + 1
```

Figure 8.1 This behaviour is counterintuitive, but it might help to imagine the indices pointing between the characters.



Traversal with a `for` loop (cont.)

- Using `for` loop:

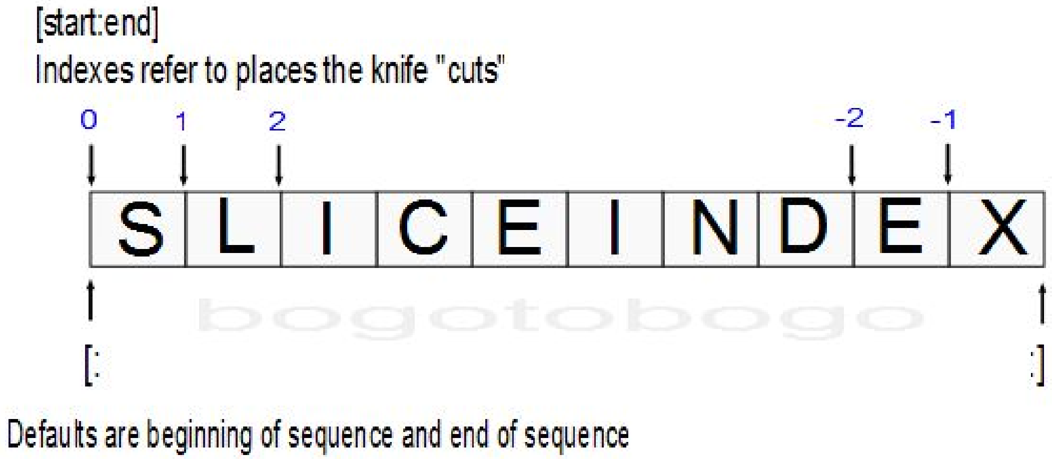
```
for letter in fruit:
    print(letter)
```

- `letter` is a variable that gets the value of the next character in the string each time through the loop
- The loop continues until no characters are left

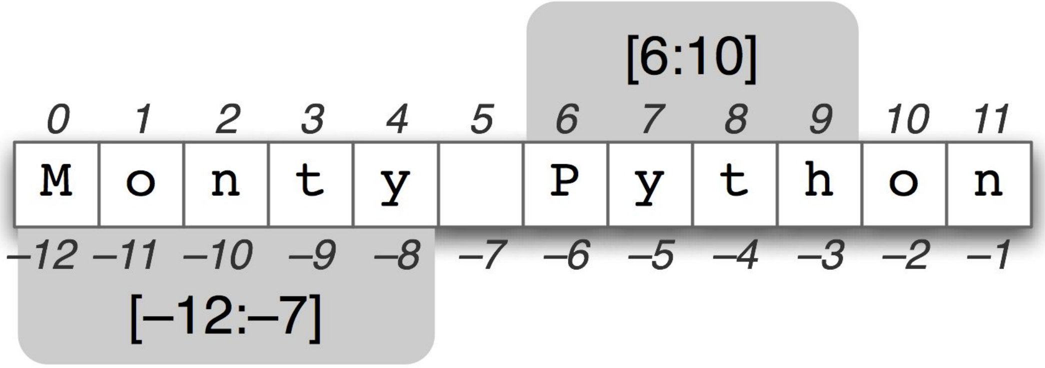
- `for` vs. `while` ?

String Slices

```
>>> s = ' Monty Python'
>>> s[0:5]
' Monty'
>>> s[6:12]
' Python'
```



[start:end]
Indexes refer to places the knife "cuts"
0
1
2
-2
-1
S L I C E I N D E X
[:
:]
bogotobogo
Defaults are beginning of sequence and end of sequence



Empty String

```
>>> fruit = ' banana'
>>> fruit[3:3]
''
```

- A string with no characters, length 0, but it is still a string.

Strings are Immutable

```
>>> fruit = 'banana'
>>> fruit[0] = ‘B’
TypeError: ' str' object does not support item assignment
```

- `object` is the string, `item` is the character to be assigned
- Strings are immutable, i.e. you can NOT change an existing string.

```
>>> new_fruit = ‘B’ + fruit[1:]
>>> new_fruit
Banana
```

Searching

```
def find(word, letter):
    index = 0
    while index < len(word):
        if word[index] == letter:
            return index
        index = index + 1
    return -1
```

Looping and Counting

```
word = ' banana'
count = 0
for letter in word:
    if letter == ' a' :
        count = count + 1
print(count)
```

String Methods

```
>>> word = 'banana'
>>> new_word = word.upper()
>>> new_word
'BANANA'
```

- We are *invoking* `upper` on `word`.

```
>>> index = word.find('a')
>>> word.find('na')
>>> word.find('na', 3)
>>> word.find('na', 3, 5)
```

- See manual

The `in` operator

```
>>> 'a' in 'banana'
True
>>> 'seed' in 'banana'
False

def in_both(word1, word2):
    for letter in word1:
        if letter in word2:
            print(letter)
```

String Comparison

```
if word < ' banana' :
    print('Your word, ' + word + ' , comes before banana. ' )
elif word > ' banana' :
    print(' Your word, ' + word + ' , comes after banana. ' )
else:
    print(' All right, bananas. ' )
```

- What about uppercase?

```
Your word, Pineapple, comes before banana.
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	**@**	96	60	`
1	1	*[START OF HEADING]*	33	21	**!**	65	41	**A**	97	61	**a**
2	2	*[START OF TEXT]*	34	22	**"**	66	42	**B**	98	62	**b**
3	3	*[END OF TEXT]*	35	23	**#**	67	43	**C**	99	63	**c**
4	4	*[END OF TRANSMISSION]*	36	24	**$**	68	44	**D**	100	64	**d**
5	5	*[ENQUIRY]*	37	25	**%**	69	45	**E**	101	65	**e**
6	6	*[ACKNOWLEDGE]*	38	26	**&**	70	46	**F**	102	66	**f**
7	7	*[BELL]*	39	27	**'**	71	47	**G**	103	67	**g**
8	8	*[BACKSPACE]*	40	28	**(**	72	48	**H**	104	68	**h**
9	9	*[HORIZONTAL TAB]*	41	29	**)**	73	49	**I**	105	69	**i**
10	A	*[LINE FEED]*	42	2A	*	74	4A	**J**	106	6A	**j**
11	B	*[VERTICAL TAB]*	43	2B	**+**	75	4B	**K**	107	6B	**k**
12	C	*[FORM FEED]*	44	2C	**,**	76	4C	**L**	108	6C	**l**
13	D	*[CARRIAGE RETURN]*	45	2D	**-**	77	4D	**M**	109	6D	**m**
14	E	*[SHIFT OUT]*	46	2E	**.**	78	4E	**N**	110	6E	**n**
15	F	*[SHIFT IN]*	47	2F	**/**	79	4F	**O**	111	6F	**o**
16	10	*[DATA LINK ESCAPE]*	48	30	**0**	80	50	**P**	112	70	**p**
17	11	*[DEVICE CONTROL 1]*	49	31	**1**	81	51	**Q**	113	71	**q**
18	12	*[DEVICE CONTROL 2]*	50	32	**2**	82	52	**R**	114	72	**r**
19	13	*[DEVICE CONTROL 3]*	51	33	**3**	83	53	**S**	115	73	**s**
20	14	*[DEVICE CONTROL 4]*	52	34	**4**	84	54	**T**	116	74	**t**
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	**5**	85	55	**U**	117	75	**u**
22	16	*[SYNCHRONOUS IDLE]*	54	36	**6**	86	56	**V**	118	76	**v**
23	17	*[ENG OF TRANS. BLOCK]*	55	37	**7**	87	57	**W**	119	77	**w**
24	18	*[CANCEL]*	56	38	**8**	88	58	**X**	120	78	**x**
25	19	*[END OF MEDIUM]*	57	39	**9**	89	59	**Y**	121	79	**y**
26	1A	*[SUBSTITUTE]*	58	3A	**:**	90	5A	**Z**	122	7A	**z**
27	1B	*[ESCAPE]*	59	3B	**;**	91	5B	**[**	123	7B	**{**
28	1C	*[FILE SEPARATOR]*	60	3C	**<**	92	5C	\	124	7C	**|**
29	1D	*[GROUP SEPARATOR]*	61	3D	**=**	93	5D	**]**	125	7D	**}**
30	1E	*[RECORD SEPARATOR]*	62	3E	**>**	94	5E	**^**	126	7E	**~**
31	1F	*[UNIT SEPARATOR]*	63	3F	**?**	95	5F	_	127	7F	*[DEL]*