Strings I

Strings are built from characters

The string "Computer" is represented internally like this:

"C"	**"o"**	**"m"**	**"p"**	**"u"**	**"t"**	**"e"**	**"r"**

- Each piece of a string is called a ***character***.
- A character is a special kind of string that is made up of exactly one letter, number, or symbol.

Accessing characters

Each character in a string is numbered by its position:

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

The numbers above the characters are called ***indices*** (singular: ***index***) or ***positions***.

Accessing characters

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

- There is a separate variable for each character in the string, which is the string variable followed by [ ] with an integer in the middle.

```
my_string = "Computer"
print(my_string[0])    # prints C
print(my_string[7])    # prints r
```

Accessing characters

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

- These individual variables can be used just like regular variables, ***except*** **you cannot assign to them**.

```
my_string = "Computer"
my_string[0] = "B"  # illegal!
```

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

- You can print them, assign them to variables, pass them to functions, etc.

```
my_string = "Computer"
first = my_string[0]
third = my_string[2]
print(first, third, my_string[4])
```

0	1	2	3	4	5	6	7
"C"	"o"	"m"	"p"	"u"	"t"	"e"	"r"

```
def which_first(letter1, letter2):
  if letter1 < letter2:
    return letter1
  else:
    return letter2

def main():
  s = "Computer"
  earlier = which_first(s[6], s[3])
  print(earlier, "comes earlier in the alphabet.")
```

Another Example

```
name = input("What is your name? ")
initial = name[0]
print("The first initial of your name
is", initial)
```

Sample output

```
What is your name? Phil
The first initial of your name is P
```

Getting the length of a string

- Assume `s` is a string variable
- `len(s)` returns the length of s
- `len("Computer") returns 8`
- `len("A B C") return 5`
- `len("") returns 0`
- `len` is uses **`return`**, meaning if you want to capture the length, you should save the return value in a variable.
  - length_of_string = len(string_variable)

Loops over strings

- Accessing characters via numbers naturally leads to using a for loop to process strings.
- What is the first numerical position in any string?
- What is the last numerical position in any string?

Loops over strings

- Accessing characters via numbers naturally leads to using a for loop to process strings.
- What is the first numerical position in any string? 0
- What is the last numerical position in any string? len(s)-1

```
# assume s is a string variable
for pos in range(0, len(s)):
  # do something with s[pos]
```

Loops over strings

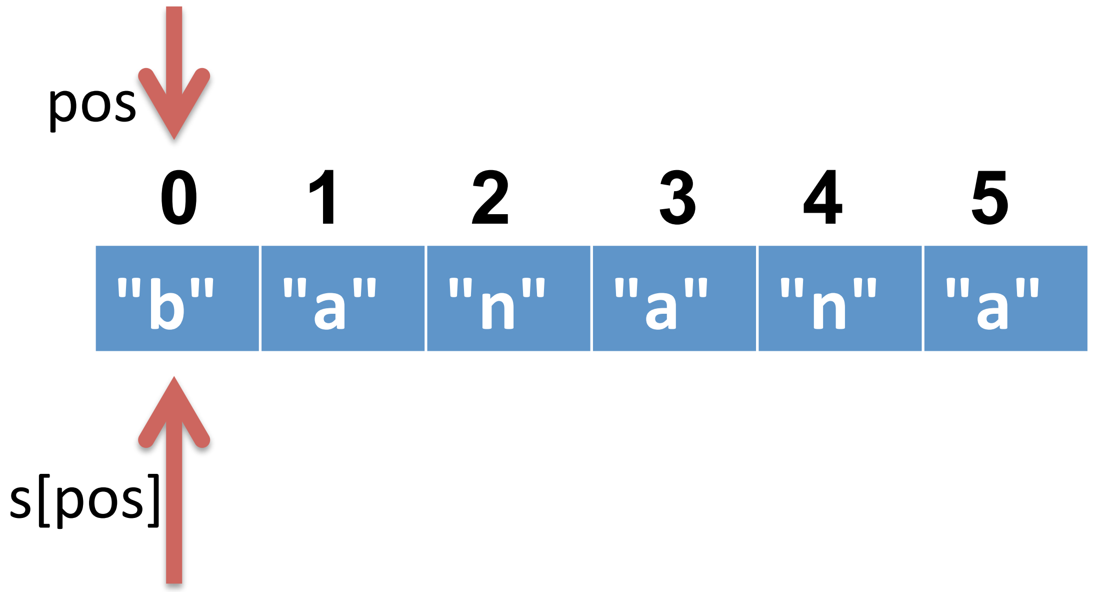
- Accessing characters via numbers naturally leads to using a for loop to process strings.

```
# assume s is a string variable
for pos in range(0, len(s)):
  print(s[pos])
```

```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```

0	1	2	3	4	5
"b"	"a"	"n"	"a"	"n"	"a"

```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```

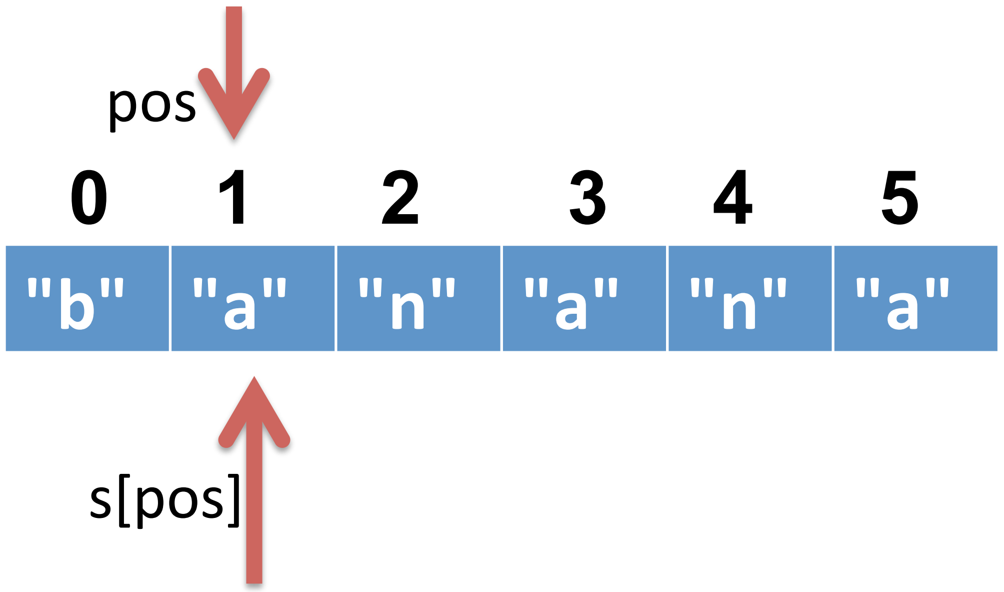


1st iteration
pos: 0
s[pos]: "b"

OUTPUT

```
b
```

```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```



2nd iteration

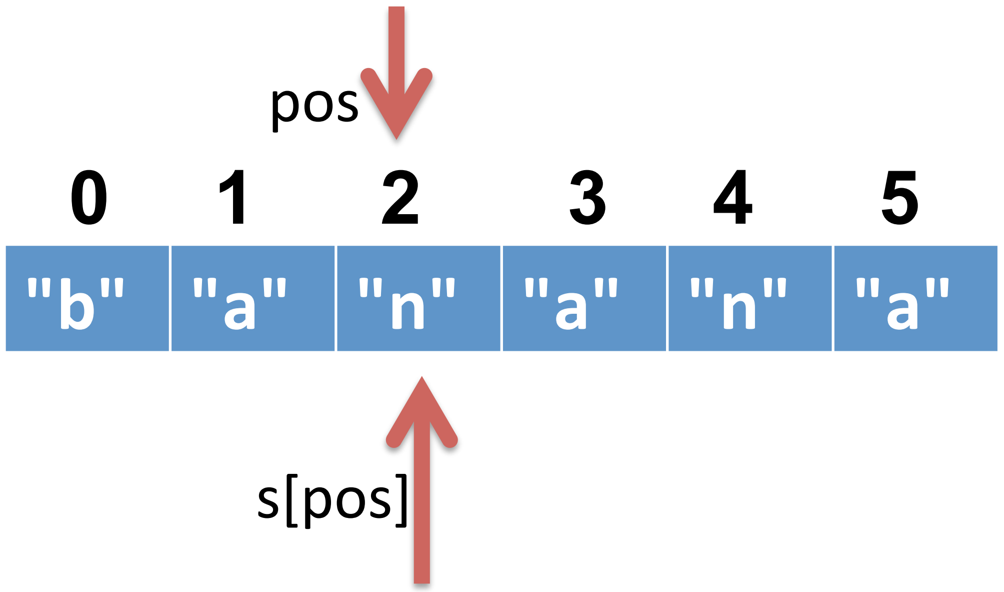
pos: 1

s[pos]: "a"

OUTPUT

```
b
a
```

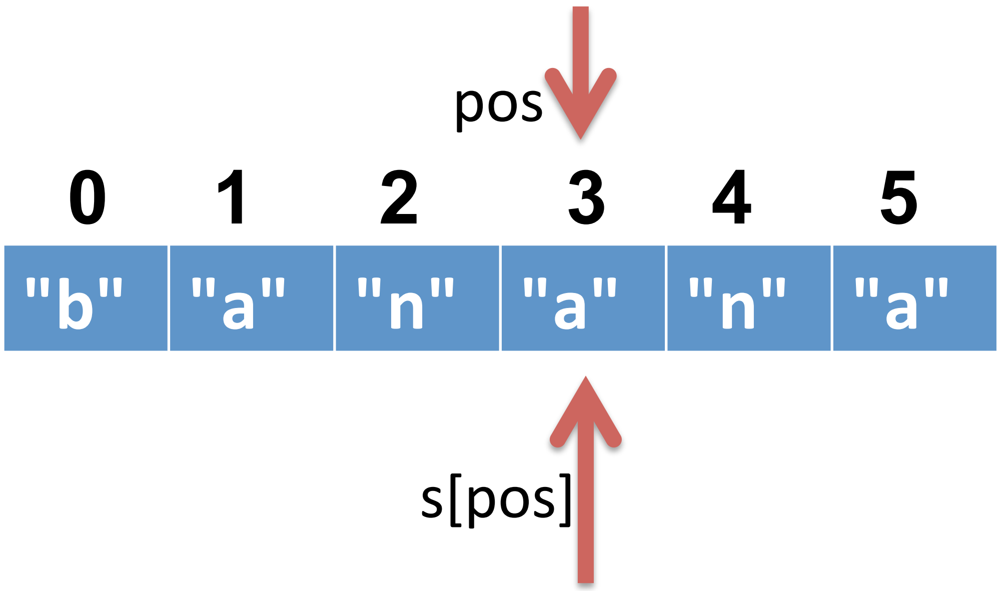
```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```



3rd iteration
pos: 2
s[pos]: "n"

OUTPUT
b
a
n

```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```

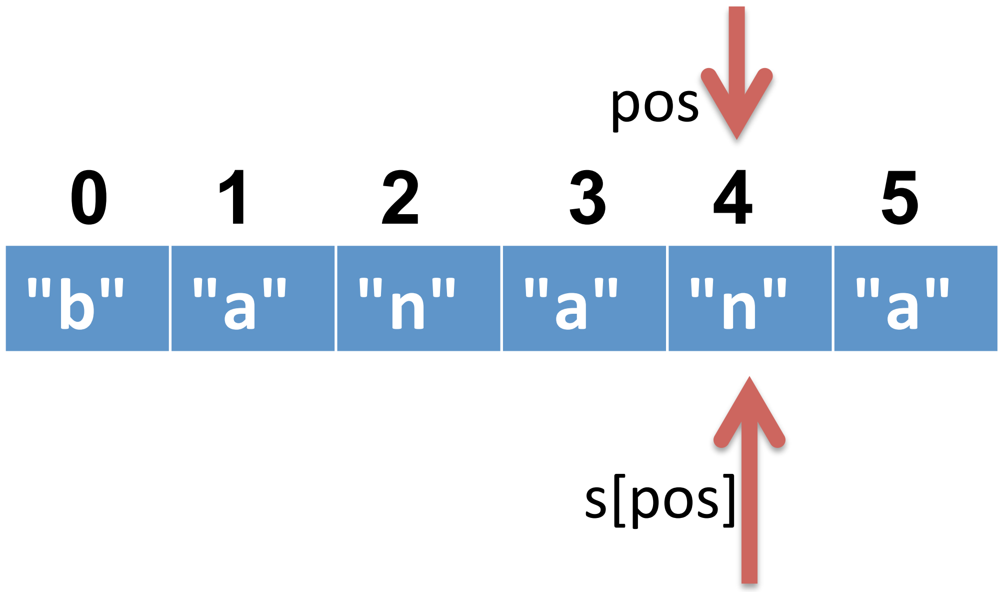


4th iteration
pos: 3
s[pos]: "a"

OUTPUT

```
b
a
n
a
```

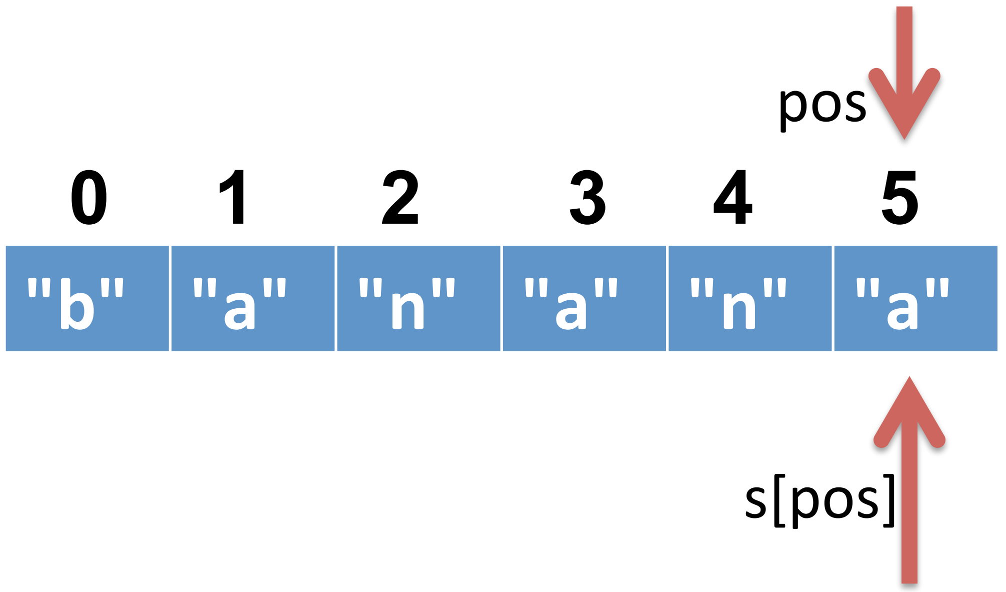
```
s = "banana"
for pos in range(0, len(s)):
  print(s[pos])
```



5th iteration
pos: 4
s[pos]: "n"

```
OUTPUT
b
a
n
a
n
```

```
s = "banana"
for pos in range(0, len(s)):
    print(s[pos])
```



6th iteration
pos: 5
s[pos]: "a"

```
OUTPUT
b
a
n
a
n
a
```

- Write a loop to print the letters in a string in reverse order.
- Write a loop to print every other character in a string, starting with the first.
- Write a loop to count the number of capital letter A's in a string.
- Write a loop to count capital or lowercase A's.
- **Challenge**: Write a loop to print the letters of a string in forward order intermixed with backward order (alternating between forward/backward).

  e.g., for "abcdef" you would print afbecd