41 Questions to Test your Knowledge of Python Strings

How to crush algorithm questions by mastering string fundamentals

Chris Follow
Apr 29 · 8 min read ★

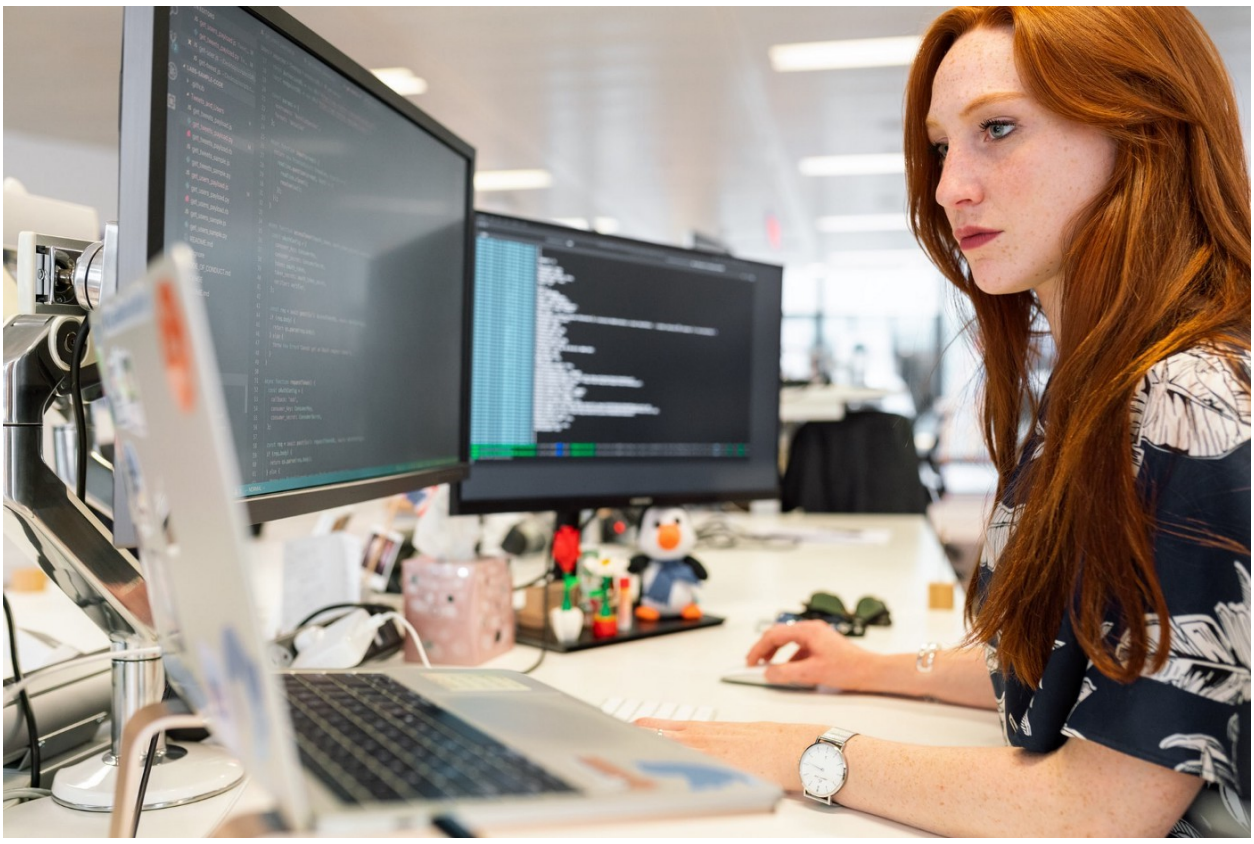

Photo by ThisisEngineering RAEng on Unsplash

I've started tracking the most commonly used functions while doing algorithm questions on LeetCode and HackerRank.

Being a good engineer isn't about memorizing a language's functions, but that doesn't mean it's not helpful. Particularly in interviews.

This is my string cheatsheet converted into a list of questions to quiz myself. While these are not interview questions, mastering these will help you solve live coding questions with greater ease.

How well do you know Python strings?

. . .

1. How would you confirm that 2 strings have the same identity?

The `is` operator returns `True` if 2 names point to the same location in memory. This is what we're referring to when we talk about identity.

Don't confuse `is` with `==`, the latter which only tests equality.

```
animals           = ['python','gopher']
more_animals      = animals

print(animals == more_animals) #=> True
print(animals is more_animals) #=> True

even_more_animals = ['python','gopher']

print(animals == even_more_animals) #=> True
print(animals is even_more_animals) #=> False
```

Notice above how `animals` and `even_more_animals` have a different identity even though they are equal.

Additionally, the `id()` function returns the `id` of a memory address associated with a name. Two objects with the same identity will return the same `id`.

```
name = 'object'
id(name)
#=> 4408718312
```

2. How would you check if each word in a string begins with a capital letter?

The `istitle()` function checks if each word is capitalized.

```
print( 'The Hilton'.istitle() ) #=> True
print( 'The dog'.istitle() ) #=> False
print( 'sticky rice'.istitle() ) #=> False
```

3. Check if a string contains a specific substring

The `in` operator will return `True` if a string contains a substring.

```
print( 'plane' in 'The worlds fastest plane' ) #=> True
print( 'car' in 'The worlds fastest plane' ) #=> False
```

4. Find the index of the first occurrence of a substring in a string

There are 2 different functions that will return the starting index, `find()` and `index()`. They have slightly different behaviour.

`find()` returns `-1` if the substring is not found.

```
'The worlds fastest plane'.find('plane') #=> 19
'The worlds fastest plane'.find('car') #=> -1
```

`index()` will throw a `ValueError`.

```
'The worlds fastest plane'.index('plane') #=> 19
'The worlds fastest plane'.index('car') #=> ValueError:
substring not found
```

5. Count the total number of characters in a string

`len()` will return the length of a string.

```
len('The first president of the organization..') #=> 19
```

6. Count the number of a specific character in a string

`count()` will return the number of occurrences of a specific character.

```
'The first president of the organization..'.count('o') #=> 3
```

7. Capitalize the first character of a string

Use the `capitalize()` function to do this.

```
'florida dolphins'.capitalize() #=> 'Florida dolphins'
```

8. What is an f-string and how do you use it?

New in python 3.6, f-strings make string interpolation really easy. Using f-strings is similar to using `format()`.

F-strings are denoted by an `f` before the opening quote.

```
name = 'Chris'
food = 'creme brulee'
```

```
f'Hello. My name is {name} and I like {food}.'
#=> 'Hello. My name is Chris and I like creme brulee'
```

9. Search a specific part of a string for a substring

`index()` can also be provided with optional start and end indices for searching within a larger string.

```
'the happiest person in the whole wide
world.'.index('the',10,44)
#=> 23
```

Notice how the above returned `23` rather than `0`.

```
'the happiest person in the whole wide world.'.index('the')
#=> 0
```

10. Interpolate a variable into a string using format()

`format()` is similar to using an f-string. Though in my opinion, it's less user friendly because variables are all passed in at the end of the string.

```
difficulty = 'easy'
thing = 'exam'

'That {} was {}!'.format(thing, difficulty)
#=> 'That exam was easy!'
```

11. Check if a string contains only numbers

`isnumeric()` returns `True` if all characters are numeric.

```
'80000'.isnumeric() #=> True
```

Note that punctuation is not numeric.

```
'1.0'.isnumeric() #=> False
```

12. Split a string on a specific character

The `split()` function will split a string on a given character or characters.

```
'This is great'.split(' ')
#=> ['This', 'is', 'great']

'not--so--great'.split('--')
#=> ['not', 'so', 'great']
```

13. Check if a string is composed of all lower case characters

`islower()` returns `True` only if all characters in a string are lowercase.

```
'all lower case'.islower() #=> True
'not aLL lowercase'.islower() # False
```

14. Check if the first character in a string is lowercase

This can be done by calling the previously mentioned function on the first index of the string.

```
'aPPLE'[0].islower() #=> True
```

15. Can an integer be added to a string in Python?

In some languages this can be done but python will throw a `TypeError`.

```
'Ten' + 10 #=> TypeError
```

16. Reverse the string "hello world"

We can split the string into a list of characters, reverse the list, then rejoin into a single string.

```
''.join(reversed("hello world"))
#=> 'dlrow olleh'
```

17. Join a list of strings into a single string, delimited by hyphens

Python's `join()` function can join characters in a list with a given character inserted between every element.

```
'-'.join(['a','b','c'])
#=> 'a-b-c'
```

18. Check if all characters in a string conform to ASCII

The `isascii()` function returns `True` if all characters in a string are included in ASCII.

```
print( 'Â'.isascii() ) #=> False
print( 'A'.isascii() ) #=> True
```

19. Uppercase or lowercase an entire string

`upper()` and `lower()` return strings in all upper and lower cases.

```
sentence = 'The Cat in the Hat'

sentence.upper() #=> 'THE CAT IN THE HAT'
sentence.lower() #=> 'the cat in the hat'
```

20. Uppercase first and last character of a string

As in a past example, we'll target specific indices of the string. Strings aren't mutable in Python so we'll build an entirely new string.

```
animal = 'fish'

animal[0].upper() + animal[1:-1] + animal[-1].upper()
#=> 'FisH'
```

21. Check if a string is all uppercase

Similar to `islower()`, `isupper()` returns `True` only if the whole string is capitalized.

```
'Toronto'.isupper() #=> False
'TORONTO'.isupper() #= True
```

22. When would you use splitlines()?

`splitlines()` splits a string on line breaks.

```
sentence = "It was a stormy night\nThe house creeked\nThe wind
blew."
```

```
sentence.splitlines()
#=> ['It was a stormy night', 'The house creeked', 'The wind
blew.']
```

23. Give an example of string slicing

Slicing a string takes up to 3 arguments, `string[start_index:end_index:step]`.

`step` is the interval at which characters should be returned. So a step of 3 would return the character at every 3rd index.

```
string = 'I like to eat apples'

string[:6] #=> 'I like'
string[7:13] #=> 'to eat'
string[0:-1:2] #=> 'Ilk oetape' (every 2nd character)
```

24. Convert an integer to a string

Use the string constructor, `str()` for this.

```
str(5) #=> '5'
```

25. Check if a string contains only characters of the alphabet

`isalpha()` returns `True` if all characters are letters.

```
'One1'.isalpha()
'One'.isalpha()
```

26. Replace all instances of a substring in a string

Without importing the regular expressions module, you can use `replace()`.

```
sentence = 'Sally sells sea shells by the sea shore'

sentence.replace('sea', 'mountain')
#=> 'Sally sells mountain shells by the mountain shore'
```

27. Return the minimum character in a string

Capitalized characters and characters earlier in the alphabet have lower indexes. `min()` will return the character with the lowest index.

```
min('strings') #=> 'g'
```

28. Check if all characters in a string are alphanumeric

Alphanumeric values include letters and integers.

```
'Ten10'.isalnum() #=> True
'Ten10.'.isalnum() #=> False
```

29. Remove whitespace from the left, right or both sides of a string

`lstrip()`, `rstrip()` and `strip()` remove whitespace from the ends of a string.

```
string = '  string of whitespace    '
string.lstrip() #=> 'string of whitespace    '
string.rstrip() #=> '  string of whitespace'
string.strip() #=> 'string of whitespace'
```

30. Check if a string begins with or ends with a specific character?

`startswith()` and `endswith()` check if a string begins and ends with a specific substring.

```
city = 'New York'

city.startswith('New') #=> True
city.endswith('N') #=> False
```

31. Encode a given string as ASCII

`encode()` encodes a string with a given encoding. The default is `utf-8`. If a character cannot be encoded then a `UnicodeEncodeError` is thrown.

```
'Fresh Tuna'.encode('ascii')
#=> b'Fresh Tuna'

'Fresh Tuna Â'.encode('ascii')
#=> UnicodeEncodeError: 'ascii' codec can't encode character
'\xc2' in position 11: ordinal not in range(128)
```

32. Check if all characters are whitespace characters

`isspace()` only returns `True` if a string is completely made of whitespace.

```
''.isspace() #=> False
' '.isspace() #=> True
'   '.isspace() #=> True
' the '.isspace() #=> False
```

33. What is the effect of multiplying a string by 3?

The string is concatenated together 3 times.

```
'dog' * 3
# 'dogdogdog'
```

34. Capitalize the first character of each word in a string

`title()` will capitalize each word in a string.

```
'once upon a time'.title()
```

35. Concatenate two strings

The additional operator can be used to concatenate strings.

```
'string one' + ' ' + 'string two'
#=> 'string one string two'
```

36. Give an example of using the partition() function

`partition()` splits a string on the first instance of a substring. A tuple of the split string is returned without the substring removed.

```
sentence = "If you want to be a ninja"

print(sentence.partition(' want '))
#=> ('If you', ' want ', 'to be a ninja')
```

37. What does it mean for strings to be immutable in Python?

Once a string object has been created, it cannot be changed. “Modifying” that string creates a whole new object in memory.

We can prove it by using the `id()` function.

```
proverb = 'Rise each day before the sun'
print( id(proverb) )
#=> 4441962336
```

```
proverb_two = 'Rise each day before the sun' + ' if its a
weekday'
print( id(proverb_two) )
#=> 4442287440
```

Concatenating `' if its a weekday'` creates a new object in memory with a new `id`. If the object was actually modified then it would have the same `id`.

38. Does defining a string twice (associated with 2 different variable names) create one or two objects in memory?

For example, writing `animal = 'dog'` and `pet = 'dog'`.

It only creates one. I found this unintuitive the first time I came across it. But this helps python save memory when dealing with large strings.

We'll prove this with `id()`. Notice how both have the same `id`.

```
animal = 'dog'
print( id(animal) )
#=> 4441985688

pet = 'dog'
print( id(pet) )
#=> 4441985688
```

39. Give an example of using maketrans() and translate()

`maketrans()` creates a mapping from characters to other characters. `translate()` then applies that mapping to translate a string.

```
# create mapping
mapping = str.maketrans("abcs", "123S")
```

```
# translate string
"abc are the first three letters".translate(mapping)
#=> '123 1re the firSt three letterS'
```

Notice above how we changed the values of every `a`, `b`, `c` and `s` in the string.

40. Remove vowels from a string

One option is to iterate over the characters in a string via list comprehension. If they don't match a vowel then join them back into a string.

```
string = 'Hello 1 World 2'

vowels = ('a','e','i','o','u')

''.join([c for c in string if c not in vowels])
#=> 'Hll 1 Wrld 2'
```

41. When would you use rfind()?

`rfind()` is like `find()` but it starts searching from the right of a string and return the first matching substring.

```
story = 'The price is right said Bob. The price is right.'
story.rfind('is')
#=> 39
```

. . .

Conclusion

As I often explained to an old product manager, engineers aren't dictionaries of stored methods. But sometimes a little less googling can make coding more seamless and enjoyable.

I hope you crushed this.

If you found it too easy, you may be interested in my other article, 54 Python Interview Questions.

Python Programming Coding Data Science Software Development