Python Strings Cheat Sheet

Python Strings

Introduction to Strings

Strings (str objects) are enclosed in single or double quotes (' or "). Just be consistent within your code and don't mix up " with '

```
message1 = 'I love Python Programming!'
message2 = "I love Python Programming!"

# The print() function displays the content  of the variable at the terminal
print(message1)

# hello1 = 'Hi there! I\'m Andrei'    # => error, cannot use ' inside ' ' or " inside " "
hello1 = 'Hi there! I\'m Andrei'      # => correct. ' inside ' ' must be escaped  using \
hello2 = "Hi there! I'm Andrei"       # you can use ' inside " " or " inside ' '

# Instructions between triple quotes (""" or ''') are treated as comments, but they are not comments but documentation.. It's recommended to use only # for commenting individual lines

"""
This is a documentation (treated as a comment)
a = 5
print(a)
Comment ends here.
"""

# Defining a multiline string
languages = """ I learn Python,
Java,
Go,
```

```
PHP. Let's get started!
"""
print(languages)

# \n is a new line
print('Hello \nWorld!')    # => it displays Hello World! on 2 lines

# \ is used to escape characters that have a special meaning
info = '\\ character is used to escape characters that have a special meaning'
print(info)     # => \ character is used to escape characters that have a special meaning
```

User Input and Type Casting

```
# Getting user input
user_input = input("Enter some data: ")    # returns the data as a string

# To perform arithmetic operations you must convert data to int or float
a = input('Enter a:')          # => a is type string
b = int(input('Enter b:'))    # => b is type int
c = float(a) * b               # => multiplying float by int

# Type casting
a = 3       # => type int
b = 4.5     # => type float
c = '1.2'    # => type str

print('a is ' + str(a))    # => str(a) returns a string from an int
print('b is ' + str(b))    # => str(b) returns a string from a float
d = b * float(c)           # => here I multiply two floats, d is type float.

str1 = '12 a'
# float(str1)    # => error
```

String Indexing, Operations and Slicing

```
# A string is an ordered sequence of UTF-8 characters
# Indexing starts from 0
language = 'Python 3'
language[0]             # => 'P'
language[1]             # => 'y'
language[-1]            # => '3'
language[-2]            # => ' '
```

```
"This is a string"[0]    # => 'T'

# language[100]         # => IndexError: string index out of range

# Cannot modify a string, it's immutable
# language[0] = 'J'      # => TypeError: 'str' object does not support item assignment

# len() returns the length of the string
len("This is a string")  # => 16

# Strings can be concatenated with + and repeated with *
print('Hello ' + 'world!')      # => 'Hello world!'
print('Hello ' * 3)             # => 'Hello Hello Hello '
print('PI:' + str(3.1415))    # => Can concatenate only strings

# Slicing returns a new string
# Slicing format [start:end:step ] where start is included, end is excluded, step is by default 1
movie = 'Star Wars'
movie[0:4]       # => 'Star' -> from index 0 included to index 4 excluded
movie[:4]        # => 'Star' -> start index defaults to zero
movie[2:]        # => 'ar Wars' -> end index defaults to the index of the last char of the string
movie[::]        # => 'Star Wars'
movie[2:100]    # => 'ar Wars -> slicing doesn't return error when using index out of range
movie[1:6:2]    # => 'trW' -> from index 1 included to 6 excluded in steps of 2
movie[6:1:-1]   # => 'aW ra' -> from index 6 included to index 1 excluded in steps of -1 (backwards)
movie[::-1]      # => 'sraW ratS -> reverses the string
```

Formatting Strings

```
price = 1.33
quantity = 5

# f-string literals (Python 3.6+) - PYTHONIC and recommended!
f'The price is {price}'                    # => 'The price is 1.33'
f'Total value is {price * quantity}'       # => 'Total value is 6.65'
f'Total value is {price * quantity:.4f}'   # => 'Total value is 6.6500' -> displaying with 4 decimals

# Classical method (old, not recommended)
'The price is {}'.format(price)                          # => 'The price is 1.33'
'The total value is {}'.format(price * quantity)         # => 'The total value is 6.65'
'The total value is {:.4f}'.format(price * quantity)    # => 'The total value is 6.6500' -> displaying with
4 decimals
```

```
'The price is {} and the total value is {}'.format(price, price * quantity)      # => 'The price is 1.33 and the total value is 6.65'
'The price is {0} and the total value is {1}'.format(price, price * quantity)   # => 'The price is 1.33 and the total value is 6.65'
'The total value is {1} and the price is {0}'.format(price, price * quantity)   # => 'The total value is 6.65 and the price is 1.33'

print('The total value is ', price * quantity)      # => 'The total value is 6.65'
```

String Methods

```
dir(str)            # => lists all string methods
help(str.find)    # => displays the help of a method

# All string methods return a new string but don't modify the original one
my_str = 'I learn Python Programming'

# str.upper() returns a copy of the string converted to uppercase.
my_str.upper()    # => 'I LEARN PYTHON PROGRAMMING'

# str.lower() returns a copy of the string converted to uppercase.
my_str.lower()    # => 'i learn python programming'

# str.strip() removes leading and trailing whitespace
my_ip = '    10.0.0.1 '
my_ip.strip()    # => '10.0.0.1'

my_ip = '$$$10.0.0.1$$'
my_ip.strip('$')    # => '10.0.0.1'

# str.lstrip() remove all leading whitespace in string
my_ip.lstrip()    # => '10.0.0.1 '

# str.rstrip() Remove all trailing whitespace of string
my_ip.rstrip()    # => '    10.0.0.1'

my_str = 'I learn Python'
my_str.replace('Python', 'Go')    # => 'I learn Go'

# str.split() returns a list from a string using a delimiter
my_ip.split('.')    # => ['10', '0', '0', '1']

# By default the delimiter is one or  more whitespaces
my_list = my_str.split()     # => my_list = ['I', 'learn', 'Python', 'Programming']
```

```
# str.join() concatenates the elements of a list into a string
':'.join( ['10', '0', '0', '1'])    # => '10:0:0:1'

# in and not in operators test membership
my_ip = '    10.0.0.1 '
'10' in my_ip          # => returns True
'10' not in my_ip    # => returns False
'20' in my_ip          # => returns False

# Other string methods
my_str = 'this is a string. this is a string'

# str.find() returns the first index in my_str where substring 'is' is found or -1 if it didn't find the
substring
my_str.find('is')     # => 2
my_str.find('xx')    # => -1

# str.capitalize() returns a capitalized copy of the string
my_str.capitalize()    # => 'This is a string. this is a string'

# str.isupper() returns True if my_str is an uppercase string, False otherwise
my_str.isupper()    # => False

# str.islower() returns True if my_str is a lowercase string, False otherwise
my_str.lower().islower()    # => True

# str.count(s) returns the number of occurrences of substring 's' in my_str
my_str.count('s')    # => 6

'0123123'.isdigit()        # => True
'0123123x'.isdigit()      # => False
'abc'.isalpha()              # => True
'0123123x'.isalnum()    # => True

# str.swapcase() inverts case for all letters in string
my_str1 = 'Hi everyone!'
my_str1.swapcase()      # => 'hI EVERYONE!'
```