វិទ្យាស្ថានសាន់រ៉ាយស៍

SUNRISE INSTITUTE

Programming for Data science

EMBARKING ON A JOURNEY INTO DATA SCIENCE

Instructor: YA MANON

Text or String in Python

Data scientists use various methods, processes, algorithms, and systems to extract insights from data. **Python's simple syntax makes it one of the easiest languages to learn**, which is a benefit to data scientists who don't come from an engineering background or haven't had extensive programming experience.

STRINGS or TEXT

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

In this section we'll review the fundamentals of **strings**, a date type used to store text, and cover functions and methods to manipulate them

TOPICS WE'LL COVER:

- String basic
- String Arithmetic
- Indexing & Slicing
- String Methods
- F-Strings

GOALS FOR THIS SECTION:

- Create strings from scratch or from other data types
- Learn to manipulate strings using arithmetic, indexing, slicing, and string methods
- Use f-strings to incorporate variables into strings and make them dynamic

STRING BASICS

Strings or Text. **Python** can manipulate text (represented by type str, so-called "strings").

We can create a string by using quotation marks, or converting from numbers:

Single *quotes*

```
new_string = 'This is a "string".'
print(new_string)
```

```
This is a "string".
```

Double *quotes*

```
newer_string = "Double quotes allow us to use ' inside."
print(newer_string)
```

```
Double quotes allow us to use ' inside.
```

Triple *quotes*

```
# triple quotes
newerer_string = '''Triple quotes allow us to
write multiline strings. We also don't
have to worry about "errors" due to using
single or double quotes. Pretty cool!
'''
print(newerer_string)
```

```
Triple quotes allow us to
write multiline strings. We also don't
have to worry about "errors" due to using
single or double quotes. Pretty cool!
```

Convert *to string*

```
number = 22.5
string = str(number)

string
```

```
'22.5'
```

The quotes indicate this is a string

```
print(string)
```

```
22.5
```

Using print() cleans up the quotes

STRING ARITHMETIC

Some **arithmetic operators** are compatible with strings

You can **concatenate**strings using +

```
'Snow' + 'board'
```

```
'Snowboard'
```

```
subject = 'My'
noun = 'Maven snowboard'
verb = 'cost'
price = 22.55

subject + ' ' + noun + ' ' + verb + ' $' + str(price) +'!'
```

```
'My Maven snowboard cost $22.55!'
```

You can concatenate any number of strings The addition of ' ' to add spaces, as well as
adding punctuation like '!' or '$' is common

You can **repeat** strings using *

```
'Repeat 3 times: ' + ("I will not steal my coworker's pen! ") * 3
```

```
"Repeat 3 times: I will not steal my coworker's pen! I will not steal
my coworker's pen! I will not steal my coworker's pen! "
```

Repeating strings is not very common in practice

STRING INDEXING

ទីស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

String basic

String Arithmetic

Indexing & Slicing

String Methods

F-Strings

Strings, along with sequence data types, can be navigated via their **index**

- Python is **0-indexed**, which means that the first item in a sequence has an index of 0, not 1

```
message = 'I hope it snows tonight!'
```

message

0 I
1
2 h
3 o
4 p
…
22 t
23 !

```
message[0]
```

```
'I'
```

```
message[1]
```

```
' '
```

```
message[2]
```

```
'h'
```

String[index] *returns a specified character:*

- *String[0] returns the first character*
- *String[1] returns the second character*
- *String[2] returns the third character*

Indexing is a critical concept to master for data analysis, and while the 0-index can be confusing at first, with practice it will be second nature

STRING INDEXING

Strings, along with sequence data types, can be navigated via their **index**

- Python is **0-indexed**, which means that the first item in a sequence has an index of 0, not 1

```
message = 'I hope it snows tonight!'
```

```
message[-1]
'!'
```

String[index *returns a specified character:*

1 .

String[-1] returns the last character

message

-24 I -23
…
-5 i -4 g
-3 h
-2 t
-1 !

*You can also access individual characters in a text string by specifying their **negative** index*
Negative indices begin at the end of the object and work backwards
Since there is no negative 0, this starts at -1

⚠ Indexing a value **greater than the length of the string or object** results in an **IndexError**

```
message[52]
```

```
IndexError: string index out of range
```

STRING SLICING

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

- String basic
- String Arithmetic
- Indexing & Slicing
- String Methods
- F-Strings

Slicing returns multiple elements in a string, or sequence, via indexing

```
[start:stop:step size]
```

Index value to start with (default is 0)

Index value to stop at, not inclusive (default is all the values after the start)

Amount to increment each subsequent index (default is 1)

As with indexing, **slicing is a critical concept to master for data analysis**, as it allows you to manipulate a wide variety of data formats

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[0:6:1]
```

```
'I hope'
```

How does this code work?

- **Start: 0**–start with the first element in the string
- **Stop: 6** –stop at the seventh element (index of 6) in the string without including it
- **Step size:1**–return every single element in the range

STRING SLICING

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

String basic

String Arithmetic

Indexing & Slicing

String Methods

F-Strings

EXAMPLE *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[:6]
```

```
'I hope'
```

How does this code work?

- **Start:** **0**–start with the first element in the string by default
- **Stop:** **6** –stop at the seventh element (index of 6) in the string without including it
- **Step size:** **1** –return every single element in the range by default

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[2:6]
```

```
'hope'
```

How does this code work?

- **Start:2**–start with the third element (index of 2) in the string
- **Stop:6** –stop at the seventh element (index of 6) in the string without including it
- **Step size:1** –return every single element in the range by default

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[6:]
```

```
' it snows tonight!'
```

How does this code work?

- **Start: 6**–start with the seventh element (index of 6) in the string
- **Stop: 25** –stop at the end of the string
- **Step size: 1** –return every single element in the range by default

STRING SLICING

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

String basic

String Arithmetic

Indexing & Slicing

String Methods

F-Strings

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[0:6:2]
```

```
'Ihp'
```

How does this code work?

- **Start: 0**–start with the first element in the string
- **Stop: 6** –stop at the seventh element (index of 6) in the string without including it
- **Step size: 2** –return every other element in the range

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[::-1]
```

```
'!thginot swons ti epoh I'
```

How does this code work?

- **Start:-1**–start with the last element in the string by default, due to the negative step size
- **Stop:-25** –stop at the beginning of the string
- **Step size:-1** –return every single element in the range, going backwards

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[-1:-9:-1]
```

```
'!thginot'
```

How does this code work?

- **Start:-1**–start with the last element in the string
- **Stop:-9** –stop at the ninth element from the end of the string without including it
- **Step size:-1** –return every single element in the range, going backwards

STRING SLICING

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

- String basic
- String Arithmetic
- Indexing & Slicing
- String Methods
- F-Strings

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[-9:-1:1]
```

```
' tonight'
```

How does this code work?

- **Start:-9**–start with the ninth element from the end of the string
- **Stop:-1** –stop at last element in the string without including it
- **Step size:1** –return every single element in the range

STRING SLICING

EXAMPLE | *Grabbing components of a text string*

```
message = 'I hope it snows tonight!'
message[-9::]
```

```
' tonight!'
```

How does this code work?

- **Start:-9**–start with the ninth element from the end of the string
- **Stop:25** –stop at the end of the string by default
- **Step size:1** –return every single element in the range by default

THE LENGTH FUNCTION

ទីឃ្លាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

The **len()** function returns the number of elements in an iterable

```
# len returns the length of strings
len('snowboard')
```

```
9
```

When used on a string, it returns the number of characters

```
# len will return the length of any iterable object
len(['beginner', 'enthusiast', 'pro'])
```

```
3
```

When used on a list, it returns the number of elements within it (more on lists later!)

METHODS

Methods are functions that only apply to a specific data type or object

- We call them by following the object with a period then the method name

```
message = 'I hope it snows tonight!'
```

```
message.replace('snow', 'rain')
```

```
'I hope it rains tonight!'
```

*replace is a **string method** being applied to message, which is a string*

```
number = 256
number.replace(5, 9)
```

```
AttributeError: 'int' object has no attribute 'replace'
```

If you try to apply a string method to an integer, you will receive an ***AttributeError***

STRING METHODS

ទីវិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

String basic

String Arithmetic

Indexing & Slicing

String Methods

F-Strings

find	*Returns the index of the first instance of a given string*	**.find(*string*)**
upper	*Converts all characters in a string to uppercase*	**.upper()**
lower	*Converts all characters in a string to lowercase*	**.lower()**
strip	*Removes leading and trailing spaces, or any other character specified, from a string*	**.strip(*optional string*)**
replace	*Substitutes a given string of characters with another*	**.replace(*string to replace*,*replacement*)**
split	*Splits a list based on a given separator (space by default)*	**.split(*optional string*)**
join	*Joins list elements into a single string separated by a delimiter*	***delimiter*.join(*list*)**

For a list of string methods, visit: ***https://docs.python.org/3/library/stdtypes.html#string-methods***

FIND

The **.find()** method searches for a specified value within a string and returns the index of its first instance

```
message = 'I hope it snows tonight!'
```

```
message.find('snow')
```

```
10
```

__'snow'__ first appears in the 11th character (index value of 10)

- **.find(*string*)**

```
message.find('rain')
```

```
-1
```

-1 is returned if the specified value isn't found

UPPER & LOWER

The **.upper()** and **.lower()** methods return a given string with all its characters in uppercase or lowercase respectively

```
message = 'I hope it snows tonight!'
```

```
print(message.upper())
```

```
I HOPE IT SNOWS TONIGHT!
```

```
print(message.lower())
```

```
i hope it snows tonight!
```

STRIP

The **.strip()** method removes leading and trailing spaces *(by default)*, or any other specified character, from a string

- **.lstrip()** removes all leading spaces, and **.rstrip()** removes all trailing spaces

```
message = "      I love the space bar          "

message.strip()
```

```
'I love the space bar'
```

This removes leading and trailing spaces by default

•**.strip(*optional string*)**

```
message = '.Remove the punctuation.'

message.strip('.')
```

```
'Remove the punctuation'
```

This removes leading and trailing periods

REPLACE

ទិទ្យាស្ថានសាន់រ៉ាយស៍
SUNRISE INSTITUTE

The **.replace()** method substitutes a specified string of characters with another

- This is a common method to remove or change punctuation while cleaning text

```
message = 'I hope it snows tonight!'
```

```
message.replace('snow', 'rain')
```

```
'I hope it rains tonight!'
```

*This replaces all instances of **'snow'** with **'rain'***

•.replace(*string to replace*,*replacement*)

```
message.replace(' ', '')
```

```
'Ihopeitsnowstonight!'
```

*This replaces all **spaces** with a **empty string**, which creates one continuous word*

F-STRINGS

SUNRISE INSTITUTE

- String basic
- String Arithmetic
- Indexing & Slicing
- String Methods
- F-Strings

You can include variables in strings by using **f-strings**

- This allows you to change the output text based on changing variable values

```
name = 'Chris'
role = 'employee'

print(f"{name} is our favorite {role}, of course!")
```

```
Chris is our favorite employee, of course!
```

Add an **f** before the quotation marks to indicate an f-string, and place the variable names into the string by using curly braces {}

```
name = 'Anand'
role = 'customer'

print(f"{name} is our favorite {role}, of course!"
```

```
Anand is our favorite customer, of course!
```

Note that the f-string code doesn't change, but the output does change if the variables get assigned new values

KEY TAKEAWAYS

- Strings are used to store **text** and other sequences of characters
  - *Analysts interact with strings in categorical data, or in large bodies of text that need to be mined for insight*
- Access single characters with **indexing**, and multiple characters with **slicing**
  - *Indexing & slicing are used with many other data types beyond strings, so they are critical to master*
- Learn and leverage **string methods** to facilitate working with strings
  - *You don't need to memorize every method and their syntax on day 1, but it's important to be aware of them*
- Use **f-strings** to incorporate variables into your strings
  - *This will make your text strings dynamic and allow you to avoid writing repetitive code*