Introduction to string manipulation

REGULAR EXPRESSIONS IN PYTHON

Maria Eugenia Inzaugarat
Data Scientist

datacamp

You will learn

- **String manipulation**
  - e.g. replace and find specific substrings
- **String formatting**
  - e.g. interpolating a string in a template
- **Basic and advanced regular expressions**
  - e.g. finding complex patterns in a string

Why it is important

- Clean dataset to prepare it for text mining or sentiment analysis
- Process email content to feed a machine learning algorithm that decides whether an email is spam
- Parse and extract specific data from a website to build a database

Strings

- Sequence of characters
- Quotes

```
my_string = "This is a string"
my_string2 = 'This is also a string'
```

```
my_string = 'And this? It's the wrong string'
```

```
my_string = "And this? It's the correct string"
```

More strings

- Length

```
my_string = "Awesome day"
len(my_string)
```

```
11
```

- Convert to string

```
str(123)
```

```
'123'
```

Concatenation

- Concatenate: `+` operator

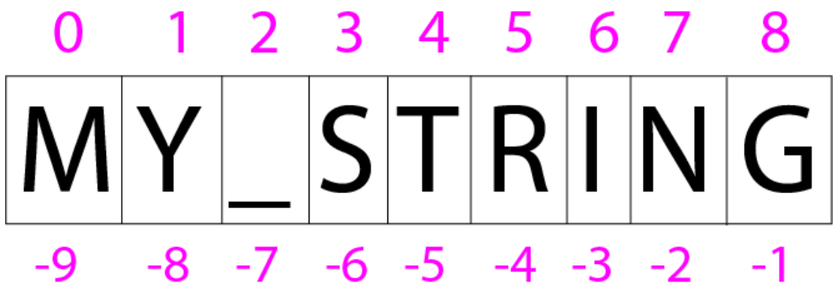
```
my_string1 = "Awesome day"
my_string2 = "for biking"
```

```
print(my_string1+" "+my_string2)
```

```
Awesome day for biking
```

Indexing

- **Bracket notation**




```
my_string = "Awesome day"
```

```
print(my_string[3])
```

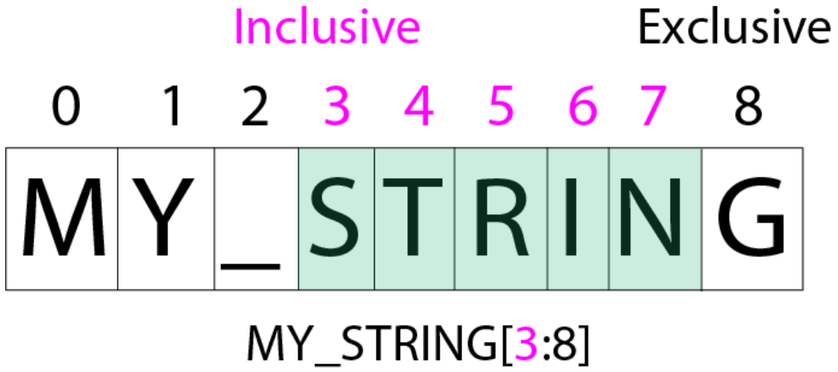
```
s
```

```
print(my_string[-1])
```

```
y
```

Slicing

- Bracket notation




```
my_string = "Awesome day"
print(my_string[0:3])
```

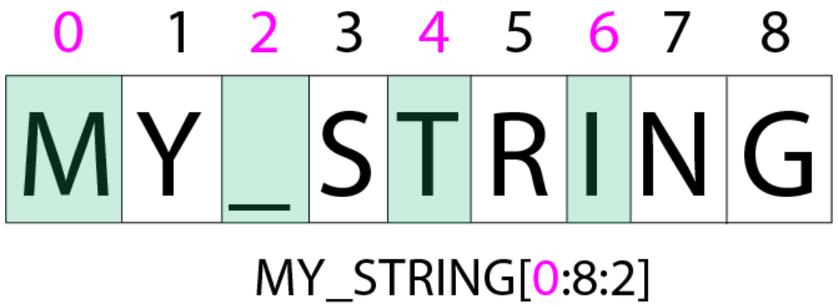
```
Awe
```

```
print(my_string[:5])
print(my_string[5:])
```

```
Aweso
me day
```

Stride

- Specifying stride




```
my_string = "Awesome day"
print(my_string[0:6:2])
```

```
Aeo
```

```
print(my_string[::-1])
```

```
yad emosewA
```

Let's practice!

REGULAR EXPRESSIONS IN PYTHON

String operations

REGULAR EXPRESSIONS IN PYTHON

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Adjusting cases

```
my_string = "tHis Is a niCe StriNg"
```

- Converting to lowercase

```
print(my_string.lower())
```

```
this is a nice string
```

- Converting to uppercase

```
print(my_string.upper())
```

```
THIS IS A NICE STRING
```

```
my_string = "tHis Is a niCe StriNg"
```

- Capitalizing the first character

```
print(my_string.capitalize())
```

```
This is a nice string
```

Splitting

```
my_string = "This string will be split"
```

- Splitting a string into a list of substrings

```
my_string.split(sep=" ", maxsplit=2)
```

```
['This', 'string', 'will be split']
```

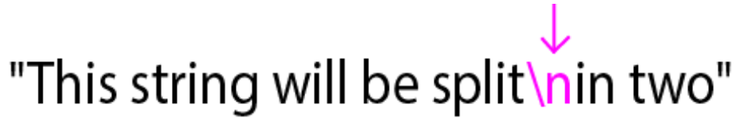
```
my_string.rsplit(sep=" ", maxsplit=2)
```

```
['This string will', 'be', 'split']
```

```
my_string = "This string will be split\nin two"
print(my_string)
```

```
This string will be split
in two
```

Escape Sequence	Character
\n	Newline
\r	Carriage return



- Breaking at line boundaries

```
my_string = "This string will be split\nin two"
```

```
my_string.splitlines()
```

```
['This string will be split', 'in two']
```

Joining

- Concatenate strings from list or another iterable

sep.join(iterable)

```
my_list = ["this", "would", "be", "a", "string"]
print(" ".join(my_list))
```

```
this would be a string
```

Stripping characters

- Strips characters from left to right: `.strip()`

```
my_string = " This string will be stripped\n"
```

```
my_string.strip()
```

```
'This string will be stripped'
```

```
my_string = " This string will be stripped\n"
```

- Remove characters from the right end

```
my_string.rstrip()
```

```
' This string will be stripped'
```

- Remove characters from the left end

```
my_string.lstrip()
```

```
'This string will be stripped\n'
```

Let's practice!

REGULAR EXPRESSIONS IN PYTHON

Finding and replacing

REGULAR EXPRESSIONS IN PYTHON

Maria Eugenia Inzaugarat
Data scientist

datacamp

Finding substrings

- Search target string for a specified substring.


```
my_string = "Where's Waldo?"
my_string.find("Waldo")
```

```
8
```

```
my_string.find("Wenda")
```

```
-1
```

Finding substrings

- Search target string for a specified substring.


```
my_string = "Where's Waldo?"
```

```
my_string.find("Waldo", 0, 6)
```

```
-1
```

Index function

- Similar to `.find()`, search target string for a specified substring.


```
my_string = "Where's Waldo?"
my_string.index("Waldo")
```

```
8
```

```
my_string.index("Wenda")
```

```
File "<stdin>", line 1, in <module>
ValueError: substring not found
```

Index function

- Similar to `.find()`, search target string for a specified substring.


```
my_string = "Where's Waldo?"
```

```
try:
    my_string.index("Wenda")
except ValueError:
      print("Not found")
```

```
"Not found"
```

Counting occurrences

- Return number of occurrences for a specified substring.


```
my_string = "How many fruits do you have in your fruit basket?"
my_string.count("fruit")
```

```
2
```

```
my_string.count("fruit", 0, 16)
```

```
1
```

Replacing substrings

- Replace occurrences of substring with new substring.


```
my_string = "The red house is between the blue house and the old house"
print(my_string.replace("house", "car"))
```

```
The red car is between the blue car and the old car
```

```
print(my_string.replace("house", "car", 2))
```

```
The red car is between the blue car and the old house
```

Wrapping up

- **String manipulation:**
  - Slice and concatenate
  - Adjust cases
  - Split and join
  - Remove characters from beginning and end
  - Finding substrings
  - Counting occurrences
  - Replacing substrings

Let's practice!

REGULAR EXPRESSIONS IN PYTHON