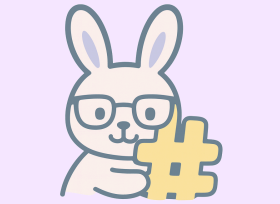
Python String Basics



ALGORITHM
WITH CHELSEA

What is a string?

- A **string** is just **text** in Python.
- We put text inside quotes `" "` or `' '`.

Examples:

```
name = "Alice"
pet = "bunny"
greeting = "Hello, world!"
empty_string = ""
```

String concatenation with +

Example:

```
first_name = "Chelsea"
last_name = "Wang"

full_name = first_name + last_name
print(full_name)
```

String concatenation with `+`

Example:

```
first_name = "Chelsea"
last_name = "Wang"

full_name = first_name + last_name
print(full_name)
```

Output:

```
ChelseaWang
```

This works only with **strings + strings**.

Add spaces when joining

We can add a space `" "` in the middle:

```
first_name = "Chelsea"
last_name = "Wang"

full_name = first_name + " " + last_name
print(full_name)
```

Add spaces when joining

We can add a space `" "` in the middle:

```
first_name = "Chelsea"
last_name = "Wang"

full_name = first_name + " " + last_name
print(full_name)
```

Output:

```
Chelsea Wang
```

Three strings are concatenated.

Q1 ✏️ Join first and last name

What is the output?

```
first_name = "Harry"
last_name = "Potter"

full_name = first_name + " " + last_name
print(full_name)
```

Q1 ✏️ Join first and last name

What is the output?

```
first_name = "Harry"
last_name = "Potter"

full_name = first_name + " " + last_name
print(full_name)
```

Output:

```
Harry Potter
```

Take one character from a string

- **Index** is the **position** of each character.
- We can use **square brackets** `[index]` to get **one character**.

Take one character from a string

- **Index** is the **position** of each character.
- We can use **square brackets** `[index]` to get **one character**.

Example:

```
word = "ABCDEF"

print(word[0])   # 'A'
print(word[1])   # 'B'
print(word[2])   # 'C'
```

Remember: Python starts counting from **0**, not **1**.

Indices: how Python counts

Let us look at `"ABCDEF"` again:

```
Index:   0   1   2   3   4   5
String:  A   B   C   D   E   F
```

Indices: how Python counts

Let us look at `"ABCDEF"` again:

```
Index:   0   1   2   3   4   5
String:  A   B   C   D   E   F
```

So:

- `word[0]` is `"A"`
- `word[1]` is `"B"`
- `word[2]` is `"C"`

Slice: a piece of a string

A **slice** takes a **range** of characters from a string.

We use:

```
word[start:stop]
```

Slice: a piece of a string

A **slice** takes a **range** of characters from a string.

We use:

```
word[start:stop]
```

Rules:

- `start` = where we start (**included**).
- `stop` = where we stop (**not included**).
- Python counts indices from **0**.

Q2 🔍 Slice of a string

What is the output?

```
word = "ABCDEF"
print(word[0:2])
print(word[2:6])
```

Q2 🔍 Slice of a string

What is the output?

```
word = "ABCDEF"
print(word[0:2])
print(word[2:6])
```

Indices of the characters:

```
Index:   0   1   2   3   4   5
String:  A   B   C   D   E   F
```

- `word[0:2]` → letters at index 0 and 1 → `"AB"`
- `word[2:6]` → letters at index 2, 3, 4, 5 → `"CDEF"`

Convert integer to string

`str()` converts `int` to `str`.

```
age = 11              # int
age_text = str(age)   # str
```

Convert integer to string

`str()` converts `int` to `str`.

```
age = 11              # int
age_text = str(age)   # str
```

We can ask Python what the type is:

```
print(type(age))
print(type(age_text))
```

Convert integer to string

`str()` converts `int` to `str`.

```
age = 11              # int
age_text = str(age)   # str
```

We can ask Python what the type is:

```
print(type(age))
print(type(age_text))
```

Output:

```
<class 'int'>
<class 'str'>
```

String + Number: It doesn't work!

This code will **not** work:

```
age = 11

message = "I am " + age + " years old."
print(message)
```

String + Number: It doesn't work!

This code will **not** work:

```
age = 11

message = "I am " + age + " years old."
print(message)
```

Output:

```
TypeError: can only concatenate str (not 'int') to str
```

Python **doesn't** allow concatenating a string with an integer.

Fix the problem with `str()`

This time we convert the number to a string:

```
age = 11

message = "I am " + str(age) + " years old."
print(message)
```

Fix the problem with `str()`

This time we convert the number to a string:

```
age = 11

message = "I am " + str(age) + " years old."
print(message)
```

Output:

```
I am 11 years old.
```

Now every piece we concatenate with `+` is a **string**.

Q3 🍎 Number and text together

Fill in the blanks:

```
apples = 5

message = "I have " + __________ + " apples."
print(message)
```

Expected output:

```
I have 5 apples.
```

Hint: we need to convert the number to a string.

Q3 🍎 Number and text together

Fill in the blanks:

```
apples = 5

message = "I have " + str(apples) + " apples."
print(message)
```

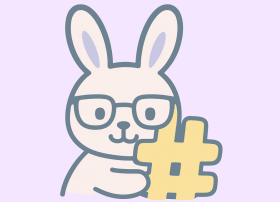
Expected output:

```
I have 5 apples.
```

Hint: we need to convert the number to a string.

Summary

- A **string** is text in quotes, like `"hello"`.
- Use `+` to **concatenate** strings.
- `+` can only concatenate **strings**.
- **string** `+` **number** doesn't work.
- Use `str()` to convert a number to a string.



ALGORITHM
WITH CHELSEA

Summary (continued)

- Index: use `word[i]` to take **one character**.
  - Python counts from **0**.
  - `word[0]` is actually the first character.
- Slice: Use `word[start:stop]` to take a substring.
  - `start` is included.
  - `stop` is **not** included.