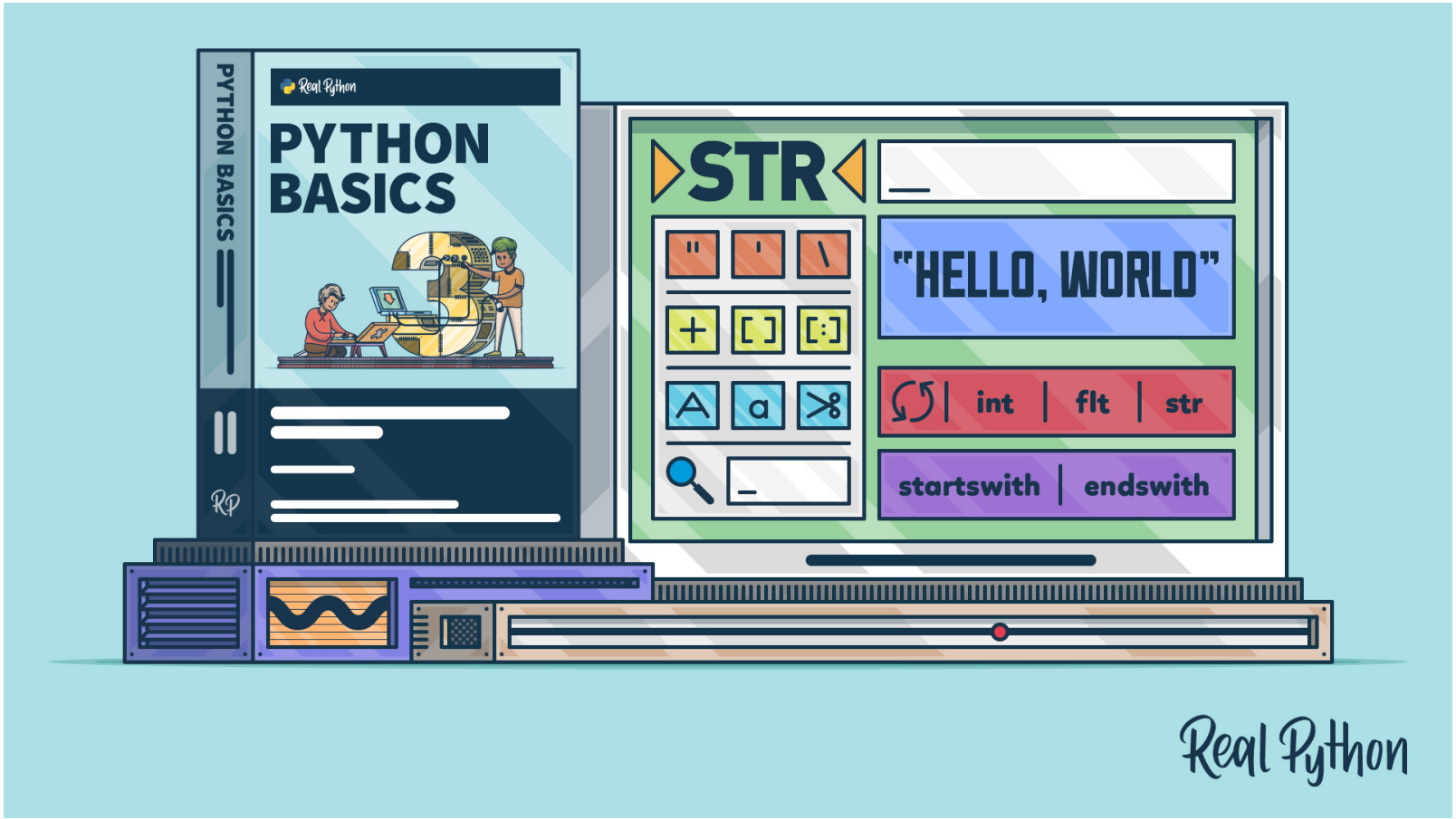
Python Basics - Strings and String Methods



Python Basics - Strings and String Methods

Python Basics - Strings and String Methods

In this course you'll:

- Manipulate strings with string methods
- Work with user input
- Deal with strings of numbers
- Format strings for printing

Python Basics - Strings and String Methods

Along the way you'll also learn about:

- f-Strings
- Multiline strings
- String indexing and slicing
- String immutablility
- Discovering additional methods

Table of Contents

Table of Contents

What is a String?

Strings are one of the *fundamental* Python data types

- An example string
  ```
  my_string = "Hello, World"
  ```
- Data Types: What kind of data a value represents
  - Strings are used to represent text
- Other types:
  - Numeric types - int (integers), float (floating point numbers)
  - Booleans (True, False)
  - Sequence types - list, tuple, string

What is a String?

Strings are a **fundamental data type**

- Can’t be broken into smaller values of a different type
- Unlike a **compound data type**, which are known as data structures

What is a String?

The string data type has a special abbreviated name in Python: `str`

- `type()` shows the data type of a given value

What is a String?

Strings have three important properties:

1. Contain individual letters or symbols called **characters**
2. Have a **length**, defined as the number of characters contained
3. Characters in a string appear in a sequence, each character has a numbered position

What is a String?

- Either single quotes or double quotes can be used
- As long as the same type is used at the beginning and end of the string

```
string1 = 'Hello, World'
string2 = "1234"
```

- Text surrounded by quotation marks creates a **string literal**
  - The name indicates a string literally written into your code

What is a String?

String literals are written in a variety of ways:

- Single quotes: `'allows embedded "double" quotes'`
- Double quotes: `"allows embedded 'single' quotes"`
- Triple quoted:
  - `'''Three single quotes'''`
  - `"""Three double quotes"""`

What is a String?

Strings can contain any valid Unicode character

```
"We're #1!"
"1234"
"×Pýŧħøŋ×"
"🐶😎💻"
```

What is a String?

Determine the Length of a String

- The built-in `len()` function
- Returns the number of characters contained in a string, including spaces

What is a String?

Multiline Strings

- PEP 8 recommends each line of Python code contain no more than 79 characters – including spaces
- Two techniques to break a string across multiple lines
  - Add a backslash (`\`) at the end of all but the last line
  - Use triple quotes as delimiters (`""" or '''`)

What is a String?

Practice what you've learned with these exercises:

1. Print a string that uses double quotation marks inside the string
2. Print a string that uses an apostrophe inside the string
3. Print a string that spans multiple lines with whitespace preserved
4. Print a string that is coded on multiple lines but gets printed on a single line

Table of Contents

Concatenation, Indexing, and Slicing

- Concatenation: Joins two strings together
- Indexing: Gets a single character from a string
- Slicing: Gets several characters from a string at once

Concatenation, Indexing, and Slicing

To combine or **concatenate** strings

- Use the `+` operator
- `string + string`

Concatenation, Indexing, and Slicing

To combine or **concatenate** strings

- Use the `+` operator
- `string + string`

```
>>> string1 = "abra"
>>> string2 = "cadabra"
>>> magic_string = string1 + string2
>>> magic_string
'abracadabra'
```

Concatenation, Indexing, and Slicing

String Indexing

Each character in a string has a numbered position called an **index**

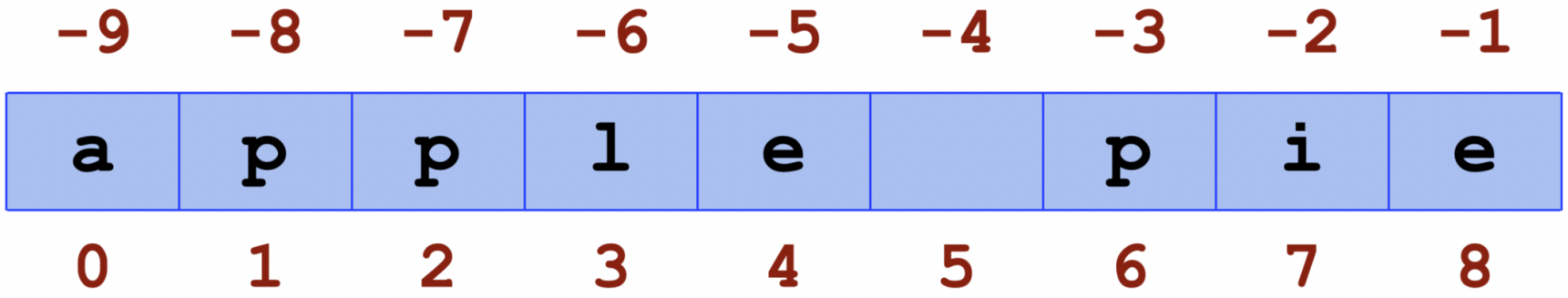
- Individual characters can be accessed by using the index
- Place the **index** number inside a pair of square brackets after the string
  - `my_string[3]`
- The index count starts with zero
  - Most programming languages do this
  - Be careful of **off-by-one errors**

Concatenation, Indexing, and Slicing

String Indexing

Concatenation, Indexing, and Slicing

Negative Indexing



Concatenation, Indexing, and Slicing

String Slicing

- To extract a portion of a string, called a **substring**
- Insert a colon between two index numbers `[0:4]`

Concatenation, Indexing, and Slicing

String Slicing

- To extract a portion of a string, called a **substring**
- Insert a colon between two index numbers `[0:5]`

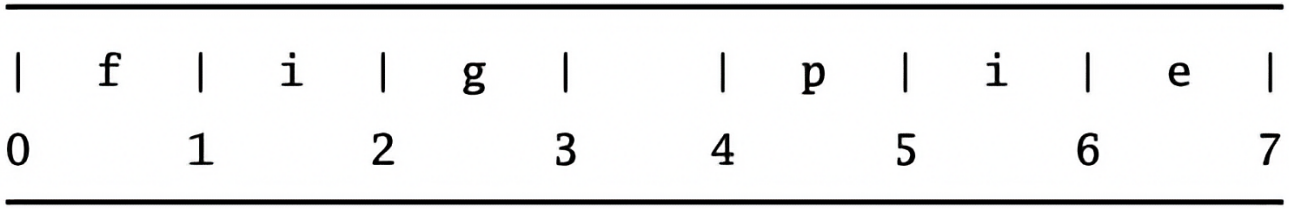
```
>>> dessert = "apple pie"
>>> dessert[0:5]
'apple'
```

- A slice of `[x:y]` starts from the character at index `x`, and goes up to, but does not include the character at the index `y`.

Concatenation, Indexing, and Slicing

String Slicing

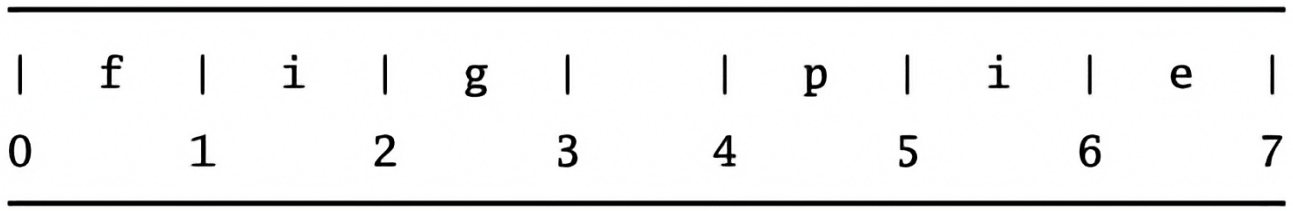
For slicing, indicies behave more like boundaries around characters



Concatenation, Indexing, and Slicing

String Slicing

For slicing, indicies behave more like boundaries around characters




```
>>> new_dessert = "fig pie"
>>> new_dessert[0:3]
'fig'
>>> new_dessert[3:7]
' pie'
```

Concatenation, Indexing, and Slicing

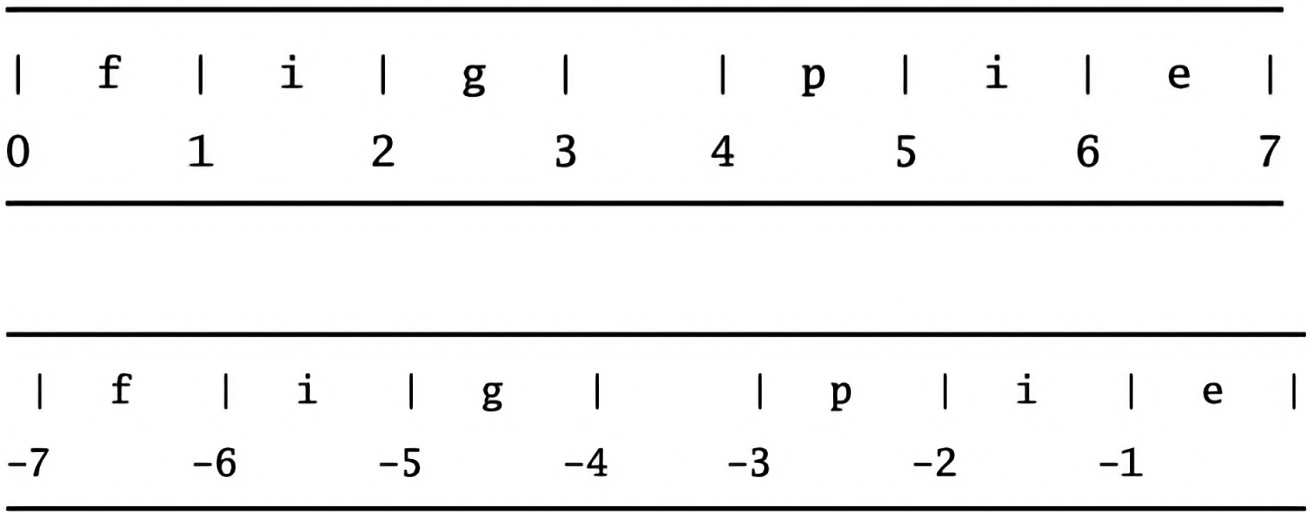
String Slicing

- Negative slicing works by the same rules
- A slice of `[x:y]` starts from the character at index `x`, and goes up to, but does not include the character at the index `y`.

Concatenation, Indexing, and Slicing

String Slicing

- Negative slicing works by the same rules
- A slice of `[x:y]` starts from the character at index `x`, and goes up to, but does not include the character at the index `y`.



Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character

Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character

```
>>> dessert = "apple pie"
>>> dessert[:5]
'apple'
```

Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character
- Omitting the index after the colon, ends with the last character

Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character
- Omitting the index after the colon, ends with the last character

```
>>> dessert = "apple pie"
>>> dessert[:5]
'apple'
>>> dessert[6:]
'pie'
```

Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character
- Omitting the index after the colon, ends with the last character
- Omitting both indexes returns the whole string

Concatenation, Indexing, and Slicing

- Omitting the index before the colon, starts with first character
- Omitting the index after the colon, ends with the last character
- Omitting both indexes returns the whole string

```
>>> dessert = "apple pie"
>>> dessert[:5]
'apple'
>>> dessert[6:]
'pie'
>>> dessert[:]
'apple pie'
```

Concatenation, Indexing, and Slicing

Strings are immutable

- Immutable objects can’t be changed

Concatenation, Indexing, and Slicing

Strings are immutable

- Immutable objects can’t be changed

```
>>> word = "boil"
>>> word[0] = "f"
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    word[0] = "f"
TypeError: 'str' object does not support item assignment
```

Concatenation, Indexing, and Slicing

Strings are immutable

- To alter a string, you must create a new one
- This creates a new string object through assignment

Concatenation, Indexing, and Slicing

Strings are immutable

- To alter a string, you must create a new one
- This creates a new string object through assignment

```
>>> word = "boil"
>>> word = "f" + word[1:]
'foil'
```

Concatenation, Indexing, and Slicing

Review Exercise:

1. Create a string and print its length using len().
2. Create two strings, concatenate them, and print the resulting string.
3. Create two strings, use concatenation to add a space between them, and print the result.
4. Print the string “zing” by using slice notation to specify the correct range of characters in the string “bazinga”.

Table of Contents

Manipulate Strings With Methods

In this lesson you'll learn:

- Convert a string to uppercase or lowercase
- Remove whitespace from a string
- Determine if a string begins or ends with certain characters

Manipulate Strings With Methods

Converting String Case

`.lower()` - Converts a string to all lowercase letters

Manipulate Strings With Methods

Converting String Case

`.lower()` - Converts a string to all lowercase letters

```
>>> "Jean-Luc Picard".lower()
'jean-luc picard'
```

Manipulate Strings With Methods

Converting String Case

`.upper()` - Converts a string to all uppercase letters

Manipulate Strings With Methods

Converting String Case

`.upper()` - Converts a string to all uppercase letters

```
>>> "Jean-Luc Picard".upper()
'JEAN-LUC PICARD'
```

Manipulate Strings With Methods

Removing whitespace from a string

- `.rstrip()` - Removes trailing spaces from right side of string
- `.lstrip()` - Removes preceding spaces from left side of string
- `.strip()` - Removes whitespaces from both left and right sides

Manipulate Strings With Methods

Determining if a string starts or ends with a particular string

- `.startswith()`
- `.endswith()`

Manipulate Strings With Methods

Using IDLE to discover additional string methods

- Type the name of a variable that is assigned a string.
  - `>>> starship = "Enterprise"`
  - `>>> starship`
- Add a period (often spoken as “dot”) and wait
  - `>>> starship.`
- Typing in a letter will narrow the results
  - `>>> starship.u`
- Pressing the `Tab` key will automatically fill in the method name
  - `>>> starship.upper()`

Review What You've Learned

1. Write a program that converts the following strings to lowercase: "Animals", "Badger", "Honey Bee", "Honey Badger". Print each lower- case string on a separate line.
2. Repeat exercise 1, but convert each string to uppercase instead of lowercase.
3. Write a program that removes whitespace from the following strings, then print out the strings with the whitespace removed:

Review What You've Learned

1. Write a program that converts the following strings to lowercase: "Animals", "Badger", "Honey Bee", "Honey Badger". Print each lower- case string on a separate line.
2. Repeat exercise 1, but convert each string to uppercase instead of lowercase.
3. Write a program that removes whitespace from the following strings, then print out the strings with the whitespace removed:

```
string1 = " Filet Mignon"
string2 = "Brisket "
string3 = " Cheeseburger "
```

Review What You've Learned

4. Write a program that prints out the result of .startswith("be") on each of the following strings:

Review What You've Learned

4. Write a program that prints out the result of .startswith("be") on each of the following strings:

```
string1 = "Becomes"
string2 = "becomes"
string3 = "BEAR"
string4 = " bEautiful"
```

5. Using the same four strings from exercise 4, write a program that uses string methods to alter each string so that .startswith("be") returns True for all of them.

Table of Contents

Interact With User Input

Making things interactive with `input()`

Interact With User Input

Making things interactive with `input()`

```
prompt = "Hey, what's up? "
user_input = input(prompt)
print("You said: " + user_input)
```

Review What You've Learned

Write a few programs that take input from the user and:

1. Have the program display that input back.
2. Have the program display the input in lowercase.
3. Have the program display the number of characters in the input.

Table of Contents

Working With Strings and Numbers

Using Strings With Arithmetic Operators

- The `+` operator concatenates two strings together
- The `*` operator concatenates **multiple** copies of a string

Working With Strings and Numbers

Converting Strings to Numbers

- `int()` - converts objects into whole numbers
- `float()` - converts objects into numbers with decimal points

Working With Strings and Numbers

Converting Numbers to Strings

- `str()` - returns a string version of an object

Review What You've Learned

1. Create a string containing an integer, then convert that string into an actual integer object using `int()`.
    - Test that your new object is a number by multiplying it by another number and displaying the result.
2. Repeat the previous exercise, but use a floating-point number and `float()`.
3. Create a string object and an integer object, then display them side by side with a single print using `str()`.
4. Write a program that uses `input()` twice to get two numbers from the user, multiplies the numbers together, and displays the result.
    - `The product of 2 and 4 is 8.0`

Table of Contents

Streamline Your Print

Formatted String Literals - More commonly known as **f-strings**

Streamline Your Print

Formatted String Literals - More commonly known as **f-strings**

```
>>> day = "Tuesday"
>>> eggs = 2
>>> bacon = 3
>>> f"{day}'s breakfast is {eggs} eggs and {bacon} pieces of bacon"
"Tuesday's breakfast is 2 eggs and 3 pieces of bacon"
```

1. A string literal starts with the letter `f` before the opening quotation mark
2. Variable names surrounded by curly braces (`{}`) are replaced by their corresponding values without using `str()`
    - Python expressions between the braces are replaced with their result

Streamline Your Print

f-strings are available only in Python version 3.6 and above

In earlier versions of Python you could use `.format()`

```
>>> day = "Tuesday"
>>> eggs = 2
>>> bacon = 3
>>> "{}'s breakfast is {} eggs and {} pieces of bacon".format(day, eggs, bacon)
"Tuesday's breakfast is 2 eggs and 3 pieces of bacon"
```

Review What You've Learned

1. Create a float object named weight with the value `0.2`, and create a string object named animal with the value `"newt"`.
    - Then use these objects to print the following string using only string concatenation:
    - `0.2 kg is the weight of the newt`
2. Display the same string by using `.format()` and empty `{}` placeholders.
3. Display the same string using an f-string.

Table of Contents

Find a String in a String

- `.find()` - finds the location of one string within another string (a substring)
  - `.find(<sub>)`
  - Returns the index of the fist occurence of the string passed to it
  - Case-sensitive

Find a String in a String

- `.replace()` - replaces each instance of a substring with another string
  - `.replace(<old>, <new>)`

Review What You've Learned

1. In one line of code, display the result of trying to `.find()` the sub-string `"a"` in the string `"AAA"`.
    - The result should be `-1`.
2. Replace every occurrence of the character `"s"` with `"x"` in the string `"Somebody said something to Samantha"`.
3. Write a program that accepts user input with `input()` and displays the result of trying to `.find()` a particular letter in that input.

Challenge: Turn Your User Into a L33t H4x0r

- Write a program called `translate.py` that asks the user for some input with the following prompt: `Enter some text:`
- Use `.replace()` to convert the text entered by the user into leetspeak by making the following changes to lowercase letters:
  - The letter `a` becomes `4`
  - The letter `b` becomes `8`
  - The letter `e` becomes `3`
  - The letter `l` becomes `1`
  - The letter `o` becomes `0`
  - The letter `s` becomes `5`
  - The letter `t` becomes `7`

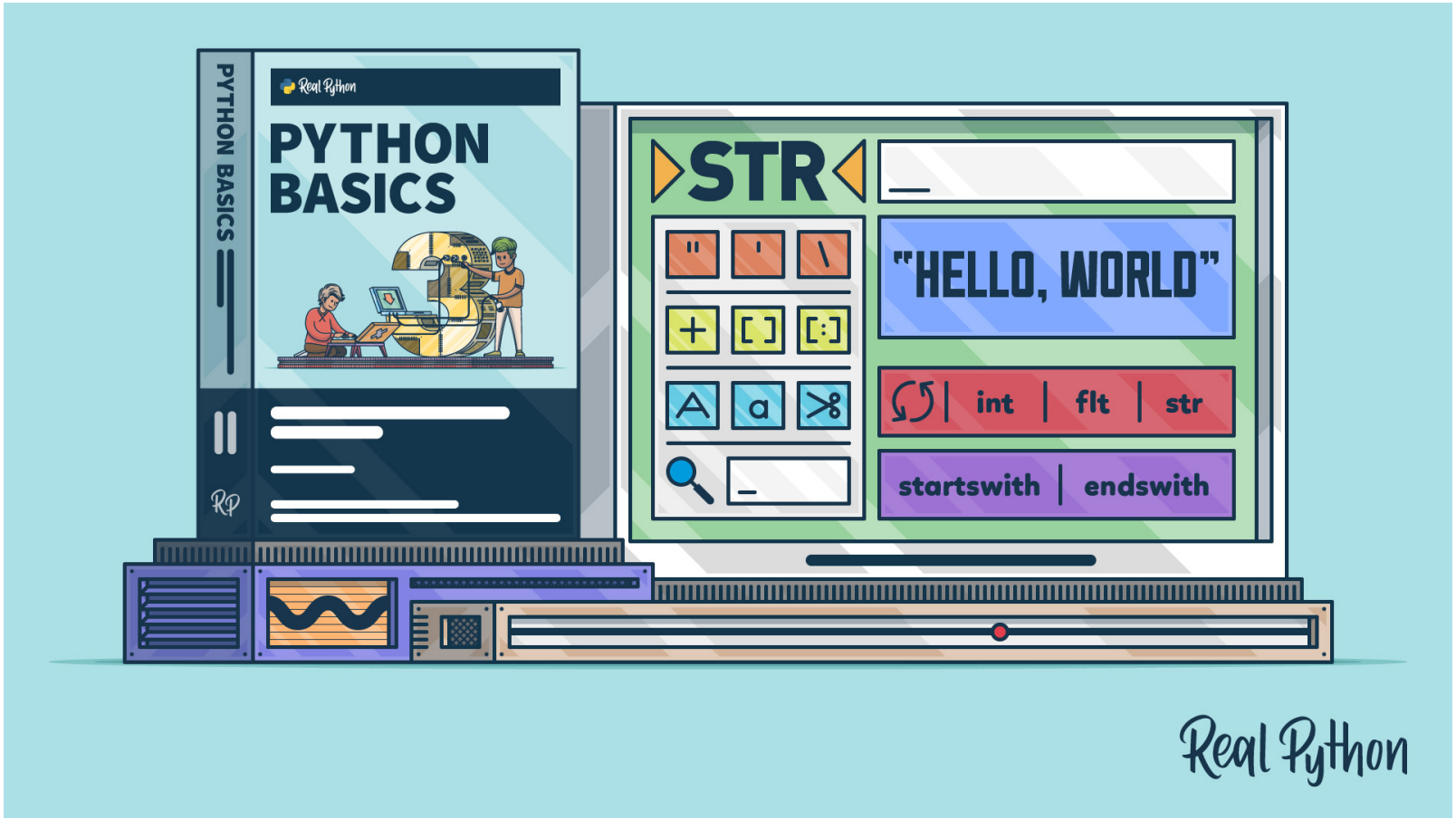
Challenge: Turn Your User Into a L33t H4x0r

Your program should then display the resulting string as output. Below is a sample run of the program:

```
Enter some text: I like to eat eggs and spam.
I 1ik3 70 347 3gg5 4nd 5p4m.
```

Table of Contents

Summary and Additional Resources



Summary and Additional Resources

In this course you learned how to:

- Manipulate strings with string methods
- Work with user input
- Deal with strings of numbers
- Format strings for printing

Summary and Additional Resources

Along the way you also learned about:

- f-Strings
- Multiline strings
- String indexing and slicing
- String immutablility
- Discovering additional methods

More Video Courses:

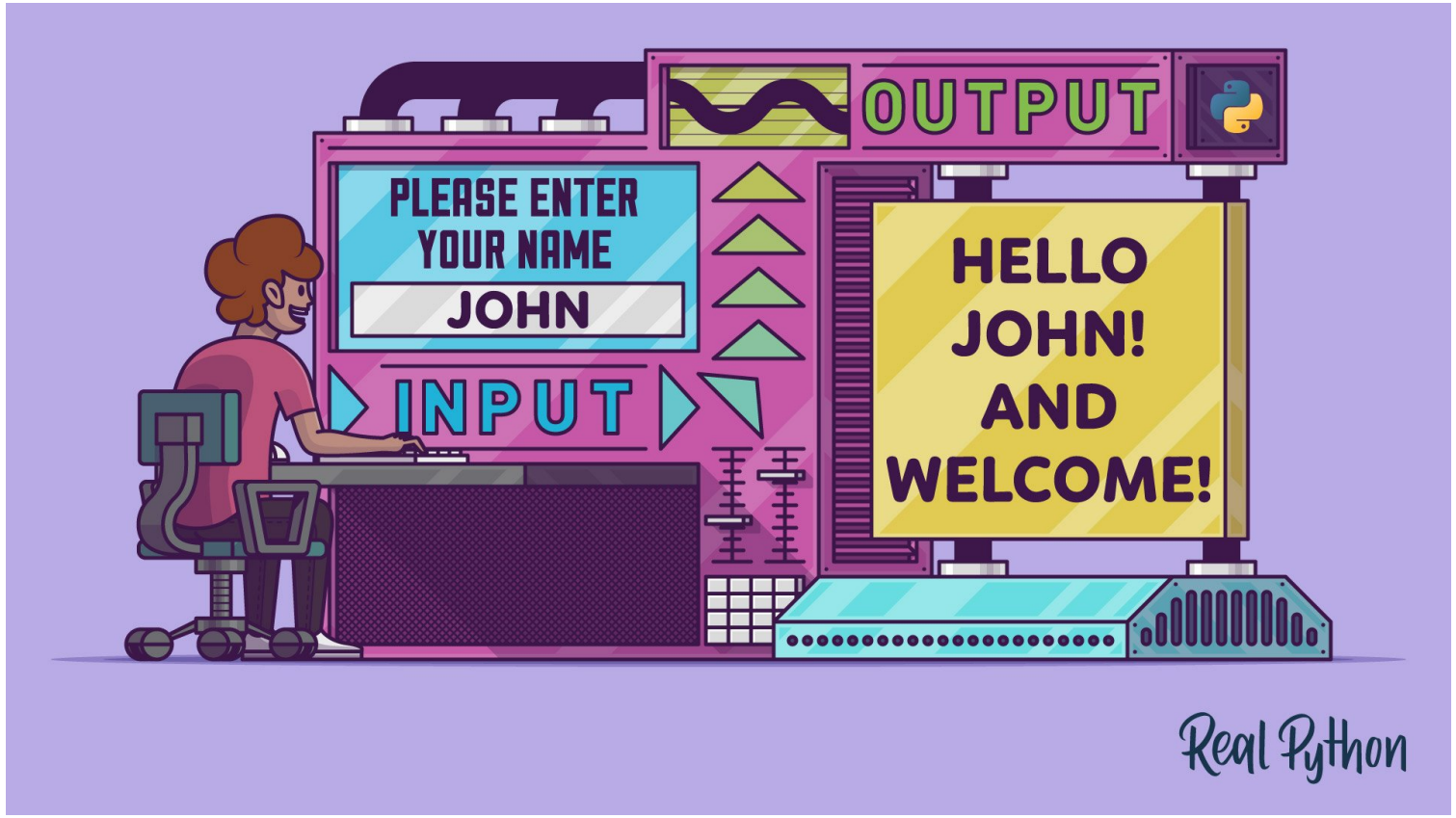
Python 3's f-Strings: An Improved String Formatting Syntax

More Video Courses:

Splitting, Concatenating, and Joining Strings in Python

More Video Courses:

Reading Input and Writing Output in Python



Congratulations and Thanks!