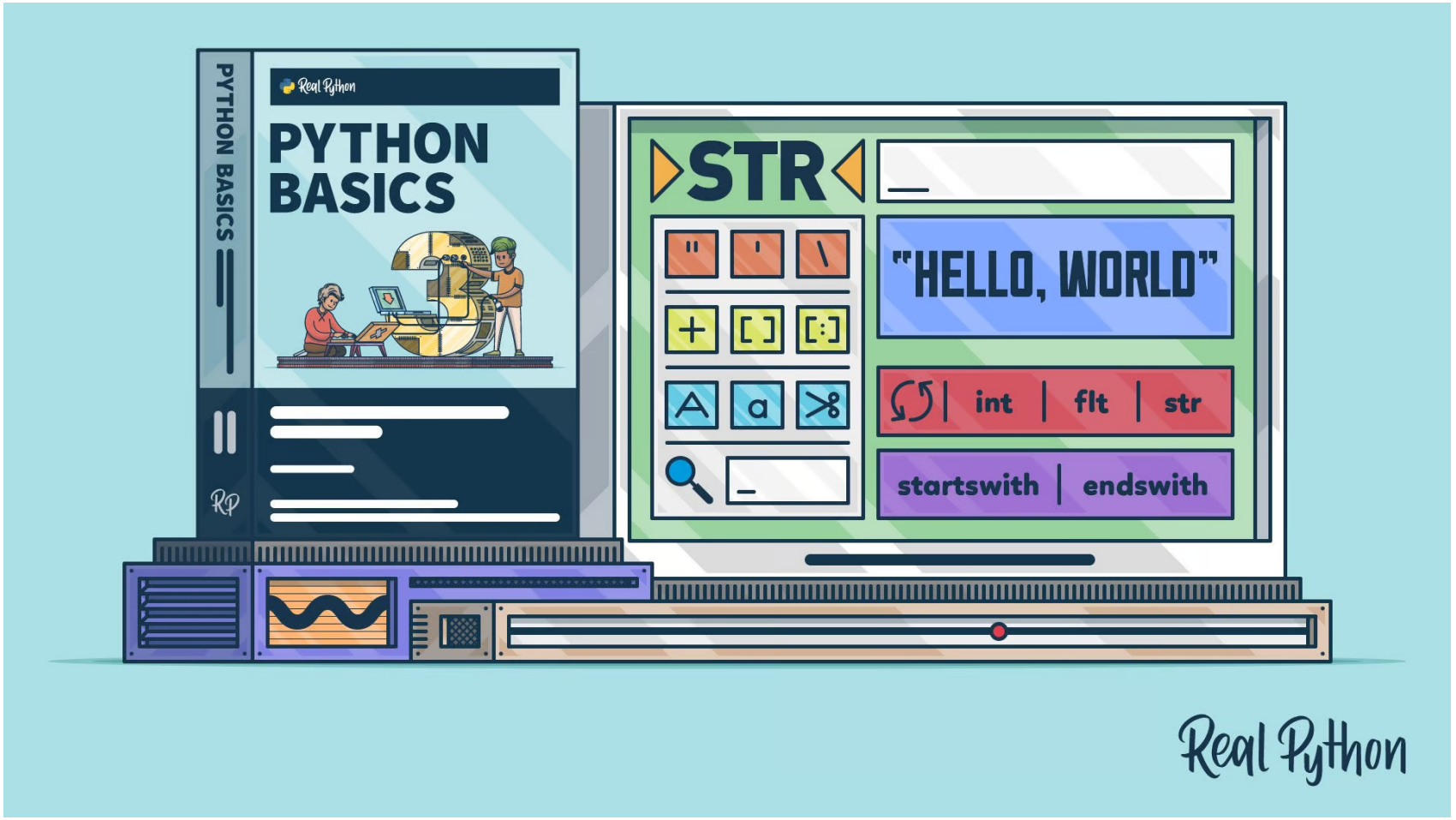
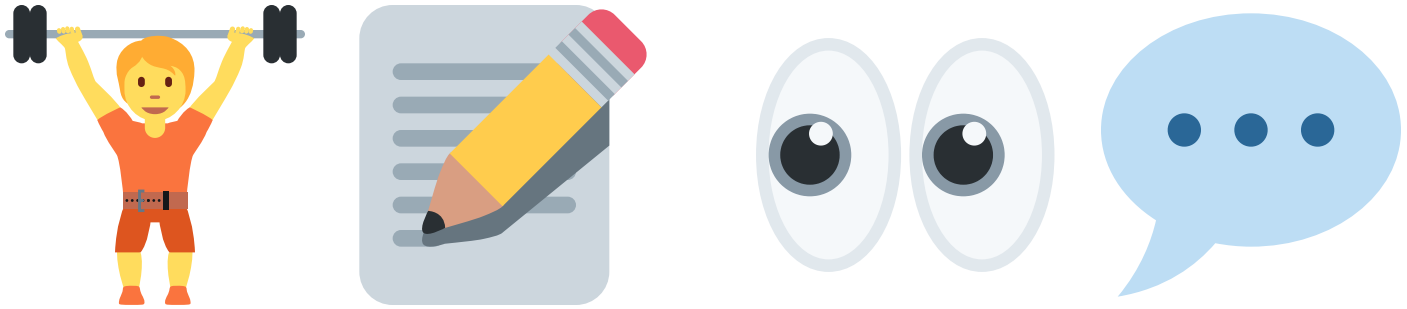
Python Basics Exercises: Strings and String Methods



Real Python Exercises Course



Real Python Exercises Course

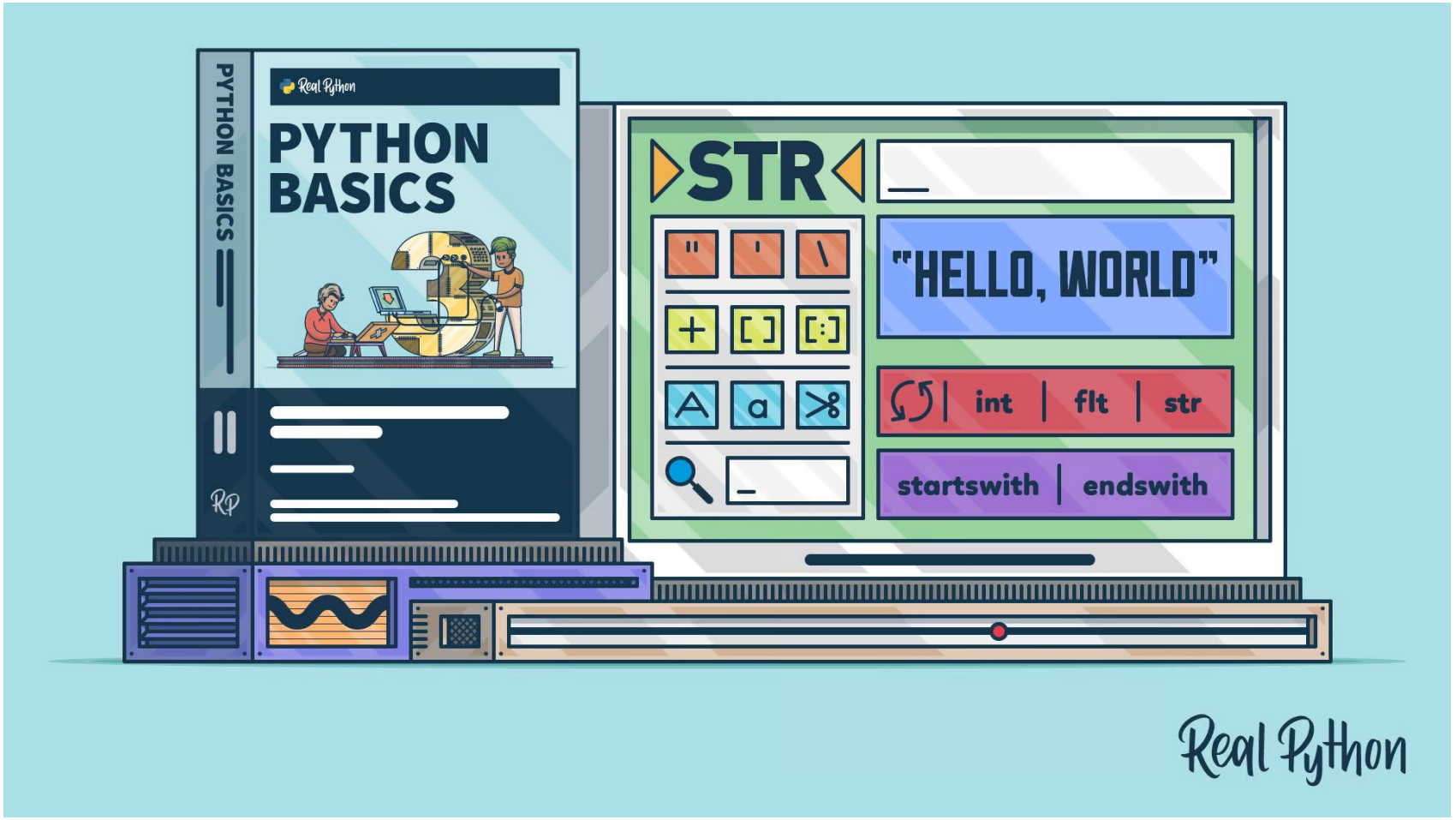
The three steps for each task:

1. 👀 Learn about the exercise
2. 👨‍💻 Code your solution
3. ↔️ Compare your solution

Python Basics Exercises: Strings and String Methods

1. Work With Strings
2. Modify Strings With String Methods
3. Collect User Input
4. Work With Strings and Numbers
5. Use More String Methods
6. **Challenge:** Leet Speak

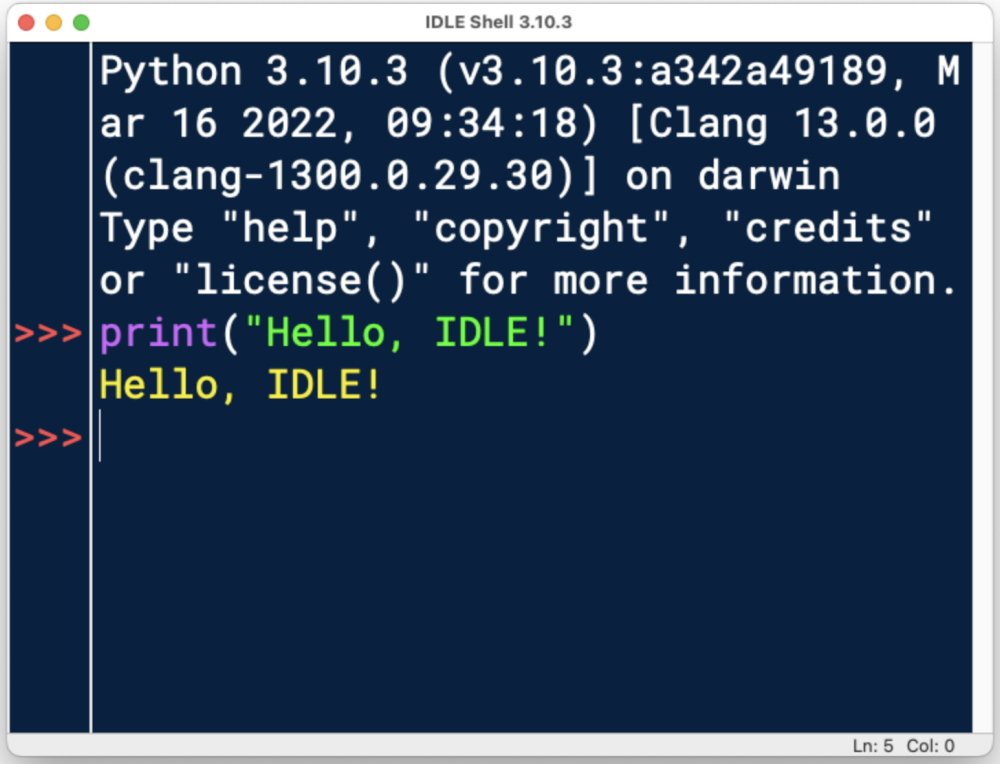
Background - Python Basics: Strings and String Methods



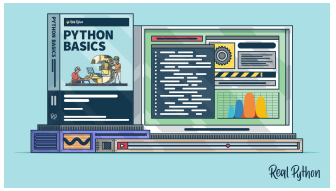
Background - Python Basics: Strings and String Methods

- Python Strings
- String methods
- User input collection
- Type conversion

Background - Using IDLE



Additional Resources



Python Basics: Setting Up Python

Getting Started With Python IDLE

Ready to Get Started?

Ready to Get Started?

```
f"YEAH!"
```

Ready to Get Started?

```
f"YEAH!"
```

🤔 Exercise: What Is a String?

Print four individual strings that fulfill certain conditions:

1. Double quotation marks inside the string
2. An apostrophe inside the string
3. Multiple lines with whitespace preserved
4. Coded on multiple lines but gets printed on a single line

💡 Solution: What Is a String?

Mixed Quotation Marks

```
# String 1
print('There are "double quotes" in this string.')

# String 2
print("This string's got an apostrophe.")
```

Multi-Line Strings

```
# String 3
print(
    """This string was written on multiple lines,
      and it displays across multiple lines"""
)

# String 4
print(
    "This one-line string was written out \
      using multiple lines"
)
```

🤔 Exercise: Concatenation

Create two strings, concatenate them, and print the resulting string. Can you add a space between them using concatenation?

💡 Solution: Concatenation

```
string_left = "bat"
string_right = "man"

print(string_left + string_right)  # No space

print(string_left + " " + string_right)  # Space
```

🤔 Exercise: Slicing

Print the string `"zing"` by using slice notation to specify the correct range of characters in the string `"bazinga"`.

💡 Solution: Slicing

```
print("bazinga"[2:6])
print("bazinga"[2:-1])
```

🤔 Exercise: Change of Case

1. Write a program that converts the following strings to lowercase:
   - `"Animals"`
   - `"Badger"`
   - `"Honey Bee"`
   - `"Honey Badger"`
2. Now convert each of these strings to uppercase instead of lowercase.

Note: Print each string on a separate line.

💡 Solution: Change of Case

```
# Exercise 1
string1 = "Animals"
string2 = "Badger"
string3 = "Honey Bee"
string4 = "Honeybadger"

print(string1.lower())
print(string2.lower())
print(string3.lower())
print(string4.lower())
```

```
# Exercise 2
print(string1.upper())
print(string2.upper())
print(string3.upper())
print(string4.upper())
```

🤔 Exercise: Remove Whitespace

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   "
```

💡 Solution: Remove Whitespace

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

print(string1.strip())  # Could also use .lstrip()
print(string2.strip())  # Could also use .rstrip()
print(string3.strip())
```

🤔 Exercise: Check the Beginning of a String

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"
```

💡 Solution: Check the Beginning of a String

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

print(string1.startswith("be"))  # False
print(string2.startswith("be"))  # True
print(string3.startswith("be"))  # False
print(string4.startswith("be"))  # False
```

🤔 Exercise: Apply String Methods

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

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

💡 Solution: Apply String Methods

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

string1 = string1.lower()
# string2 will pass unmodified
string3 = string3.lower()
string4 = string4.strip().lower()

print(string1.startswith("be"))
print(string2.startswith("be"))
print(string3.startswith("be"))
print(string4.startswith("be"))
```

🤔 Exercise: Interact With User Input

Write a program that takes input from the user and displays that input back after converting it to lowercase.

Can you also display the number of characters in the input?

💡 Solution: Interact With User Input

```
input_string = input("Type something: ")

print(input_string.lower())
print(len(input_string))
```

🤔 Exercise: Pick Apart Your User's Input

Write a program named `first_letter.py` that prompts the user for input with the string `"Tell me your password:"`. The program should then determine the first letter of the user's input, convert that letter to uppercase, and display it back.

For example, if the user input is `"no"`, then the program should display the following output:

```
The first letter you entered was: N
```

> **Note:** For now, it's okay if your program crashes when the user enters nothing as input—that is, when they just hit *Enter* instead of typing something.

💡 Solution: Pick Apart Your User's Input

```
user_input = input("Tell me your password: ")

first_letter = user_input[0]

print("The first letter you entered was: " + first_letter.upper())
```

🤔 Exercise: Working With Strings and Numbers

Write a program that uses `input()` twice to get two numbers from the user, multiplies the numbers together, and displays the result. If the user enters `2` and `4`, then your program should print the following text:

```
The product of 2 and 4 is 8.0.
```

Keep in mind that:

1. `input()` always creates a string.
2. You can convert certain strings into actual number objects using `int()` and `float()`.
3. You can convert numbers to strings using `str()`.

💡 Solution: Working With Strings and Numbers (1/2)

```
# Convert 'string numbers' to actual numbers
my_integer_string = "6"
print(int(my_integer_string) * 7)

my_float_string = "6.01"
print(float(my_float_string) * 7)



# Display strings and integers together
my_string = "mp"
my_int = 3
print(my_string + str(my_int))
```

💡 Solution: Working With Strings and Numbers (2/2)

```
# Get two numbers from the user, multiply them,
# and display the result

a = input("Enter a number: ")
b = input("Enter another number: ")

product = float(a) * float(b)

print("The product of " + a + " and " + b + " is " + str(product) + ".")
```

🤔 Exercise: Streamline Your Prints

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.

💡 Solution: Streamline Your Prints

```
weight = 0.2
animal = "newt"

# Concatenation
print(str(weight) + " kg is the weight of the " + animal + ".")

# str.format()
print("{} kg is the weight of the {}.".format(weight, animal))

# F-string
print(f"{weight} kg is the weight of the {animal}.")
```

🤔 Exercise: Proof That ‘a’ Is Not in ‘AAA’

In one line of code, display the result of trying to find the substring `"a"` in the string `"AAA"` using `.find()`.

The result should be `-1`.

💡 Solution: Proof That ‘a’ Is Not in ‘AAA’

```
print("AAA".find("a"))
```

🤔 Exercise: Replace a Character

Replace every occurrence of the character `"s"` with `"x"` in the following string:

```
"Somebody said something to Samantha."
```

💡 Solution: Replace a Character

```
phrase = "Somebody said something to Samantha."
phrase = phrase.replace("s", "x")
print(phrase)
```

Note: This code will only replace the lowercase `"s"`, resulting in:

```
"Somebody xaid xomething to Samantha."
```

This is intentional to remind you that uppercase and lowercase characters are different characters.

🤔 Exercise: Find a Letter in a String

Write a program that accepts user input with `input()` and displays the result of trying to find a particular letter in that input using `.find()`.

💡 Solution: Find a Letter in a String

```
my_input = input("Type something: ")

letter = input("Which letter to search for: ")

print(my_input.find(letter))
```

🤔 Challenge: Turn Your User Into a L33t H4ck3r (1/3)

Write a program called `translate.py` that asks the user for some input with the following prompt:

```
Enter some text:
```

🤔 Challenge: Turn Your User Into a L33t H4ck3r (2/3)

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 H4ck3r (3/3)

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.
```

💡 Solution: Turn Your User Into a L33t H4ck3r

```
my_text = input("Enter some text: ")

my_text = (
    my_text.replace("a", "4")
    .replace("b", "8")
    .replace("e", "3")
    .replace("l", "1")
    .replace("o", "0")
    .replace("s", "5")
    .replace("t", "7")
)

print(my_text)
```

Summary and Additional Resources

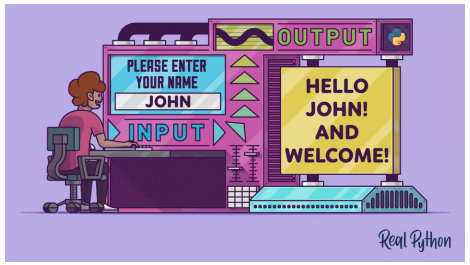
In this course, you **practiced** using:

- Strings
- Different quotation marks
- Multi-line strings
- String indices
- String methods
- User input collection
- Type conversion
- Method chaining

Tips

- 💬 Use code comments to help you get organized
- 🧩 Break exercises into smaller tasks
- ✅ Test repeatedly to see whether the code does what you expect it to do

Additional Resources

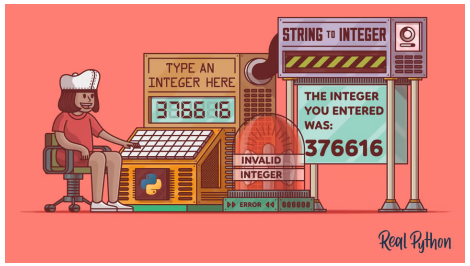


Basic Input, Output, and String Formatting in Python

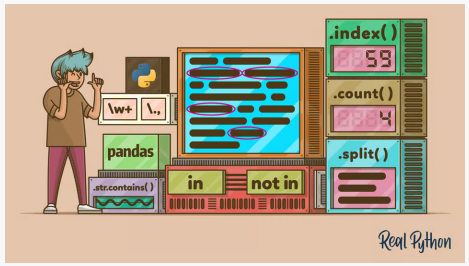
Python's F-String for String Interpolation and Formatting

Splitting, Concatenating, and Joining Strings in Python

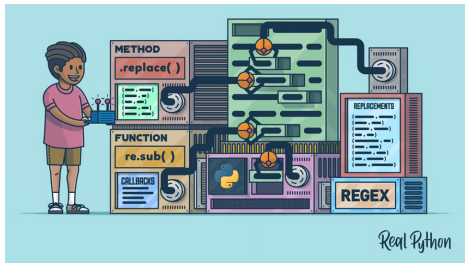
Additional Resources



How to Read Python Input as Integers



How to Check if a Python String Contains a Substring



How to Replace a String in Python

Congratulations and F"anks"!