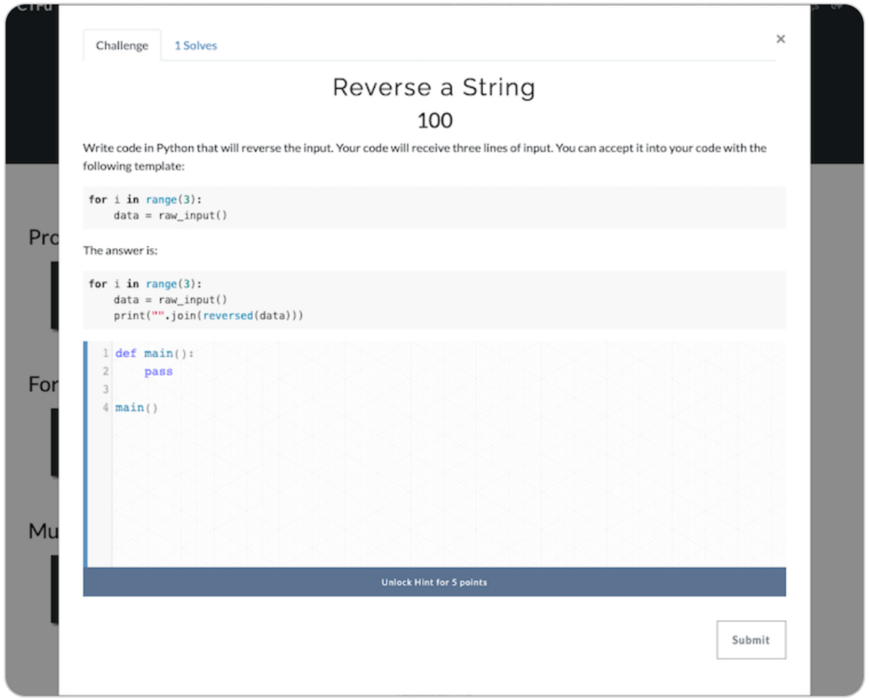
Code Challenges

CAUTION

Code Challenges are only available on Hosted CTFd instances or Enterprise CTFd Instances

Code Challenges allow CTFd to grade and evaluate code submissions in a variety of programming languages. Code submissions are provided a `stdin` value which the submission must then process to get a `stdout` value.



Reverse a String

100

Write code in Python that will reverse the input. Your code will receive three lines of input. You can accept it into your code with the following template:

```
for i in range(3):
    data = raw_input()
```

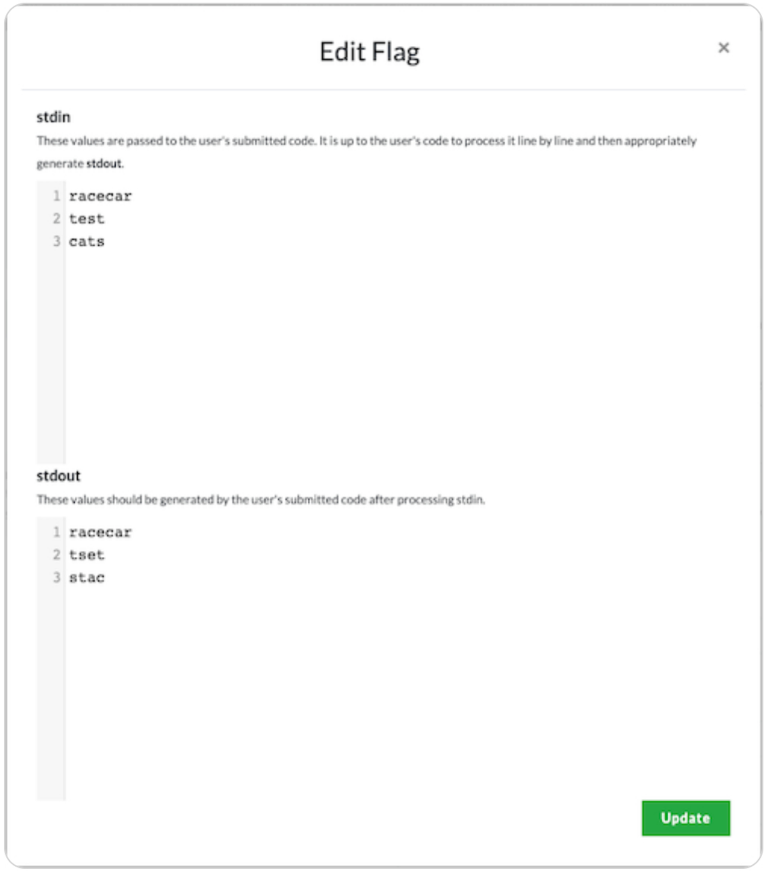
The answer is:

```
for i in range(3):
    data = raw_input()
    print("".join(reversed(data)))
```

```
def main():
    pass

main()
```

Grading is done by creating a code flag alongside the code challenge. Code flags allow you to specify the `stdin` and `stdout` that the user's code should receive and generate.


Edit Flag

stdin

These values are passed to the user's submitted code. It is up to the user's code to process it line by line and then appropriately generate **stdout**.

```
racecar
test
cats
```

stdout

These values should be generated by the user's submitted code after processing stdin.

```
racecar
tset
stac
```

INFO

CTFd will always normalize line endings to Linux line endings (i.e. `\n`) in `stdin` and `stdout` when processing or grading.

Test Cases

Each code flag for a code challenge is considered a seperate test case. The submission must pass all test cases (in this case a flag), in order for the challenge to be marked correct.

Reading input

A user's submissions should read data from `stdin` and then process it somehow to generate the expected `stdout`. Here are the different supported ways to read `stdin`.

Directly

Users can read directly from stdin in their chosen programming language. This may have difficulties with detecting when stdin is empty depending on the challenge.

For example with Python code:

```python
while True:
    try :
        line = input()
        # Process line
        print(line)
    except EOFError:
        # Break out if stdin is empty
        break
```

From a file

Users can read from the special `input.txt` file which contains the stdin value represented as a file.

For example with Python code:

```python
with open("input.txt") as f:
    lines = f.read().splitlines()
    for line in lines:
        # Process line
        print(line)
```

Security Concerns

Leaking stdin

Code challenges support the ability to run the user's submission and display the output to the user.

However keep in mind that this may leak out the input test cases which may allow the user to cheat their way past the challenge.

Network Access

Code challenges by design do not support network access. If you have a specific use case for this reach out to support@ctfd.io to discuss.

Supported Languages

We currently support the following programming languages. However if you need a programming language reach out to support@ctfd.io and we can likely add support!

Python

- 2.7
- 2.5
- 3.6
- 3.8

C/C++

- C++14

Java

- 11

NodeJS

- 8.x

Ruby

- 2.5

Bash

- 4.4

Golang

- 1.18

Rust

- 1.62.1

Previous
« Manual Verification

Next
King of the Hill »

Was this page helpful? 👍 👎

Share your feedback