

# CS50's Introduction to Programming with Python


OpenCourseWare

Donate  (<https://cs50.harvard.edu/donate>)

David J. Malan (<https://cs.harvard.edu/malan/>)

[malan@harvard.edu](mailto:malan@harvard.edu)

 (<https://www.facebook.com/dmalan>)  (<https://github.com/dmalan>) 

(<https://www.instagram.com/davidjmalan/>)  (<https://www.linkedin.com/in/malan/>)

 (<https://www.reddit.com/user/davidjmalan/>) 

(<https://www.threads.net/@davidjmalan>)  (<https://twitter.com/davidjmalan>)

## Bitcoin Price Index

Bitcoin (<https://en.wikipedia.org/wiki/Bitcoin>) is a form of digital currency, otherwise known as [cryptocurrency](https://en.wikipedia.org/wiki/Cryptocurrency) (<https://en.wikipedia.org/wiki/Cryptocurrency>). Rather than rely on a central authority like a bank, Bitcoin instead relies on a distributed network, otherwise known as a [blockchain](https://en.wikipedia.org/wiki/Blockchain) (<https://en.wikipedia.org/wiki/Blockchain>), to record transactions.



Because there's demand for Bitcoin (i.e., users want it), users are willing to buy it, as by exchanging one currency (e.g., USD) for Bitcoin.

In a file called `bitcoin.py`, implement a program that:

- Expects the user to specify as a command-line argument the number of Bitcoins,  $n$ , that they would like to buy. If that argument cannot be converted to a `float`, the program should exit via `sys.exit` with an error message.
- Queries the API for the CoinDesk Bitcoin Price Index at <https://api.coindesk.com/v1/bpi/currentprice.json> (<https://api.coindesk.com/v1/bpi/currentprice.json>), which returns a [JSON](https://en.wikipedia.org/wiki/JSON) (<https://en.wikipedia.org/wiki/JSON>) object, among whose nested keys is the current price of Bitcoin as a `float`. Be sure to catch any [exceptions](https://requests.readthedocs.io/en/latest/api/#exceptions) (<https://requests.readthedocs.io/en/latest/api/#exceptions>), as with code like:

```
import requests

try:
    ...
except requests.RequestException:
    ...
```

- Outputs the current cost of  $n$  Bitcoins in USD to four decimal places, using `,` as a thousands separator.

## ▼ Hints

- Recall that the `sys` module comes with `argv`, per [docs.python.org/3/library/sys.html#sys.argv](https://docs.python.org/3/library/sys.html#sys.argv) (<https://docs.python.org/3/library/sys.html#sys.argv>).
- Note that the `requests` module comes with quite a few methods, per [requests.readthedocs.io/en/latest](https://requests.readthedocs.io/en/latest/) (<https://requests.readthedocs.io/en/latest/>), among which are `get`, per [requests.readthedocs.io/en/latest/user/quickstart/#make-a-request](https://requests.readthedocs.io/en/latest/user/quickstart/#make-a-request) (<https://requests.readthedocs.io/en/latest/user/quickstart/#make-a-request>), and `json`, per [requests.readthedocs.io/en/latest/user/quickstart/#json-response-content](https://requests.readthedocs.io/en/latest/user/quickstart/#json-response-content) (<https://requests.readthedocs.io/en/latest/user/quickstart/#json-response-content>). You can install it with:

```
pip install requests
```

- Note that CoinDesk's API returns a JSON response like:

```
{
  "time":{
    "updated":"May 2, 2022 15:27:00 UTC",
    "updatedISO":"2022-05-02T15:27:00+00:00",
    "updateduk":"May 2, 2022 at 16:27 BST"
  },
  "disclaimer":"This data was produced from the CoinDesk Bitcoin Price Index",
  "chartName":"Bitcoin",
  "bpi":{
    "USD":{
      "code":"USD",
      "symbol":"$",
      "rate":"38,761.0833",
      "description":"United States Dollar",
      "rate_float":38761.0833
    },
    "GBP":{
      "code":"GBP",
      "symbol":"£",
      "rate":"30,827.6198",
      "description":"British Pound Sterling",
      "rate_float":30827.6198
    },
    "EUR":{
      "code":"EUR",
      "symbol":"€",
      "rate":"36,800.2764",
```

```
        "description": "Euro",
        "rate_float": 36800.2764
    }
}
```

- Recall that you can format USD to four decimal places with a [thousands separator](https://docs.python.org/3/library/string.html#formatspec) (<https://docs.python.org/3/library/string.html#formatspec>) with code like:

```
print(f"${amount:,.4f}")
```

## Demo

---

This demo was recorded when the price of Bitcoin was \$38,761.0833. Your own output may vary.

## Before You Begin

---

Log into [cs50.dev](https://cs50.dev) (<https://cs50.dev/>), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
mkdir bitcoin
```

to make a folder called `bitcoin` in your codespace.

Then execute

```
cd bitcoin
```

to change directories into that folder. You should now see your terminal prompt as `bitcoin/ $`. You can now execute

```
code bitcoin.py
```

to make a file called `bitcoin.py` where you'll write your program.

## How to Test

---

Here's how to test your code manually:

- Run your program with `python bitcoin.py`. Your program should use `sys.exit` to exit with an error message:

```
Missing command-line argument
```

- Run your program with `python bitcoin.py cat`. Your program should use `sys.exit` to exit with an error message:

```
Command-line argument is not a number
```

- Run your program with `python bitcoin.py 1`. Your program should output the price of a single Bitcoin to four decimal places, using `,` as a [thousands separator](https://docs.python.org/3/library/string.html#formatspec) (<https://docs.python.org/3/library/string.html#formatspec>).
- Run your program with `python bitcoin.py 2`. Your program should output the price of two Bitcoin to four decimal places, using `,` as a [thousands separator](https://docs.python.org/3/library/string.html#formatspec) (<https://docs.python.org/3/library/string.html#formatspec>).
- Run your program with `python bitcoin.py 2.5`. Your program should output the price of 2.5 Bitcoin to four decimal places, using `,` as a [thousands separator](https://docs.python.org/3/library/string.html#formatspec) (<https://docs.python.org/3/library/string.html#formatspec>).

You can execute the below to check your code using `check50`, a program that CS50 will use to test your code when you submit. But be sure to test it yourself as well!

```
check50 cs50/problems/2022/python/bitcoin
```

Green smilies mean your program has passed a test! Red frownies will indicate your program output something unexpected. Visit the URL that `check50` outputs to see the input `check50` handed to your program, what output it expected, and what output your program actually gave.

## How to Submit

---

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2022/python/bitcoin
```

