Objectives

- Continuing with dictionaries
- Exceptions
- Broader Issue: Cryptocurrency

1

Lab Comment: Pair Programming

- Every choice has tradeoffs
- The discussions and your use of vocabulary improved!
- Slowed down your lines/minute
- Better understanding and, hopefully (if not overconfident), less buggy!

2

Review: Dictionaries

- What is a dictionary in Python?
- What is the syntax for creating a new dictionary?
- How do we access a key's value from a dictionary? (2 ways)
  - What happens if there is no mapping for that key?
- How do we create a key → value mapping in a dictionary?
- How can we iterate through a dictionary?

Review: Creating Dictionaries in Python

Syntax:

```
{<key>:<value>, …,
 <key>:<value>}
```

```
empty = {}
charToAscii = { 'a':97, 'b':98, …, 'z':122 }
```

Review: Dictionary Operations

Indexing	`<dict>[<key>]`
Length (# of keys)	`len(<dict>)`
Iteration	`for <key> in <dict>:`
Membership	`<key> in <dict>`
Deletion	`del <dict>[<key>]`

Unlike strings and lists, doesn't make sense to do slicing, concatenation, repetition for dictionaries

5

Review: Dictionary Methods

Method Name	Functionality
`<dict>.clear()`	Remove all items from dictionary
`<dict>.keys()`	Returns a copy of dictionary's keys (a set-like object)
`<dict>.values()`	Returns a copy of dictionary's values (a set-like object)
`<dict>.get(x [, default])`	Returns `<dict>[x]` if x is a key; Otherwise, returns None (or default value)

6

Review: Accessing Values Using Indexing

- Syntax:

  `<dictionary>[<key>]`

- Examples:

```
charToAscii['z']

nameToPhoneNum['friendname']
```

- **KeyError** if key is not in dictionary
  - Runtime error; exits program

Review: Adding/Modifying Key-Value Pairs

- Syntax:

  `<dictionary>[<key>] = <value>`

- Example:

```
nameToPhoneNum['registrar'] = 8455
```

  - Adds mapping for `'registrar'` to 8455

  OR

  - If mapping already existed, *modifies* old mapping to 8455

Review: Problem

`years_dictionary.py`

- Part 1:
  - Given a file of the form
    - <firstname> <gradyear>
  - Goal: Quickly find out what a student's grad year is
    - How do we want to model the data?
    - What is the key? What is the value?
- Part 2:
  - Repeatedly prompt user for the first name of the student
  - Display the student's graduation year

9

Review: Algorithm to Problem

- Create an empty dictionary
- Read in the file line by line
  - Split the line
  - From the split, get the name and the year
  - Add a mapping of the name to the year in the dictionary
    - (*accumulate* the data in the dictionary)
- Process the data in the dictionary, e.g.,
  - Display it, in sorted order
  - Get user input to get answers

10

Problem

Example file:

```
Person1 2023
Person2 2022
Person3 2023
Person4 2024
Person5 2023
…
```

- Given a file of the form
  - <firstname> <classyear>
- Goal: Report the *number* of students in each graduation year
  - How do we want to model the data?
  - What is the key? What is the value?
- Problem-solving Approach:
  - Pretend you are the computer, how would you solve this problem?

Equivalent Solutions: A Dictionary of Accumulators

```
if key not in dictionary :
   dictionary[key] = 1
else:
   count = dictionary[key] + 1
   dictionary[key] = count
```

```
if key not in dictionary :
   dictionary[key] = 1
else:
   dictionary[key] += 1
```

Broader Issue: Cryptocurrencies

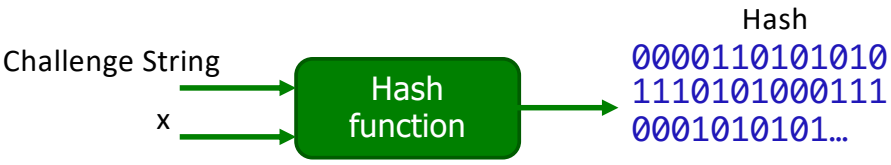
- What are cryptocurrencies?
- With your newly attained computer science knowledge, what ideas make more sense?
  - What is still confusing?
- What questions about cryptocurrencies do you still have?

BitCoin: Proof of Work

Hash function

- ***Hash*** functions map from one "thing" to a string of fixed length
  - Output may look random but it's not
  - Given the same input, always results in the same output
  - Unlikely that other inputs will result in same output
- Example: URL → TinyURL
  - https://mybig.url/thatisreallylong/but/helpful → https://tinyurl.com/4f3c38wp
  - Recreating the URL will result in the same URL

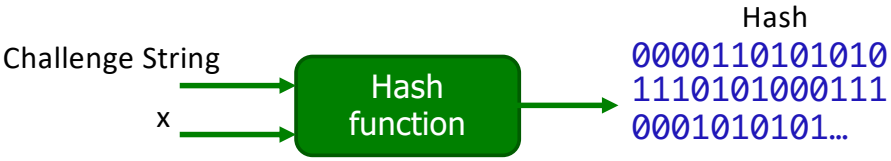
BitCoin: Mathematical Puzzle



- Problem: given a *challenge string*, find *x* such that the result of hashing the challenge string and x gives a result with certain properties
  - x is called the *proof string*

15

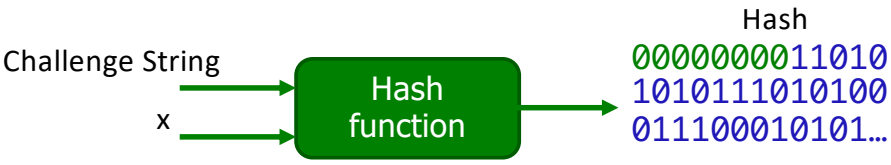
BitCoin: Mathematical Puzzle



- Problem: given a *challenge string*, find *x* such that the result of hashing the challenge string and x gives a result with certain properties
  - x is called the *proof string*
- Example property: resulting hash must start with at least 8 zeros
  - This resulting hash does not meet criteria

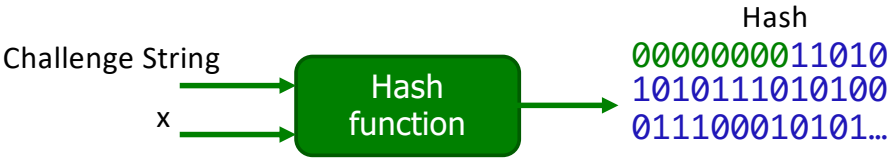
16

BitCoin: Mathematical Puzzle



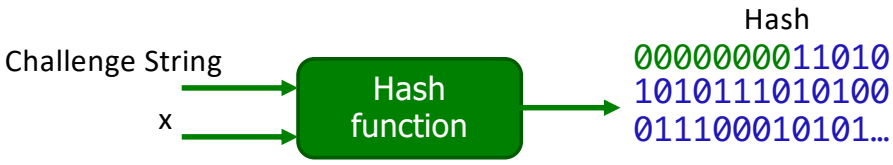
- Difficult to find the x that results in the desired property
  - Have to try lots of different x's
- But, easy to validate!
  - If someone says *x* results in a hash with desired properties, we can run the hash function and verify if the resulting hash has the desired properties

BitCoin: Mathematical Puzzle



- As computational power increases, it takes less time to find an x.
- How can we [easily] make the problem harder?

BitCoin: Proof of Work



- How does this relate to *proof of work*?
  - We know, on average/probabilistically, how many tries it would take to get a hash that satisfies the requirements

Public Key Cryptography