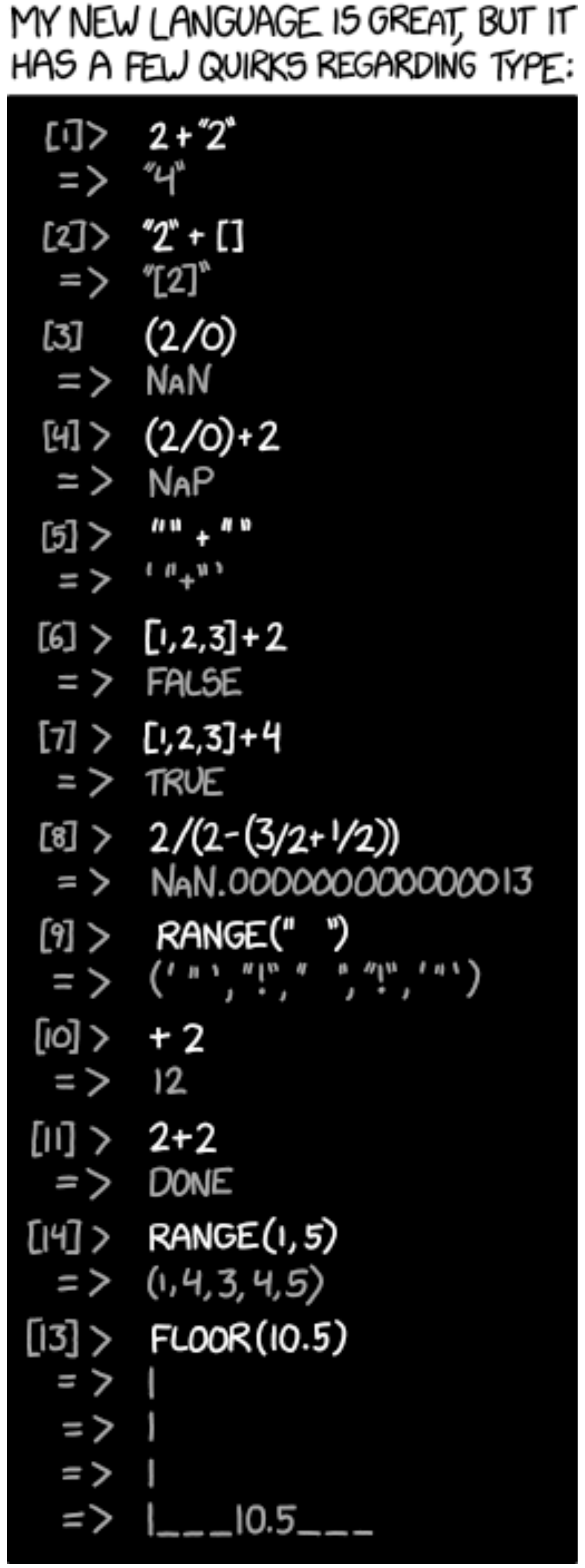
CSCI 141

Lecture 3
Introduction to Data:
Types, Values, Function Calls,
Variables


MY NEW LANGUAGE IS GREAT, BUT IT HAS A FEW QUIRKS REGARDING TYPE:

```
[1]>  2+"2"
 =>   "4"
[2]>  "2" + []
 =>   "[2]"
[3]   (2/0)
 =>   NaN
[4]>  (2/0)+2
 =>   NaP
[5]>  "" + ""
 =>   '"+"'
[6]>  [1,2,3]+2
 =>   FALSE
[7]>  [1,2,3]+4
 =>   TRUE
[8]>  2/(2-(3/2+1/2))
 =>   NaN.0000000000000013
[9]>  RANGE(" ")
 =>   ('"', "!", " ", "!", '"')
[10]> +2
 =>   12
[11]> 2+2
 =>   DONE
[14]> RANGE(1,5)
 =>   (1,4,3,4,5)
[13]> FLOOR(10.5)
 =>   |
 =>   |
 =>   |
 =>   |___10.5___
```

Happenings

- CS Resume Workshop 5-6pm on Wednesday, October 2nd CF 115
  Presented by Filip Jagodzinski, students questions about writing resumes and cover letters will be answered while enjoying… PIZZA!

- Tech Talk: Google on Monday, October 7th 5-6pm in CF 115
  Googlers share a day-in-the-life as a software engineer
  Resume review with Google on Monday, 10/7 during the day—sign up here

- **Accenture** on Tuesday October 8th 4-6pm CF 110
  Resume prep, with interviews to follow October 30th-November 1st

- Tech Talk: Microsoft on Wednesday, October 9th 5-6:30pm in CF 115
  Powershell: From Windows to the Cross-Platform Cloud

Announcements

Announcements

- Assignment 1 will be released later today

Announcements

- Assignment 1 will be released later today
  - 3 small programming problems; Due next Monday night

Announcements

- Assignment 1 will be released later today
  - 3 small programming problems; Due next Monday night
- Everything you need to complete A1 will be covered by Wednesday.

Announcements

- Assignment 1 will be released later today
    - 3 small programming problems; Due next Monday night
- Everything you need to complete A1 will be covered by Wednesday.
- You can work on it in the labs (details on the syllabus) or on your own computer.

Announcements

- Assignment 1 will be released later today
    - 3 small programming problems; Due next Monday night
- Everything you need to complete A1 will be covered by Wednesday.
- You can work on it in the labs (details on the syllabus) or on your own computer.
    - Reminder: You can download Thonny from thonny.org.

Announcements

- Assignment 1 will be released later today
    - 3 small programming problems; Due next Monday night
- Everything you need to complete A1 will be covered by Wednesday.
- You can work on it in the labs (details on the syllabus) or on your own computer.
    - Reminder: You can download Thonny from thonny.org.
- Please keep track of the hours you spend

Socrative

Please log in at the beginning of class so you're ready when poll questions come up.

Reminder:

- socrative.com (or get the app)
- Room: 9AM141
- Student ID: Your WWU username.

QOTD

- You are given 3 "slip days" that allow you to submit something 24 hours late without penalty.
  T/F: These can be used for labs, assignments, or QOTDs.

QOTD

- You are given 3 "slip days" that allow you to submit something 24 hours late without penalty.
  T/F: These can be used for labs, assignments, or QOTDs.

✗ **False.**

QOTD

- You are given 3 "slip days" that allow you to submit something 24 hours late without penalty.
  T/F: These can be used for labs, assignments, or QOTDs.

✗ **False.**

Slip days are only usable on programming assignments.

QOTD

- You are given 3 "slip days" that allow you to submit something 24 hours late without penalty.
  T/F: These can be used for labs, assignments, or QOTDs.

✗ **False.**

Slip days are only usable on programming assignments.

Special circumstances for missing lab or submitting late? Email me.

QOTD

- **T/F:** All programming assignments are expected to take approximately the same amount of time to complete.

QOTD

- **T/F:** All programming assignments are expected to take approximately the same amount of time to complete.

✗ **False.**

QOTD

Where are lecture slides posted after lecture?

A. Socrative

B. Gradescope

C. Canvas

D. The course webpage

QOTD

Where are lecture slides posted after lecture?

A. Socrative

B. Gradescope

C. Canvas

[✓ D. The course webpage]

QOTD

According to the academic honesty policy, which of the following are permitted?

A. Talking about your code with your classmates.

B. Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

C. Submitting someone else's program as your own.

D. Copying a few lines of someone else's code into your solution, if you understand those lines in detail.

QOTD

According to the academic honesty policy, which of the following are permitted?

[✓ A.] Talking about your code with your classmates.

B. Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

C. Submitting someone else's program as your own.

D. Copying a few lines of someone else's code into your solution, if you understand those lines in detail.

QOTD

According to the academic honesty policy, which of the following are permitted?

✓ A. Talking about your code with your classmates.

✗ B. Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

C. Submitting someone else's program as your own.

D. Copying a few lines of someone else's code into your solution, if you understand those lines in detail.

QOTD

According to the academic honesty policy, which of the following are permitted?

✓ A. Talking about your code with your classmates.

✗ B. Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

✗ C. Submitting someone else's program as your own.

D. Copying a few lines of someone else's code into your solution, if you understand those lines in detail.

QOTD

According to the academic honesty policy, which of the following are permitted?

✓ A. Talking about your code with your classmates.

✗ B. Looking at a classmate's code, then immediately sitting down and typing out a very similar program, but with different variable names.

✗ C. Submitting someone else's program as your own.

✗ D. Copying a few lines of someone else's code into your solution, if you understand those lines in detail.

Goals

- Understand that data of different types is represented on a computer in different ways, and know the meaning of the following types:
    - `str`, `int`, `float`
- Know how to use the type conversion functions `int`, `float`, `str`
- Understand the syntax for calling functions with arguments, and know how to use the following functions:
    - `print` (with multiple arguments) `input` (with a prompt argument)
    - `type`
- Know how to name and store values using variables and the assignment operator

Last time…

Last time…

- An algorithm is a step by step procedure to solve a problem.

Last time…

- An algorithm is a step by step procedure to solve a problem.

- We sometimes use pseudocode - a description of the steps of an algorithm that is not in any particular programming language.

Last time…

- An algorithm is a step by step procedure to solve a problem.
- We sometimes use pseudocode - a description of the steps of an algorithm that is not in any particular programming language.
- Functions and function calls...

Last time: Function Calls

Last time: Function Calls

- We’ve seen two functions so far:

Last time: Function Calls

- We’ve seen two functions so far:
    - `print` and `input`

Last time: Function Calls

- We’ve seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

Last time: Function Calls

- We've seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

```
print("some text")
```

Last time: Function Calls

- We've seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

```
print("some text")
```

`"some text"` is an argument to the `print` function call

Last time: Function Calls

- We've seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

```
print("some text")
```

`"some text"` is an argument to the `print` function call

- or not:

Last time: Function Calls

- We've seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

```
print("some text")
```

`"some text"` is an argument to the `print` function call

- or not:

```
input()
```

Last time: Function Calls

- We've seen two functions so far:
  - `print` and `input`
- Functions can take inputs, called arguments

```
print("some text")
```

`"some text"` is an argument to the `print` function call

- or not:

```
input()
```

`input` is called with no arguments here

Function Calls

- Syntax for a function call:

```
print("I am", 31, "years old")
```

Function Calls

- Syntax for a function call:

```python
print("I am", 31, "years old")
```

Function name

Function Calls

- Syntax for a function call:

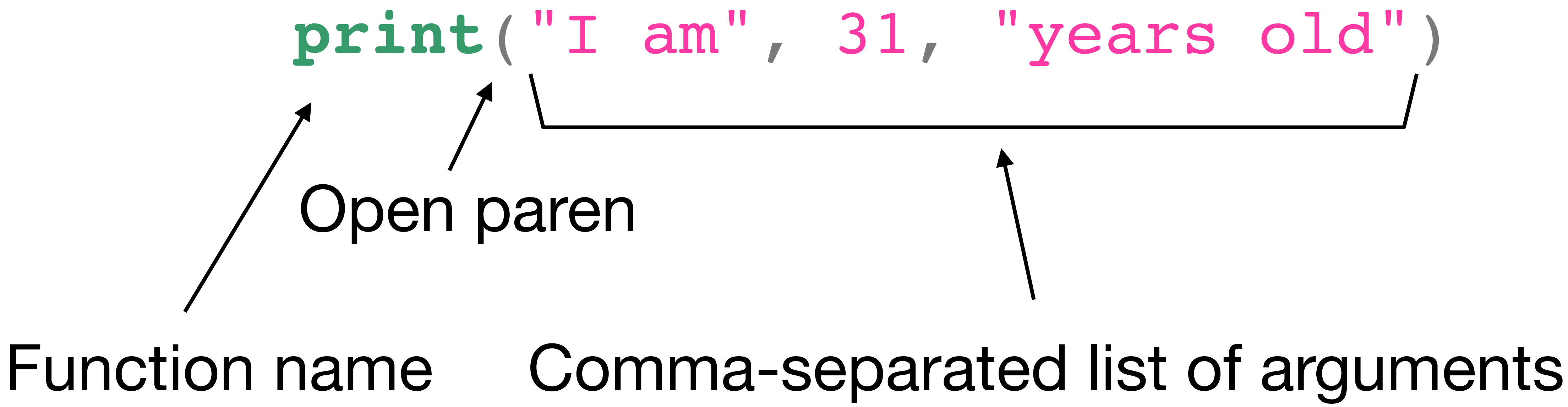
```
print("I am", 31, "years old")
```

Open paren

Function name

Function Calls

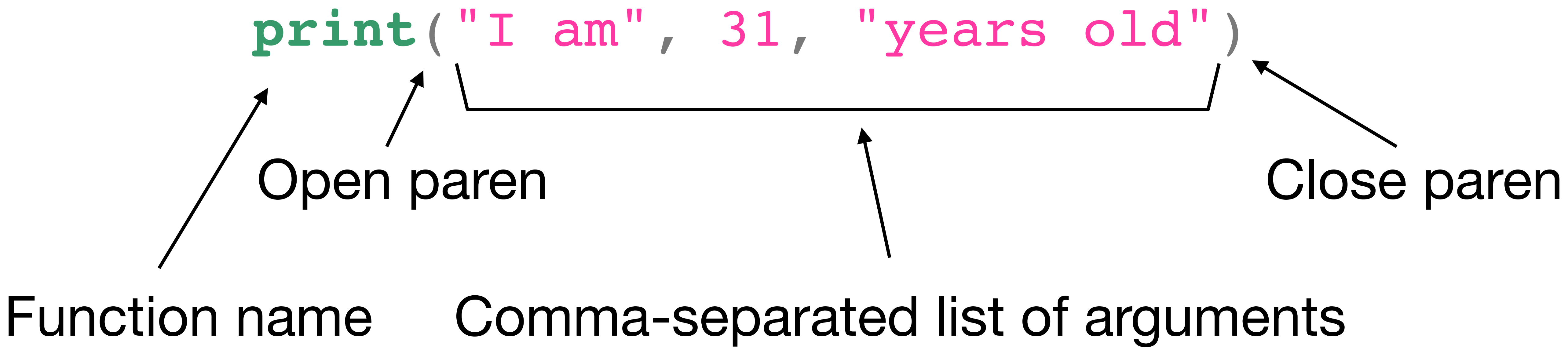
- Syntax for a function call:



```
print("I am", 31, "years old")
```

Function Calls

- Syntax for a function call:



```
print("I am", 31, "years old")
```

Poll: Print 1

What does the following code print?

```
print("CSCI", 99 + 42, "at WWU")
```

A. `CSCI141atWWU`

B. `"CSCI 141 at WWU"`

C. `CSCI 141 at WWU`

D. `CSCI 99 + 42 at WWU`

Poll: Print 2

How many **arguments** are there to the following call to the print function?

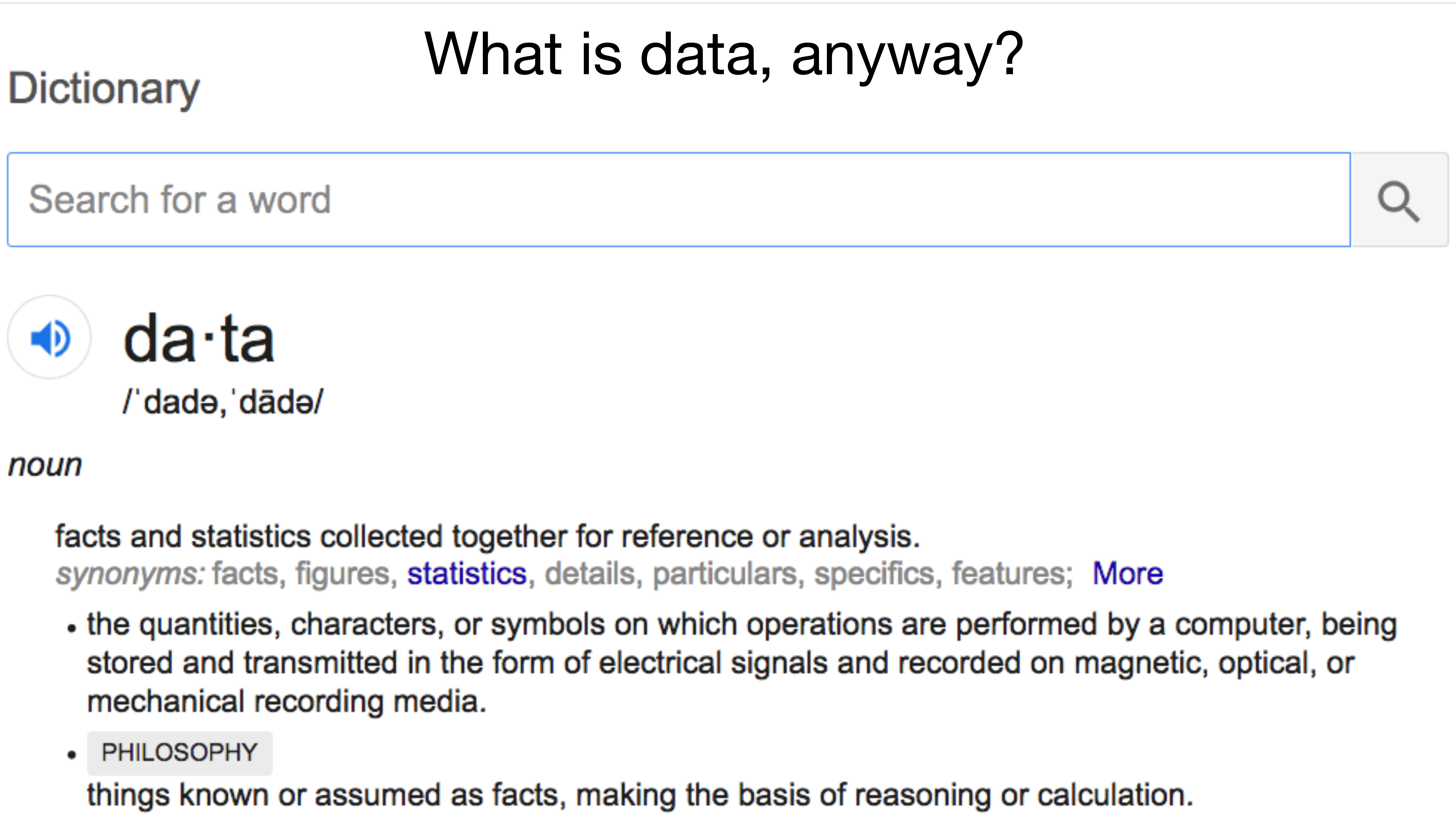
```
print("CSCI", 99 + 42, "at WWU")
```

Today: Data

What is data, anyway?

Today: Data

What is data, anyway?



Dictionary

Search for a word

da·ta

/ˈdadə,ˈdādə/

noun

facts and statistics collected together for reference or analysis.

synonyms: facts, figures, statistics, details, particulars, specifics, features; More

- the quantities, characters, or symbols on which operations are performed by a computer, being stored and transmitted in the form of electrical signals and recorded on magnetic, optical, or mechanical recording media.
- PHILOSOPHY

  things known or assumed as facts, making the basis of reasoning or calculation.

Data Types

Data Types

- Different kinds of data are stored differently.

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)
- We’ve seen 2 already:
    - “Hello world!”
    - 3 (as in 3 * 4 + 2)

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)
- We've seen 2 already:
  - "Hello world!" String (type `str`)
  - 3 (as in 3 * 4 + 2)

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)
- We've seen 2 already:
  - "Hello world!" String (type `str`)
  - 3 (as in 3 * 4 + 2) Integer (type `int`)

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)
- We've seen 2 already:
    - "Hello world!" String (type `str`)
    - 3 (as in 3 * 4 + 2) Integer (type `int`)
- Here's another:
    - 3.14

Data Types

- Different kinds of data are stored differently.
- All pieces of data have a **type** (sometimes also called **class**)
- We've seen 2 already:
  - "Hello world!" String (type `str`)
  - 3 (as in 3 * 4 + 2) Integer (type `int`)
- Here's another:
  - 3.14 Floating-point number (type `float`): a number with a decimal point

Data Types: Why?

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)

- Practical reasons:

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)
- Practical reasons:
    - Need to know how to store it in memory (how to encode it as 1's and 0's)

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)
- Practical reasons:
  - Need to know how to store it in memory (how to encode it as 1's and 0's)
  - Need to know what you can *do* with it

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)

- Practical reasons:

  - Need to know how to store it in memory (how to encode it as 1's and 0's)

  - Need to know what you can *do* with it
    - can you compute `10 + "Scott"`?

Data Types: Why?

- All pieces of data have a **type** (sometimes also called **class**)
- Practical reasons:
  - Need to know how to store it in memory (how to encode it as 1's and 0's)
  - Need to know what you can *do* with it
    - can you compute `10 + "Scott"`?
    - what about `1.1 + 2`?

Data Types

- How do you find out what type a piece of data is?
    - Just ask!
    - Python has a function called `type` which tells you the type, or class, of any value.

The `type` Function

- The type function takes one piece of data (a value) and gives back the type of the value.

- Examples:

Function call:	Result:
`type(16)`	`<class ‘int’>`
`type(“CSCI 141”)`	`<class ‘str’>`
`type(16.0)`	`<class ‘float’>`

16.0 is (mathematically) an integer, but the decimal point causes it to be interpreted as a `float`.

Got that?

What will be the result of calling:

```
type(1.2)
```

A. `<class ‘str’>`

B. `<class ‘float’>`

C. `<class ‘int’>`

D. `<class ‘String’>`

Got that?

What will be the result of calling:

```
type("1.2")
```

A. `<class 'str'>`

B. `<class 'float'>`

C. `<class 'int'>`

D. `<class 'String'>`

Data Type Conversions

- What if you have `“1.4”` (class `str`) but you want `1.4` (class `float`)?
- Here are three more functions:

```
int()

float()

str()
```

- Each tries to convert its argument to the given type, and throws an error if it’s not possible.

`type` and type conversions: demo

Types and type conversions: demo

- int to int
- int to string
- float to int
- string to int
- string to float

`print` and `input`

- `print` can take any number of arguments, of any type.
  - Non-string arguments will be converted into strings
  - Arguments are printed in sequence, separated by a space
- `input` can take zero or one arguments
  - If given one argument, the argument is printed as a prompt before waiting for input.

Advanced Print and Input: Demo

Advanced Print and Input: Demo

- Print with multiple arguments, including non-strings
- Print with no arguments
- Input with a prompt

Variables

Variables

- Variables are a basic component of all programming languages

Variables

- Variables are a basic component of all programming languages
- They simply allow you to **store** (or remember) **values**.

Variables

- Variables are a basic component of all programming languages
- They simply allow you to **store** (or remember) **values**.

Variables

- Variables are a basic component of all programming languages
- They simply allow you to **store** (or remember) **values**.
- Computers are pretty dumb, but they're really good at a few things, for example:

Variables

- Variables are a basic component of all programming languages
- They simply allow you to **store** (or remember) **values**.
- Computers are pretty dumb, but they're really good at a few things, for example:
  - arithmetic

Variables

- Variables are a basic component of all programming languages
- They simply allow you to **store** (or remember) **values**.
- Computers are pretty dumb, but they're really good at a few things, for example:
    - arithmetic
    - remembering things

Variables: Definition

- A variable is a name in your program that refers to a piece of data (or a value).

Variables: Usage

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
    1. Decide what value you want to store in the variable

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
    1. Decide what value you want to store in the variable

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
    1. Decide what value you want to store in the variable
    2. Decide on a sensible name

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
  1. Decide what value you want to store in the variable
  2. Decide on a sensible name

```
my_age    31
```

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
  1. Decide what value you want to store in the variable
  2. Decide on a sensible name
  3. In your program, use the assignment operator to assign that variable name to the value:

```
my_age    31
```

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
  1. Decide what value you want to store in the variable
  2. Decide on a sensible name
  3. In your program, use the assignment operator to assign that variable name to the value:

```
my_age = 31
```

Variables: Usage

- A variable is a name in your program that refers to a piece of data (or a value).
- How do you use them?
  1. Decide what value you want to store in the variable
  2. Decide on a sensible name
  3. In your program, use the assignment operator to assign that variable name to the value:

```
my_age = 31
```

The assignment operator.

Variables: Usage

```
my_age = 31
```

The assignment operator.

Variables: Usage

```
my_age = 31
```

$\nwarrow$ The assignment operator.

- For now, think of `my_age` as a named place where we can store any value.
- You can replace the current value with a different one:

Variables: Usage

```
my_age = 31
```

The assignment operator.

- For now, think of `my_age` as a named place where we can store any value.
- You can replace the current value with a different one:

```
my_age = 32
```

Variables: Usage

```
my_age = 31
```

The assignment operator.

- For now, think of `my_age` as a named place where we can store any value.
- You can replace the current value with a different one:

```
my_age = 32
```

The Assignment Operator: Not “Equals”

```
my_age = 32
```

The assignment operator.

- This is not *stating an equality*, like in math.
- It is *associating a name with a value*.

```
my_age = 31
my_age = 32
```

(whiteboard) a simple diagram of what's happening here

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 31
my_age = 32
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

"my_age equals 32"

"my_age gets 32"

"my_age becomes 32"

"the variable my_age takes on the value 32"

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 31
my_age = 32
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

✗ “my_age equals 32”

“my_age gets 32”

“my_age becomes 32”

“the variable my_age takes on the value 32”

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 31
my_age = 32
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

✗ "my_age equals 32"

✓ "my_age becomes 32"

"my_age gets 32"

"the variable my_age takes on the value 32"

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 31
my_age = 32
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

✗ "my_age equals 32"

✓ "my_age gets 32"

✓ "my_age becomes 32"

"the variable my_age takes on the value 32"

Using Variables

- Assigning a value is **not** stating an equality, like in math: it's storing a value.

```
my_age = 31
my_age = 32
```

A variable's value can be **updated** (overwritten) by a new value using the assignment operator.

✗ "my_age equals 32"

✓ "my_age becomes 32"

✓ "my_age gets 32"

✓ "the variable my_age takes on the value 32"

What can you do with variables?

Use them anywhere you'd use a value!

```
print(5)
```

```
a = 5
print(a)
```

These two programs both print 5.

Variable Names

Variable Names

- How do you use variables?
  1. Decide what value you want to store in the variable
  2. **Decide on a sensible name**
  3. In your program, use the assignment operator to store that value in the variable

Variable Names

- How do you use variables?
  1. Decide what value you want to store in the variable
  2. **Decide on a sensible name**
  3. In your program, use the assignment operator to store that value in the variable
- Great power, great responsibility: variables names can be almost anything!

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

```
True   2plus2    a_number   firstOfThreeValues
```

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

~~`True`~~ `2plus2` `a_number` `firstOfThreeValues`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

~~`True`~~ ~~`2plus2`~~ `a_number` `firstOfThreeValues`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

```
✗ True    ✗ 2plus2    ✓ a_number    firstOfThreeValues
```

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- **Valid** variable names:
  - start with a letter or an underscore (_)
  - can contain any letters and digits
  - are case-sensitive (name is not the same as Name)
  - are not the same as any Python language **keywords** (words that already mean something else):

```
False, None, True, and, as, assert, async, await, break, class,
continue, def, del, elif, else, except, finally, for, from,
global, if, import, in, is, lambda, nonlocal, not, or, pass,
raise, return, try, while, with, yield
```

✗ `True`   ✗ `2plus2`   ✓ `a_number`   ✓ `firstOfThreeValues`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

```
current_time              a4           hair_color
        midterm_exam_grade_as_a_percent
```

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

```
✓current_time            a4           hair_color
        midterm_exam_grade_as_a_percent
```

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

```
✓current_time            a4          ✓hair_color
          midterm_exam_grade_as_a_percent
```

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

✓ `current_time` `a4` ✓ `hair_color`

✗ `midterm_exam_grade_as_a_percent`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long
  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

✓ `current_time` ✗ `a4` ✓ `hair_color`

✗ `midterm_exam_grade_as_a_percent`

Variable Names

- Great power, great responsibility:
  variables names can be almost anything!
- A **good** variable name:
  - is descriptive - tell a reader what data they refer to
  - is not too long

  these depend on context!

  - follows a standard naming convention, e.g.:
    - starts with lower case letter
    - words are separated by underscores

✓`current_time` ✗`a4` ✓`hair_color`

✗`midterm_exam_grade_as_a_percent`

Next time

- More variables
- Operators
- Expressions
- Arithmetic