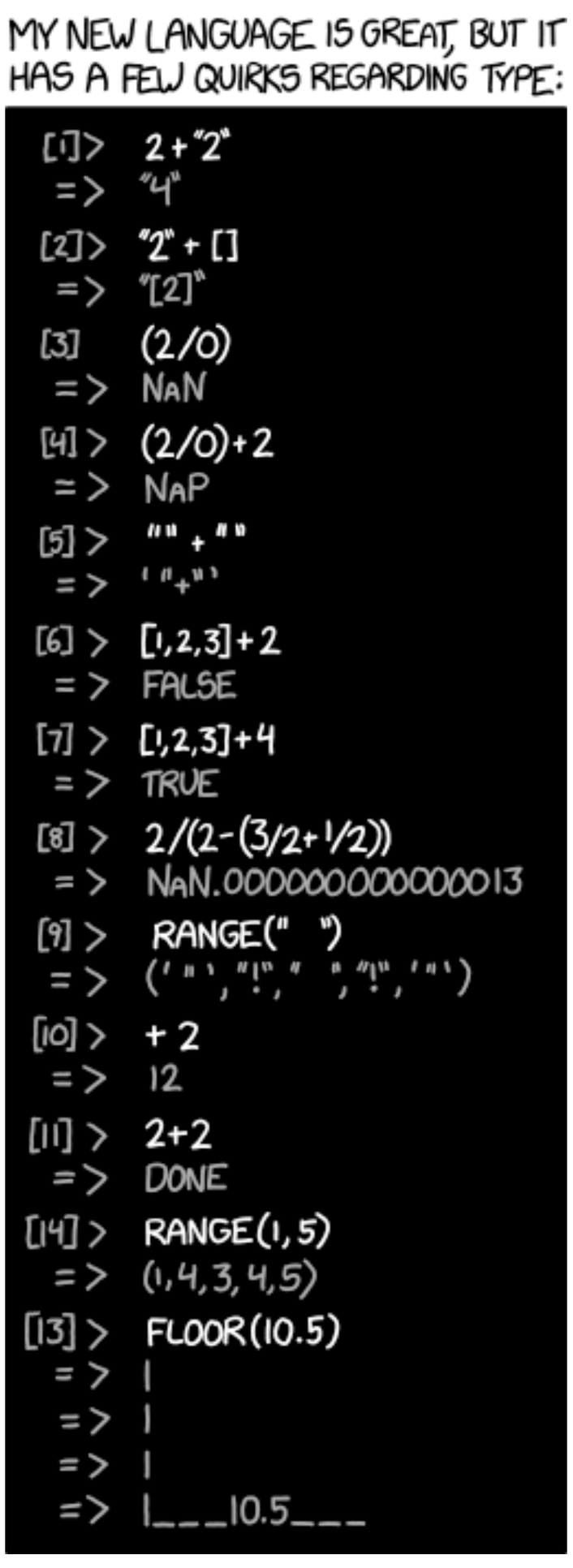
CSCI 141

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



Announcements

- Assignment 1 is out
  - Written questions and a programming problem
  - Due next Monday
  - You'll know everything you need to know to complete it by Friday's lecture, but you can get started earlier than that.
  - Please keep track of the hours you spend

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

Last time…

- Recall: 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.

Warm-Up: Sandwich

- Write pseudocode for an algorithm to make a PB&J sandwich.

- Suppose you're given:
  - A fridge that contains a jar of peanut butter and a jar of jam
  - A counter on which there is a bag with a loaf of sliced bread.

(3 minutes)

Exercise: Sandwich

- Compare your pseudocode to your neighbor's. Could you follow the instructions?
- Could an alien who'd never heard of a sandwich follow the instructions?

The Point

- Computers are the aliens in this story:
  - they can’t “fill in the gaps”
  - they don’t “know what you meant”
- Computers are stupid.
- You have to be **precise** and **patient** in order to communicate with them.

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

- 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?

- 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.

Detour: Calling Functions

- We’ve seen two functions so far:
  - `print` and `input`
  - What exactly is a function? More on this later.
  - For now: it’s a thing that calculates or does something.

Calling Functions

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

```
print("A string")
```

  "`A string`" is an argument to the `print` function call

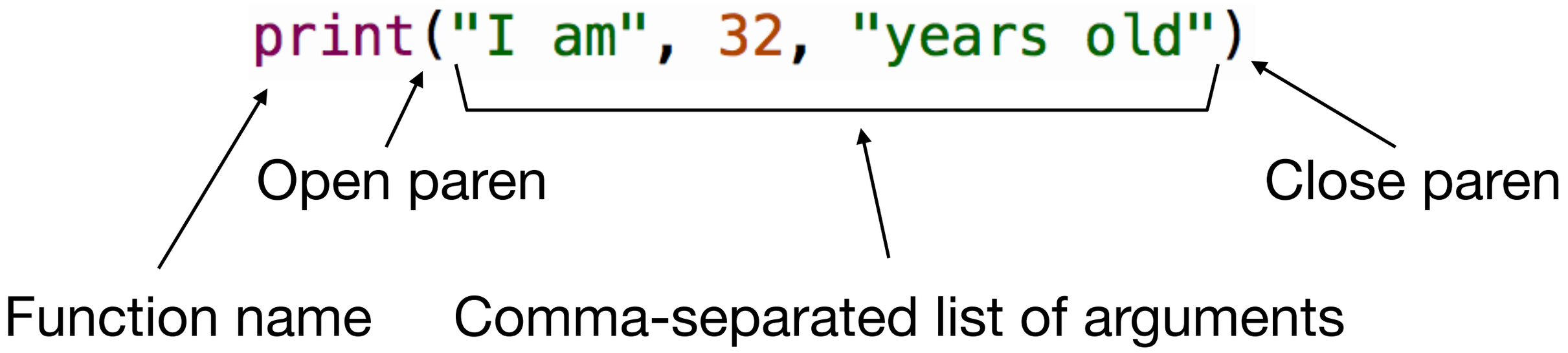
- or not:

```
input()
```

  `input` is called with no arguments here

Calling Functions

- Syntax for a function call:



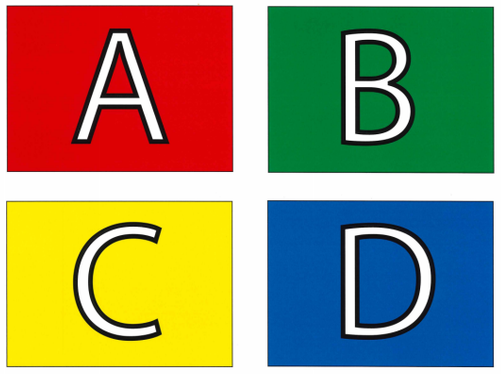
The `type` Function

- 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'>`

Even though 16.0 is an integer, the decimal causes it to be interpreted as a float.



Got that?

What will be the result of calling:

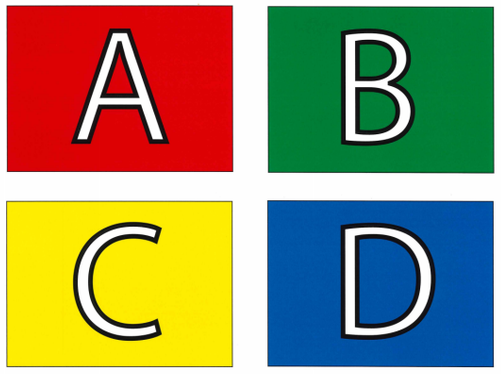
```
type(1.2)
```

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.

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, Revisited

- `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.

Print and Input: Demo

Print and Input: Demo

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

Variables

- Variables are a basic component of all programming languages

- They simply allow you to **store** (or remember) **values**.

- Remembering is one thing computers are better at than humans. Try remembering these numbers:

5, 8, 12, 44, 89, 65, 44, -67, 43.4, 32

Variables: Definition

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

Variables: Usage

- A variable is a name in a 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 store that value in the variable:

```
my_age = 32
```

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 store that value in the variable:

```
my_age = 32
```

The assignment operator.

Variables: Usage

```
my_age = 32
```

The assignment operator.

- 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 = 33
```

The Assignment Operator: Not “Equals”

```
my_age = 32
```

The assignment operator.

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

```
my_age = 32
my_age = 33
```

The Assignment Operator: Not “Equals”

```
my_age = 32
```

The assignment operator.

❌ “my_age equals 32”

✅ “my_age becomes 32”

✅ “my_age gets 32”

✅ “the variable my_age takes on the value 32”