CS 161: Introduction to Programming and Problem-solving

Warren Harrison

Variables and Simple I/O

PORTLAND STATE UNIVERSITY

Python Output

- We’ve learned how to display ***string constants*** using the print function:

```
print(“hello world”)
```

- A string like “hello world” has a *constant* value
- Only useful if you want to display the same message every time you run the program

2

PORTLAND STATE UNIVERSITY

Variables

- Recall memory (RAM) is a contiguous sequence of addressable "boxes" that can hold data
- Variables are *symbolic* representations of memory location addresses
- A variable can be considered to "point to" a memory location just like the URL of a web page

3

PORTLAND STATE UNIVERSITY

URLs

- A URL like

  http://web.pdx.edu/~warren

  is the *symbolic address* of a web page, it isn't the web page itself
- Your browser will display the contents of the page – if someone changes the page contents, those will be displayed next time you visit that URL

4

PORTLAND STATE UNIVERSITY

Variables and URLs

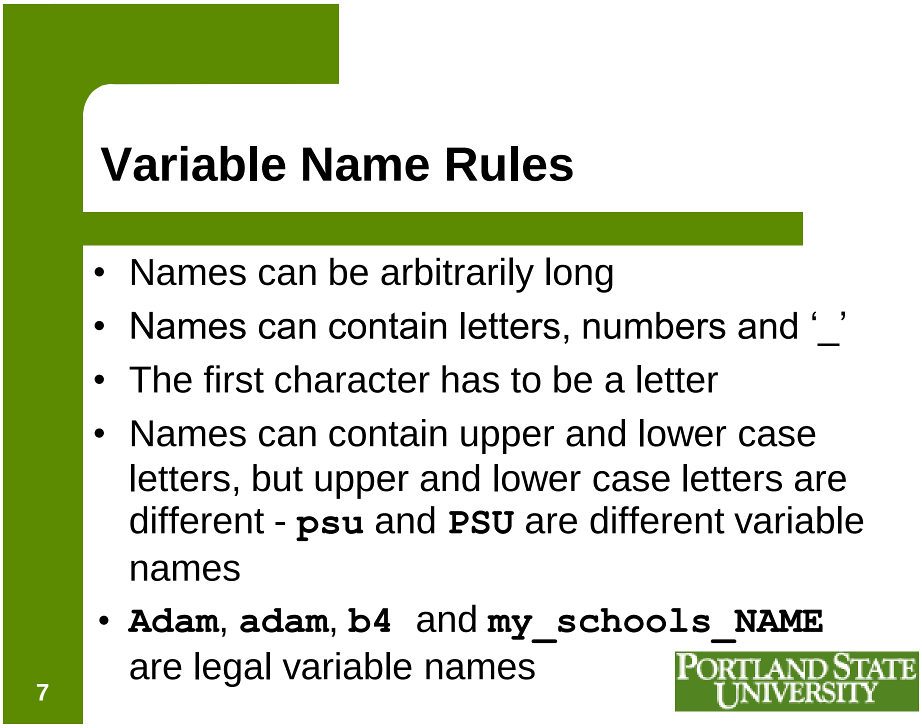
- We say a URL “points” to a web page
- We say a variable “points” to a memory location
- If you use the name of a variable in a Python instruction, the contents of the memory location the variable “points to” will be used - we often say that the variable *returns* a particular value

Variable Assignments

- In order for a variable to return a value from a memory location, we have to put a value there
- We use an ***assignment statement*** to put a value in a memory location:

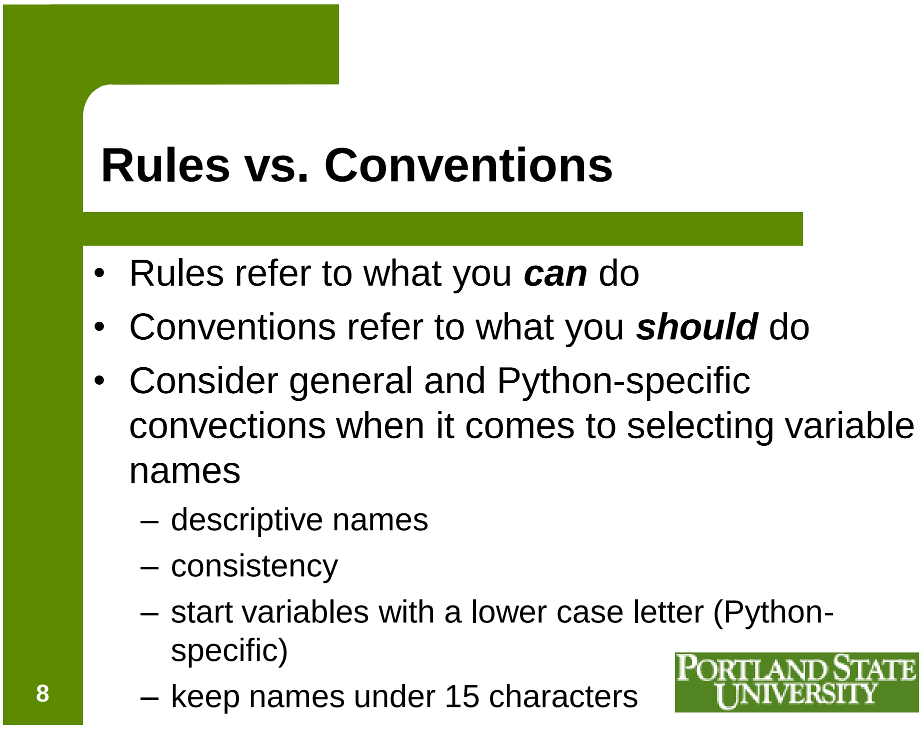
```
X = “Hello World”
```

- This associates the variable name **`X`** with a memory location, and places (or ***assigns***) the string “**`Hello World`**” in that memory location



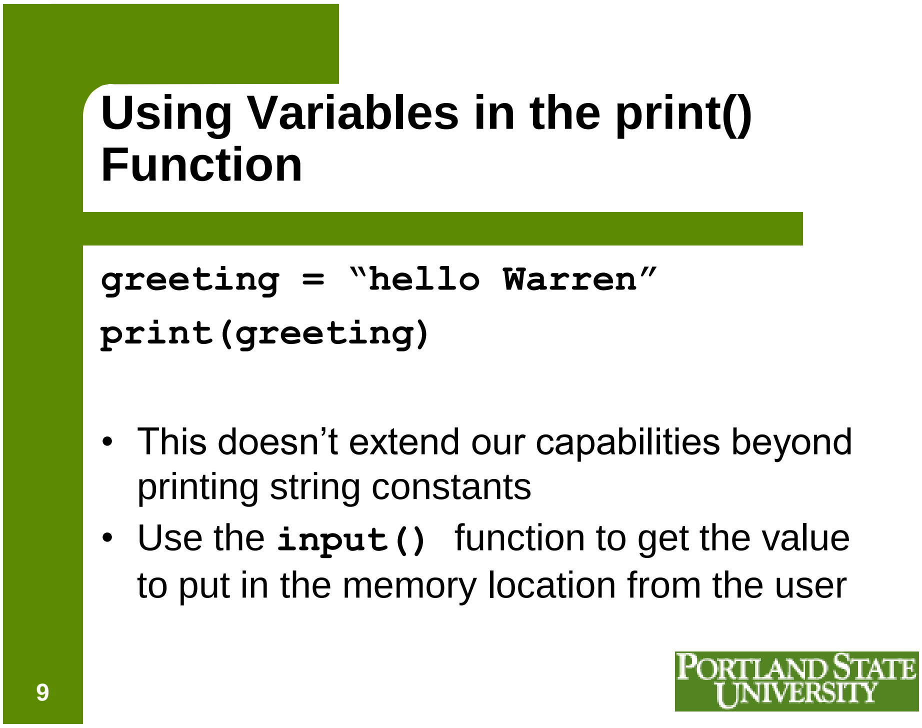
Variable Name Rules

- Names can be arbitrarily long
- Names can contain letters, numbers and '_'
- The first character has to be a letter
- Names can contain upper and lower case letters, but upper and lower case letters are different - **`psu`** and **`PSU`** are different variable names
- **`Adam`**, **`adam`**, **`b4`** and **`my_schools_NAME`** are legal variable names



Rules vs. Conventions

- Rules refer to what you ***can*** do
- Conventions refer to what you ***should*** do
- Consider general and Python-specific convections when it comes to selecting variable names
  - descriptive names
  - consistency
  - start variables with a lower case letter (Python-specific)
  - keep names under 15 characters

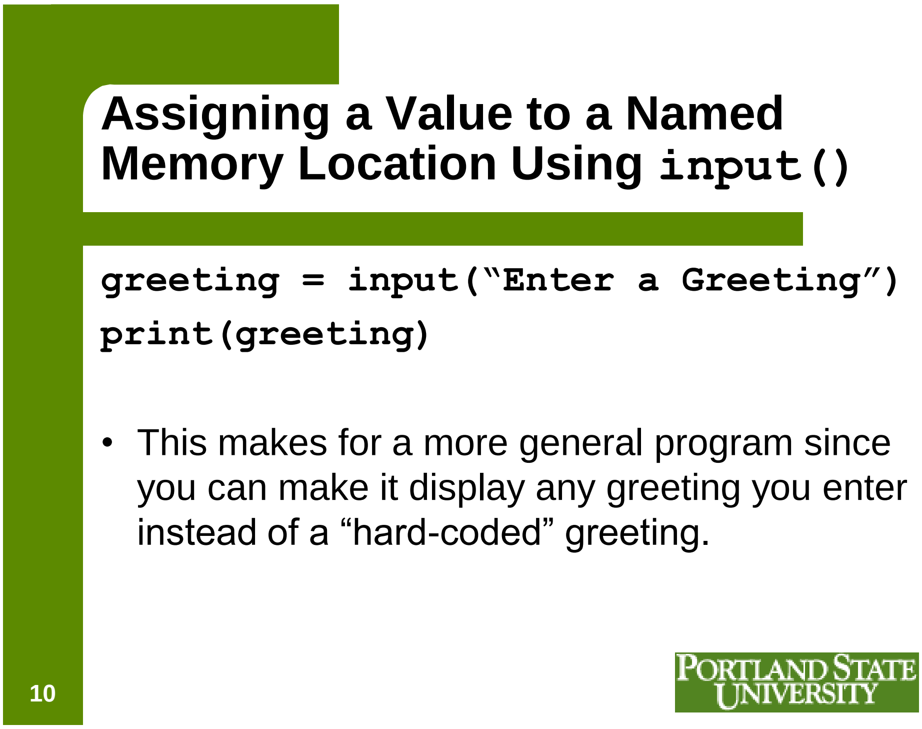


Using Variables in the print() Function

```
greeting = “hello Warren”
print(greeting)
```

- This doesn’t extend our capabilities beyond printing string constants
- Use the `input()` function to get the value to put in the memory location from the user

9



Assigning a Value to a Named Memory Location Using `input()`

```
greeting = input(“Enter a Greeting”)
print(greeting)
```

- This makes for a more general program since you can make it display any greeting you enter instead of a “hard-coded” greeting.

10

We Can Also Have Multiple Variables

```
firstName=input(“First Name: “)
lastName=input(“Last Name: “)
print(“Dear Mr./Ms. “,lastName)
print()
print(“May I call you “,firstName,”?”)
print(“I’d like to sell you a swamp”)
print(“in Damascus, Oregon”)
```

Swamp Sales R Us

```
>>>
First Name: Warren
Last Name: Harrison
Dear Mr./Ms.  Harrison

May I call you Warren ?
I’d like to sell you a swamp
in Damascus, Oregon
>>>
```

The Syntax & Semantics of the `input()` Function

- *variable name* = **`input(`***argument***`)`**
- Allows a user to enter strings at the keyboard
- The argument is a single string that “prompts” the user to enter something – it can be a string constant or a variable referencing a string
- The *variable name* references the memory location that the value the user enters should be placed

PORTLAND STATE UNIVERSITY

Escape Sequences for `print()`

- An escape sequence allows you to put “special characters” or characters you wouldn’t ordinarily be able to use into a string
- An escape sequence always begins with a backslash: “**`\`**”
- You can insert a quote: `\’` or `\”`
- … or a non-printable character

PORTLAND STATE UNIVERSITY

Escape Sequences

- \\ prints a backslash
- \’ prints a single quote
- \” prints a double quote
- \n inserts a new line
- \t inserts a tab
- \a rings the bell (may not work within IDLE)

Data Types

- A string is a **type** of data
- It can contain letters, numbers or special characters

  `myAddress=“1900 SW 4th Avenue”`
- You can’t add, subtract, multiply or divide strings – even if they have numbers in them
- A string is a string because of its “intended usage”, not simply what characters are in it
- e.g., Student ID Number

Arithmetic

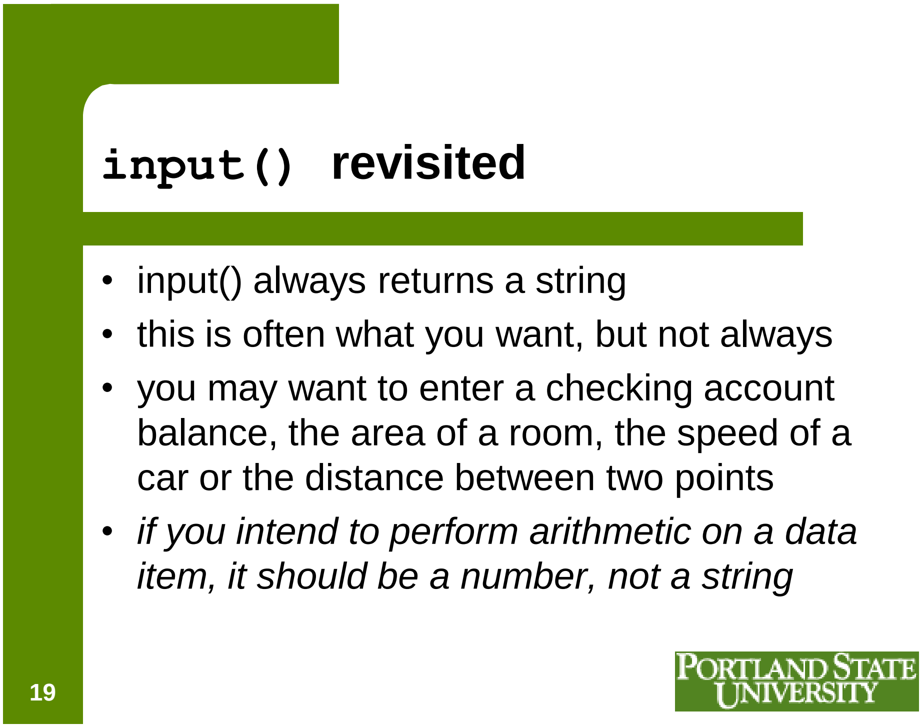
- We can’t do arithmetic on strings, but we often *do* want to do arithmetic
- We need a numeric data type
- A numeric data type can contain numbers for use in computations
- +
- -
- *
- /

17

For Example

- speed = distance / time
- area = length * width
- celsius = (fahrenheit - 32) * 5 / 9
- fahrenheit = celsius * 9 / 5 + 32

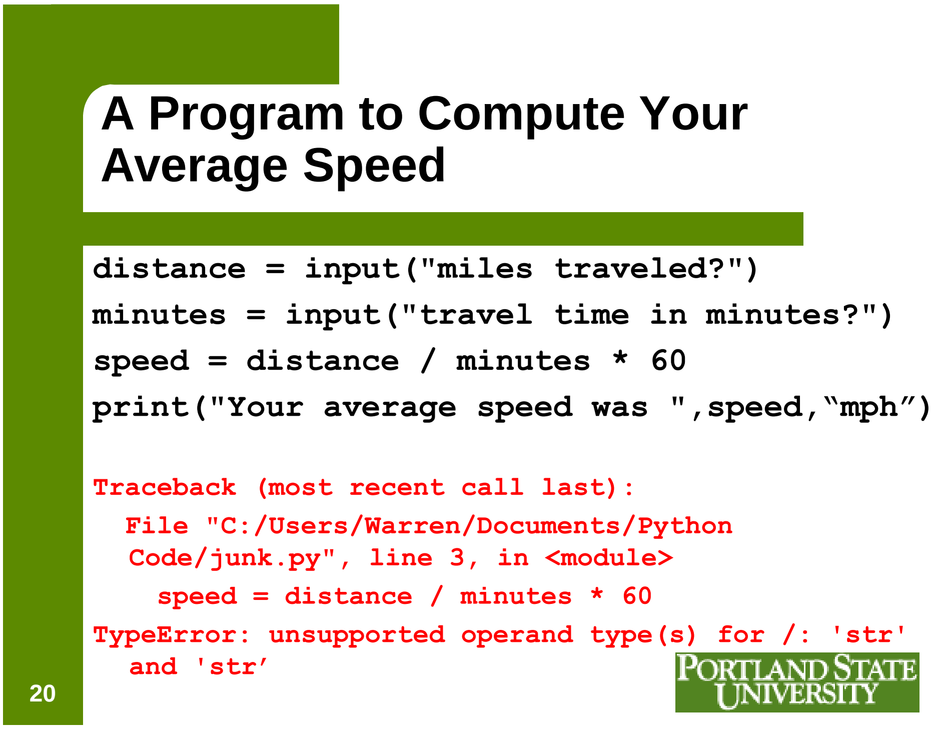
18



`input()` revisited

- input() always returns a string
- this is often what you want, but not always
- you may want to enter a checking account balance, the area of a room, the speed of a car or the distance between two points
- *if you intend to perform arithmetic on a data item, it should be a number, not a string*

19

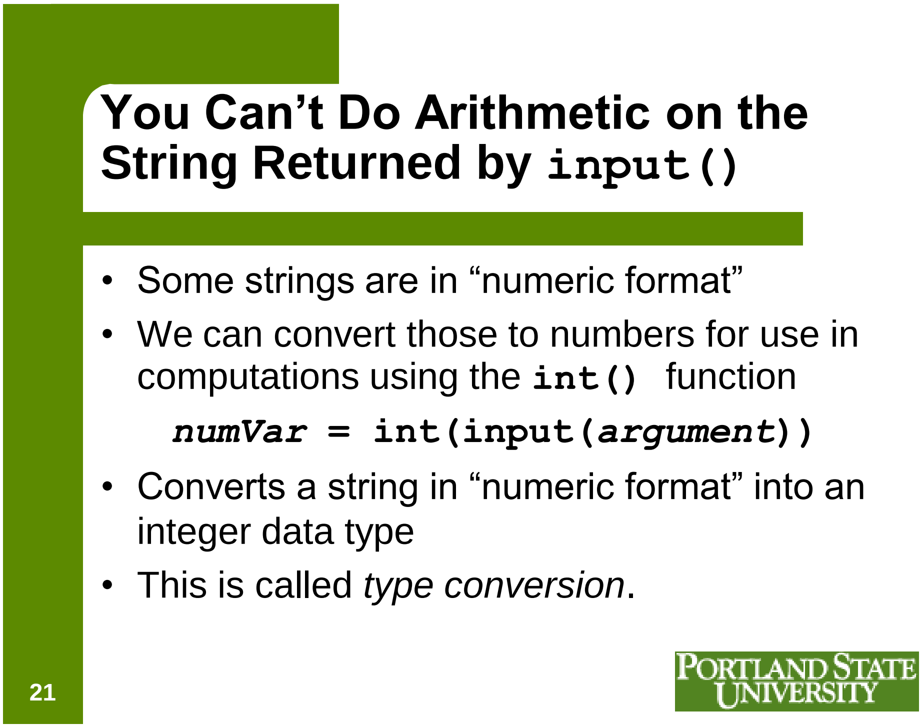


A Program to Compute Your Average Speed

```
distance = input("miles traveled?")
minutes = input("travel time in minutes?")
speed = distance / minutes * 60
print("Your average speed was ",speed,“mph”)

Traceback (most recent call last):
  File "C:/Users/Warren/Documents/Python
  Code/junk.py", line 3, in <module>
    speed = distance / minutes * 60
TypeError: unsupported operand type(s) for /: 'str'
  and 'str’
```

20

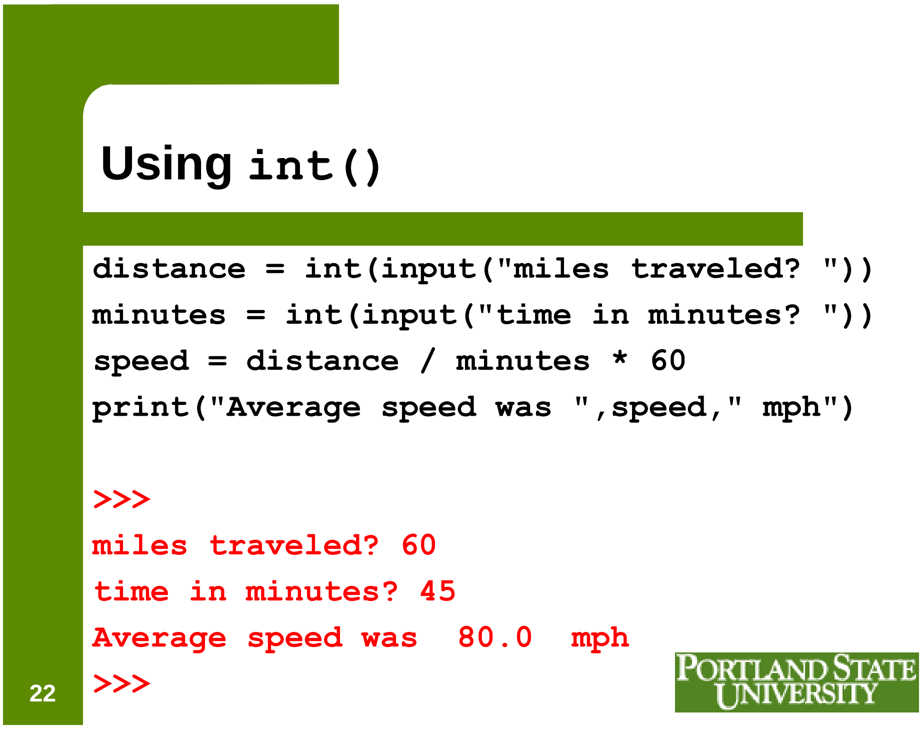


You Can’t Do Arithmetic on the String Returned by `input()`

- Some strings are in “numeric format”
- We can convert those to numbers for use in computations using the **`int()`** function

  numVar **`= int(input(`*argument*`))`**

- Converts a string in “numeric format” into an integer data type
- This is called *type conversion*.



Using `int()`

```
distance = int(input("miles traveled? "))
minutes = int(input("time in minutes? "))
speed = distance / minutes * 60
print("Average speed was ",speed," mph")

>>>
miles traveled? 60
time in minutes? 45
Average speed was  80.0  mph
>>>
```

Other Types of Type Conversion

- int()
- float()
- str()

- Differences between an integer and a float …