Variables, Input, Output

CS 110 Beste Filiz Yuksel

Review

To run a Python script from the command line you need to be in the correct directory where the script is located or list out the correct filepath from where you can located.

E.g. To run `hello_world.py` you need to be in your

.../Documents/cs110/in_class/week1 directory where your hello_world.py file is located.

Then type

```
python3 hello_world.py
```

Program Basics

- ➔ Input
- ➔ Processing
- ➔ Output

What do we know about output so far?

```
print(‘Hello world!’)
```

Display *output* with `print`

A *function* is a piece of prewritten code that performs an operation (we will cover functions in depth later in the semester).

`print` function displays output on the screen.

An *argument* is the data/information given to a function.

e.g. in this example the data that is printed to screen is `Hello world`:

```
print (‘Hello world’)
```

Input

What kind of input might you want from a user?

What should we do with the input?

Where should we store that input?

How should we access that data?

- A *variable*

Variable

A variable is a name that represents a value stored in the computer's memory.

OR: a variable is just a name that refers to a piece of data in memory.

Programs use variables to access and manipulate data that is stored in memory.

The variable ***references*** a value.

Creating Variables with Assignment Statements

```
dollars = 100
```

An assignment statement is written as: *variable = expression*

(where *expression* is a value or any piece of code that results in a value).

The equal sign = is known as the assignment operator.

After an assignment statement executes, the variable on the left side of the *assignment operator (=)* will *reference* the value given on the right side.

Outputting a variable (in interactive mode):

```
>>> dollars = 100

100

>>> print (dollars)

100
```

Variable Reassignment

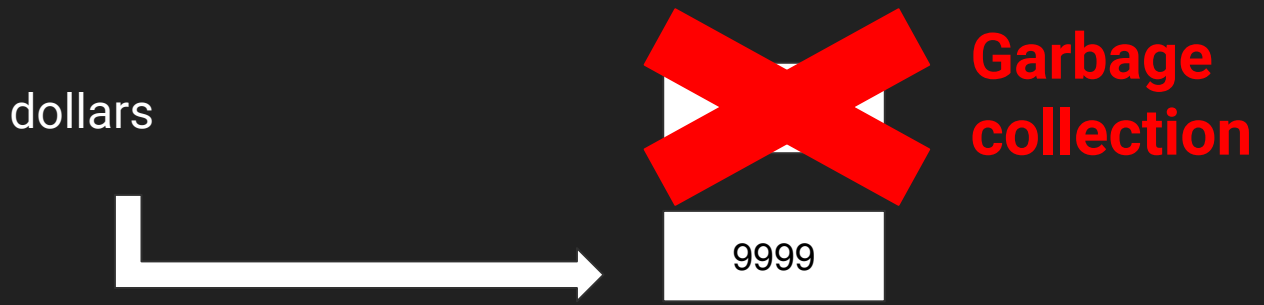
```
dollars = 100
```


```
dollars = 9999
```

Variable Reassignment

```
dollars = 100
```


```
dollars = 9999
```



Data Types

Data types are used to categorize values in memory.

When an integer is stored in memory, it is classified as an `int`.

(An integer is a whole number (not a fraction)).

When a real number is stored in memory, it is classified as a `float`.

(In mathematics, a real number is a value of continuous quantity that can represent a distance along a line)

You can use the built-in `type` function in interactive mode to determine the data type of a value.

```
>>> dollars_int = 10

>>> type (dollars_int)

<class ‘int’>

>>> dollars_float = 10.0

>>> type (dollars_float)

<class ‘float’>
```

What happens if you type:

```
dollars_float = 10
```

is it an int or a float type?

String Data Type

The data type `str` is used for storing strings in memory.

```
first_name = ‘Ada’                    # https://en.wikipedia.org/wiki/Ada_Lovelace

last_name = ‘Lovelace’

print (first_name, last_name)
```

This is a mathematical formula that results in an integer:

```
>>> print (2 + 3)
5
```

This is a String:

```
>>> print (‘2 + 3’)
2 + 3


>>> print (‘2 + 3 = ’, 2+3)
2 + 3 = 5
```

addition and multiplication

```
>>> print (2 + 3)
5


>>> print (2 * 3)
6
```

Revisting Variables

Remember that we said:

A variable is a name that represents a value stored in the computer's memory.

and

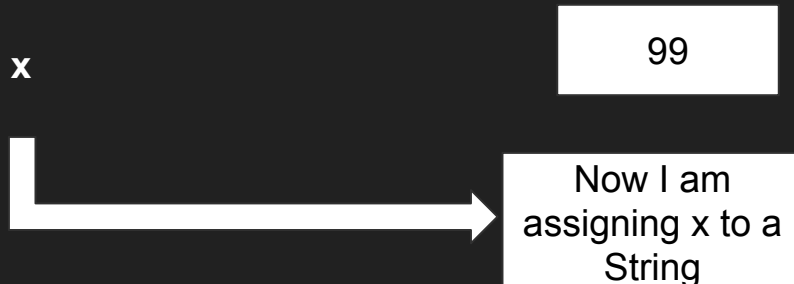
The variable ***references*** a value.

The value a variable references can be of any *type*.

Reassigning Variables to a Different Type versus Data Type Conversion

Reassigning a variable to a different type:

```
>>> x = 99
>>> print (x)
99
>>> x = ‘now I am assigning x to a String’
>>> print (x)
now I am assigning x to a String
```



Reassigning Variables to a Different Type versus Data Type Conversion

Data Type Conversion:

To convert a value to int or float you can use the **`int( )`** and **`float( )`** functions

```
>>>float_value = 3.7

>>>type(float_value)

<class ‘float'>

>>>int_value = int(float_value)

>>>type(int_value)

<class 'int'>
```

Now try converting an int into a float

Data Type Conversion

```
float_value = 3.7
```


```
int_value = int(float_value)
```

Data Type Conversion

```
float_value = 3.7
```


```
int_value = int(float_value)
```

int ( ) *truncates* the floating-point argument

Input Function

Many programs that you will write will need to read input and perform an operation on that input.

We will look at the input operation of reading data that has been typed on the keyboard `input( )`

`variable = input(`*`prompt`*`)`

E.g.

```
>>>name = input (‘What is your name? ’)

What is your name?  Countess Lovelace

>>>print(name)

Countess Lovelace
```

Input - Reading Numbers

The input function always returns the user's input as a **string** even if the user enters numeric data.

```
>>>age = input ('How old are you, Countess? ')

How old are you, Countess?    202

>>>print(age)

202

>>>type(age)

<class 'str'>
```

To overcome this problem you can use *nested function calls* so that the value that is returned from the `input()` function is passed as an argument to the `int()` function

```
>>>age = int (input (‘How old are you?’))

How old are you? 202

>>>type(age)

<class ‘int’>
```

Try the age prompt in a script and run the script.

Now you can do In-Class Exercise 1