Python, Day 1.5: Data Types and Variables

Andrew Bydlon

January 2, 2019

Data Types

Definition

A **data type** is a categorization of information that is understood by Python's interpreter. It is used to distinguish functions/operations that can be performed on a given piece of data.

Examples are as follows:

- Integers: Numbers $0, 1, 2, 3, 4, -1, -2, \ldots$. Of type *"int"*
- Floating Points: Decimals; $0.1246812, 3.14159, -2.0, \ldots$. Of type *"float"*
- Complex Numbers: $i, 1 + 2i, 1.0 + 2.0i$. Of type *"complex"*
- String: "Once upon a time", "Logic will get you from A to Z", 'Imagination will take you everywhere', '- Albert Eistein!'. Of type *"string"*.
- (The list goes on)

Checking the *type* of an object

In python, it is easy to check what the type of an object is. Just use the **type()** function.

- type(5)
  <class 'int'>
- type(5.0)
  <class 'float'>
- type(5+0j)
  <class 'complex'>
- type("Hello, World!")
  <class 'str'>
- type({1,2,3,4})
  <class 'set'>

Variables

Definition

Variables are stored memory locations to store values. This data can be manipulated and accessed.

The precise identifier of a variable (intuitively, an address) can be detected with the *"id"* function:

```
MyVariable = 25
id(MyVariable)
139747977115616
```

This is often more useful for under the hood operations such as programming python functions than to you, the user of python.

Declaring variables

Variables are always declared, with the variable appearing on the left, and its value on the right:

```
variable_name = some_expression
```

The effect of this code is to

a) Place some_expression in memory somewhere.

b) point variable_name to it.

Important Note: Note that variable_name doesn't contain data itself. It is merely a *pointer* to the data.

Additionally, some_expression needs to be declared in advance.

If we continue with this line of code, we can manipulate the variable.

```
id(variable_name)
139747977115616
variable_name = variable_name + variable_name
id(variable_name)
139747977116416
```

Remark

We have done an operation on some_expression. The effect of this is to create a new memory address, store the new value, and to update the pointers to the new expression.

The old data is removed from memory by garbage collection.

Rules for declaring variables

Variable names cannot be arbitrary strings of characters. Here are the rules:

1. The set of usable characters are
   - Capital letters: (A-Z)
   - Lower case letters: (a-z)
   - Numbers: (0-9)
   - An underscore: (_)
2. It cannot begin with a number, e.g. 8Variable.
3. Certain words are reserved keywords, for data types or functions. You cannot name variables identically to these words.
4. Python is case sensitive: HoMe, hOmE, HOME, and home can all be declared independently.

Note: Spaces are not in the list of valid variable characters.

A quick note on multi-assignment

We don't need to use a separate line for each variable assignment.

Here is the syntax:
var1, var2, . . . , varN = dec1, dec2, . . . , decN

Writing comments

We are able to write comments for ourselves or other coders within our code. To do this, we use the # symbol, followed by a space.

The remainder of the line will then be ‘commented out’, or not accessed by the executed program.

Example

```
print("Hello, World!") # This prints the phrase "Hello, World!" to user.
# Hello fellow programmer!
```

Using the print command

As you may have noticed, print() is used to supply information to the user.

When writing a script, unlike in the interactive shell, no information will be returned to the user.

print() can be executed on many elements in succession:

print(arg1, arg2, arg3, arg4, . . .)

Example

```
My_Favorite_Number = 5
print("My favorite number is", My_Favorite_Number, "!")
```

The result is: My favorite number is 5 !

User input

A program that couldn't be used without accessing the source code is relatively useless; it should be usable by the user!

The desired function to perform such an action is input().

It always returns a string. To convert it to a different type, you can precede it with the type you would like to use, e.g. int(input(..)) or float(input(..))

Example

```
UserInfo = input("Please enter your name: ")
print("Hello ", UserInfo, ". Nice to meet you!")
```

Quick exercise 1

Write a program that collects the following information

1. The user's favorite type of candy (string)
2. The user's prefered amount of the given candy (int)
3. The cost of 1 unit of the given candy (float)

Then print the following statement:

The cost of (Fill in Blank) units of (Fill in Blank) is $(Fill in Blank).

Products are done with *.

Submit your .py file to canvas when you finish.