Code Academy

Chapter Two

PROGRAMMING WITH NUMBERS AND STRINGS

Sections 2.1
Variables

Code Academy

Introduction

- **Numbers** and character **strings** are important **data types** in any Python program
  - These are the fundamental building blocks we use to build more complex **data structures**
- In this chapter, you will learn how to work with numbers and text. We will write several simple programs that use them

Code Academy

Lecture Goals

- To **declare** and **initialize** variables and constants
- To understand the properties and limitations of **integers** and **floating-point** numbers
- To appreciate the importance of comments and good code layout

Code Academy

Variables

- A **variable** is a named storage location in a computer program
- There are many different **types** of variables, each type used to store different things
- You **define** a variable by telling the compiler:
  - - What **name** you will use to refer to it
  - - The **initial value** of the variable
- You use an **assignment** statement to place a value into a variable

```
cansPerPack = 6
print(cansPerPack)
```

An example: soda deal

- Soft drinks are sold in cans and bottles. A store offers a six-pack of 12-ounce cans for the same price as a two-liter bottle. Which should you buy?

  12 fluid ounces equal approximately 0.355 liters.

List of variables:	Type of Number
Number of cans per pack	Whole number
Ounces per can	Whole number
Ounces per bottle	Number with fraction

The assignment statement

- Use the **assignment statement** '=' to place a new value into a variable

```
cansPerPack = 6   # define & initializes the variable
cansPerPack
```

- The left-hand side (LHS) of an assignment statement consists of a **variable**
- The right-hand side (RHS) is an **expression** that has a value
- Beware: The "**=**" sign is NOT used for comparison:
  - It copies the value on the right side into the variable on the left side
  - You will learn about the comparison operator in the next chapter

Why different types?

- There are three different types of data that we will use in this chapter:
  - A whole number (no fractional part) `-7` `(integer or int)`
  - A number with a fraction part `8.88` `(float)`
  - A sequence of characters `"Bob”` `(string)`
- The data type is associated with the **value**, not the **variable**:

```
cansPerPack = 6     # int
canVolume = 12.0    # float
```

Updating a Variable (assigning a value)

- If an existing variable is assigned a new value, that value replaces the previous contents of the variable.
- For example:
  - cansPerPack = 6
  - cansPerPack = 8

1 Because this is the first assignment, the variable is created.

```
cansPerPack =
```

2 The variable is initialized.

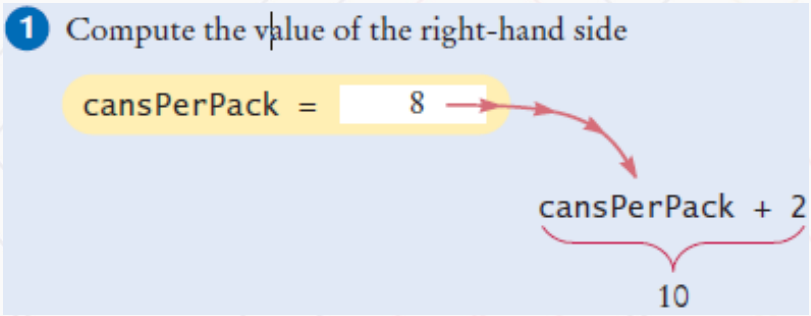
```
cansPerPack = 6
```

3 The second assignment overwrites the stored value.

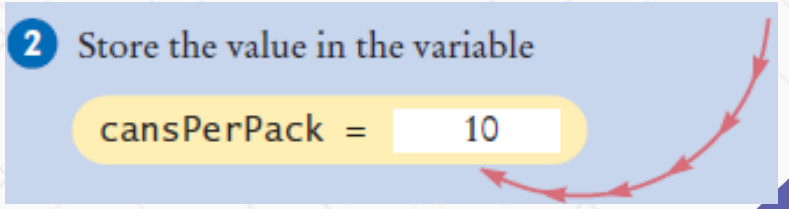
```
cansPerPack = 8
```

Updating a Variable (computed)

- Executing the Assignment:
  cansPerPack = cansPerPack + 2
- Step by Step:
  - Step 1: Calculate the right hand side of the assignment. Find the value of cansPerPack, and add 2 to it.



  - Step 2: Store the result in the variable named on the left side of the assignment operator



Number Types

- In Python, there are several different types of numbers
  - An **integer** value is a whole number without fractions

    Example

    6 -32 -2 0 24
  - A **floating-point** number is used when a fractional part is required.

    Example

    0.5 1.0 -9.4 1E6 2.96E-2
- When a value such as 6 or 0.355 occurs in a Python program, it is called a **number literal**.

Table 1: Number Literals in Python

Table 1 Number Literals in Python

Number	Type	Comment
`6`	`int`	An integer has no fractional part.
`-6`	`int`	Integers can be negative.
`0`	`int`	Zero is an integer.
`0.5`	`float`	A number with a fractional part has type `float`.
`1.0`	`float`	An integer with a fractional part .0 has type `float`.
`1E6`	`float`	A number in exponential notation: 1×10^6 or 1000000. Numbers in exponential notation always have type `float`.
`2.96E-2`	`float`	Negative exponent: $2.96 \times 10^{-2} = 2.96 / 100 = 0.0296$
`100,000`		**Error:** Do not use a comma as a decimal separator.
`3 1/2`		**Error:** Do not use fractions; use decimal notation: 3.5.

A Warning…

- Since the data type is associated with the value and not the variable:
  - A variable can be assigned different values at different places in a program

```
taxRate = 5                    # an int
```

Then later…

```
taxRate = 5.5                  # a float
```

And then

```
taxRate = "Non- taxable"       # a string
```

- If you use a variable and it has an unexpected type an error will occur in your program

Naming variables

- Variable names should describe the purpose of the variable
  `canVolume` is better than `cv`
- Use These Simple Rules
  1. Variable names must start with a letter or the underscore (_) character
     - Continue with letters (upper or lower case), digits or the underscore
  2. You cannot use other symbols (? or %...) and spaces are not permitted
  3. Separate words with *camelCase* notation
     - Use upper case letters to signify word boundaries
  4. Names are case sensitive
     - `canVolume` and `canvolume` are different names
  5. Don't use reserved Python words (see Appendix C, pages A6 and A7). For example, `list`, `import`, `print`, `len`, `max`, `min`, …etc.

Code Academy

Table 2: Variable Names in Python

Table 2 Variable Names in Python	
Variable Name	**Comment**
`canVolume1`	Variable names consist of letters, numbers, and the underscore character.
`x`	In mathematics, you use short variable names such as *x* or *y*. This is legal in Python, but not very common, because it can make programs harder to understand (see Programming Tip 2.1 on page 36).
⚠ `CanVolume`	**Caution:** Variable names are case sensitive. This variable name is different from `canVolume`, and it violates the convention that variable names should start with a lowercase letter.
🚫 `6pack`	**Error:** Variable names cannot start with a number.
🚫 `can volume`	**Error:** Variable names cannot contain spaces.
🚫 `class`	**Error:** You cannot use a reserved word as a variable name.
🚫 `ltr/fl.oz`	**Error:** You cannot use symbols such as / or.

More Examples

- Mark the following variables as either valid or invalid

`theDog`	`all-In-One`	✗
`recycling`	`DVD_ROM`	✓
`elizabeth_the_2nd`	`2morrow`	✗
`page#`	`age_Of_Cat`	✓
`taxRateY2K`	`age Of Horse`	✗
`2000TaxRate`	`print`	✗

Code Academy

Programming Tip: Use Descriptive Variable Names

- Choose descriptive variable names
- Which variable name is more self descriptive?

```
canVolume = 0.35
```

```
cv = 0.355
```

- This is particularly important when programs are written by more than one person.

Constants

Code Academy

- In Python a **constant** is a variable whose value *should not* be changed after it is assigned an initial value.
  - It is a good practice to use all caps when naming constants

```
BOTTLE_VOLUME = 2.0
```

- It is good style to use named constants to explain numerical values to be used in calculations
  - Which is clearer?

```
totalVolume = bottles * 2
```

```
totalVolume = bottles * BOTTLE_VOLUME
```

  - A programmer reading the first statement may not understand the significance of the "2"

- Python will let you change the value of a **constant**
  - Just because you can do it, doesn't mean you should do it

Constants: Naming & Style

- It is customary to use all UPPER_CASE letters for constants to distinguish them from variables.

  - It is a nice visual way cue

```
BOTTLE_VOLUME = 2    # Constant
MAX_SIZE = 100       # Constant
taxRate = 5          # Variable
```

Code Academy

Python comments

- Use **comments** at the beginning of each program, and to clarify details of the code
- Comments are a courtesy to others and a way to document your thinking

  - Comments to add explanations for humans who read your code.
- The compiler ignores comments.

Code Academy

Commenting Code

```
 9 ##
10 #  This program computes the volume (in liters) of a six-pack of soda
11 #  cans and the total volume of a six-pack and a two-liter bottle
12 #
13
14 # Liters in a 12-ounce can
15 CAN_VOLUME = 0.355
16
17 # Liters in a two-liter bottle.
18 BOTTLE_VOLUME = 2
19
20 # Number of cans per pack.
21 cansPerPack = 6
22
23 # Calculate total volume in the cans.
24 totalVolume = cansPerPack * CAN_VOLUME
25 print("A six-pack of 12-ounce cans contains", totalVolume, "liters.")
26
27 # Calculate total volume in the cans and a 2-liter bottle.
28 totalVolume = totalVolume + BOTTLE_VOLUME
29 print("A six-pack and a two-liter bottle contain", totalVolume, "liters.")
```

Code Academy

Undefined Variables

- You must define a variable before you use it: (i.e. it must be defined somewhere above the line of code where you first use the variable)

```
In [77]: canVolume = 12 * literPerOunce
    ...: literPerOunce = 0.0296
    ...: print(canVolume)
Traceback (most recent call last):

  File "<ipython-input-77-c38d74c41cd0>", line 1, in <module>
    canVolume = 12 * literPerOunce

NameError: name 'literPerOunce' is not defined
```

- The correct order for the statements is:

```
In [78]: literPerOunce = 0.0296
    ...: canVolume = 12 * literPerOunce
    ...: print(canVolume)
0.3552
```

Summary: variables

- A **variable** is a storage location with a name.
- When defining a variable, you must specify an **initial value**.
- By convention, variable names should start with a lower case letter.
- An **assignment** statement stores a new value in a variable, replacing the previously stored value.
- Variables whose initial value should not change are typically capitalized by convention.