

PROGRAMMING IN PYTHON I

Comments, Variables, Console



Andreas Schörgenhumer
Institute for Machine Learning

Copyright Statement

This material, no matter whether in printed or electronic form, may be used for personal and non-commercial educational use only. Any reproduction of this material, no matter whether as a whole or in parts, no matter whether in printed or in electronic form, requires explicit prior acceptance of the authors.

Contact

Andreas Schörgenhumer

Institute for Machine Learning
Johannes Kepler University
Altenberger Str. 69
A-4040 Linz

E-Mail: schoergenhumer@ml.jku.at

Write mails only for personal questions

[Institute ML Homepage](#)

A PYTHON PROGRAM



A Python Program

- Python program code is stored in text files
- Standard filename suffix indicating a Python file: `.py`
 - Example filename: `myfile.py`
- Python program code is executed **line by line (from first line to last line)**
- See file `01_comments_variables_console.py` for an example Python file
- We will use Python in version ≥ 3.9

Python Code Execution

- **Expressions** are **evaluated from left to right**

$a + b + c$

- ☐ Is equivalent to $(a + b) + c$

- **Assignments** are **evaluated from right to left**

$x = a + b$

- ☐ Is equivalent to $x = (a + b)$

- Different **operators have different precedence**

$x = a + b / c$

- ☐ Is equivalent to $x = (a + (b / c))$
- ☐ <https://docs.python.org/3/reference/expressions.html#operator-precedence>

Python Style

- Python will not force you to follow a certain style but there are recommendations (as you will see later)
- Adhering to conventions helps to achieve easier maintainability and consistency, and it will aid programmers in understanding other people's code more quickly
- Recommendation details (naming conventions, comment styles, formatting):

<https://www.python.org/dev/peps/pep-0008/>

COMMENTS



Comments

- Parts of the program code which are **not executed**
- Have no effect on the behavior of the program
- Start with **hashtag character** #
- Used for **documenting code**
- **Good comments will make your life much easier!**

Comments: Examples

- The following line only contains a comment:

```
# This is a comment
```

- The following line contains an assignment operation, followed by a comment:

```
var = "hello" # This is a comment
```

For execution, this it is equivalent to:

```
var = "hello"
```

DATA TYPES



Data Types

- We can use a group of bits to encode a value
- There are different ways to encode values as bits (=data types)
- The more bits per value we use, the more unique values we can encode (typically multitudes of bytes)
- Our main data types will be
 - bool** Boolean – Either False or True
 - int** Integer – Integral numbers
 - float** Float – Floating point numbers
 - str** String – (String of) characters

Data Types: Boolean

- Often, we want to check conditions → for convenience, data type **boolean**
- Can only store two values: **False** or **True**
- In Python, this is actually an integer (see next slide):
 - 0 = False
 - 1 = True

Data Types: Integer

- **Integer** data type assigns one bit pattern to one value
- **Precise**, there is no information lost
- **Only integral numbers** in certain range¹

2 bits	decoding	value
0 0	→	0
0 1	→	1
1 0	→	2
1 1	→	3

¹In Python, integers are actually not limited by a certain fixed size (e.g., 32 bits) but are variable-length objects of arbitrary size.

Data Types: Float

- **Float** data type uses the formula

$$value = significand \times base^{exponent}$$

where *significand* and *exponent* are integers extracted from the bit pattern, and *base* is fixed (e.g., 2)

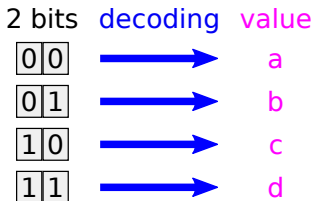
- **Not precise** because values are **approximated**
- Allows for **floating point numbers** in very large range²

2 bits	decoding	value		
<table border="1"><tr><td>0</td><td>0</td></tr></table>	0	0	→	0.0
0	0			
<table border="1"><tr><td>0</td><td>1</td></tr></table>	0	1	→	0.5
0	1			
<table border="1"><tr><td>1</td><td>0</td></tr></table>	1	0	→	1.0
1	0			
<table border="1"><tr><td>1</td><td>1</td></tr></table>	1	1	→	16.0
1	1			

²In Python, you typically have **double-precision floats** (64 bits), which can represent numbers between about $\pm 2.2 \cdot 10^{-308}$ and $\pm 1.8 \cdot 10^{308}$.

Data Types: String

- **Character** data type assigns one bit pattern (typically a byte) to one **character/letter**
- Such characters are concatenated, which gives data type **string** (we will see more about this later)³
- Different encoding formats: UTF-8, ASCII, ...



³In Python, a single character is also represented as a string.

VARIABLES



Variables

- A **variable** is something that **can hold a changeable value**
- We can **store (assign), access and modify** the information in the variable
- Example in Python code:
 - Assign integer 5 to variable `var`:
`var = 5`
 - Modify content of existing variable `var`:
`var = 6`
 - Assigning to variable `var2` by accessing `var`:
`var2 = var - 5`

Variables: Realization

- We can use bits to store, access and modify information
- You can think of a **variable** as a **named set of bits that hold a value**
- A variable has a **symbolic name** (variable name)
- A variable is a **storage location** (memory address)
 - We have to know which and how many bits are used
- A variable holds some **value**
 - We have to know the data type to encode/decode the value

Variables: Static and Dynamic Typing

■ Static typing:

- Data type of variable is known at **compile time**
- **Variable itself is associated with data type**
- Example: In Java, a variable uses a fixed data type that has to be set when defining the variable

```
int var = 5;
```

■ Dynamic typing:

- Variable data type is determined during **run time**
- **Data type is associated with value** itself, not with variable
- Example: In Python, a variable is a reference to an object (value) which itself stores the information about the type

```
var = 5 # 5 is an integer object
```

Variables in Python

- Variables in Python are just **references to objects** stored and generated automatically in the background, i.e., they are essentially only names or identifiers that reference objects – they themselves do not store any information
- These actual **objects** hold information on **data type, number of bits used and if the object is used**
- A (64bit CPython) object consists of $16 + x$ bytes:
 - ☐ type pointer: 8 bytes
 - ☐ reference count: 8 bytes
 - ☐ object bytes: x bytes
- What is an object in Python?

Variables in Python

- Variables in Python are just **references to objects** stored and generated automatically in the background, i.e., they are essentially only names or identifiers that reference objects – they themselves do not store any information
- These actual **objects** hold information on **data type, number of bits used and if the object is used**
- A (64bit CPython) object consists of $16 + x$ bytes:
 - ☐ type pointer: 8 bytes
 - ☐ reference count: 8 bytes
 - ☐ object bytes: x bytes
- What is an object in Python? **Everything!**

Variables in Python: Consequences

■ Consequences:

- 16 bytes **overhead when using variables**
- Variables **not bound to single data type** (change data type by changing the object it references to)
- **Memory** (=bits) of variables that are no longer used are **automatically freed by garbage collector**
- If multiple variables are holding the same value (=referencing the same object), this **object is not duplicated but reused**

- We can still write memory-efficient code by using Python packages such as `numpy`

Using Variables in Python

- Assigning to a variable that does not exist yet, creates this variable
- Variable names must start with characters that are not digits and not operators
- Variable names are case sensitive
- Variable names are by convention in lower case format and words are separated with an underscore. Example:
`my_variable_name`
- Certain names (**keywords**) are reserved and cannot be used, e.g., `if`, `while`, `for`, `def`, `class`, etc.⁴

⁴[https:](https://docs.python.org/3/reference/lexical_analysis.html#keywords)

[//docs.python.org/3/reference/lexical_analysis.html#keywords](https://docs.python.org/3/reference/lexical_analysis.html#keywords)

Using Variables in Python: Example

- Consider the following Python code:⁵

```
x = 42
```

```
y = x
```

How often is 42 stored in memory?

⁵For a longer discussion of this example, click [here](#)

Using Variables in Python: Example

- Consider the following Python code:⁵

```
x = 42  
y = x
```

How often is 42 stored in memory? Once! `x` and `y` refer to the same integer object with value 42.

- Next we do:

```
y = 3
```

What is the value of `x` now?

⁵For a longer discussion of this example, click [here](#)

Using Variables in Python: Example

- Consider the following Python code:⁵

```
x = 42  
y = x
```

How often is 42 stored in memory? Once! `x` and `y` refer to the same integer object with value 42.

- Next we do:

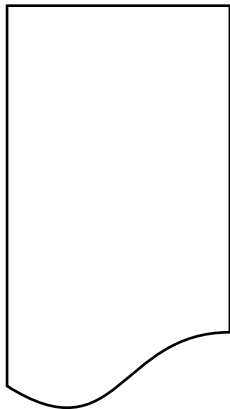
```
y = 3
```

What is the value of `x` now? Still 42! If a value is assigned to a variable, it refers to a new object, i.e., it does not overwrite the object it referred to before the assignment.

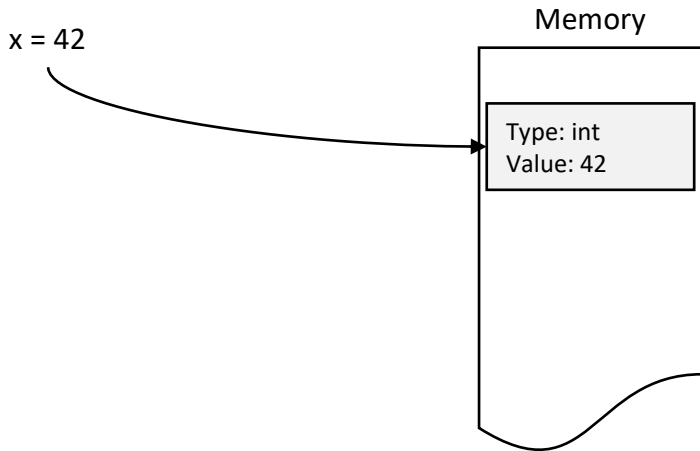
⁵For a longer discussion of this example, click [here](#)

Example Visualization

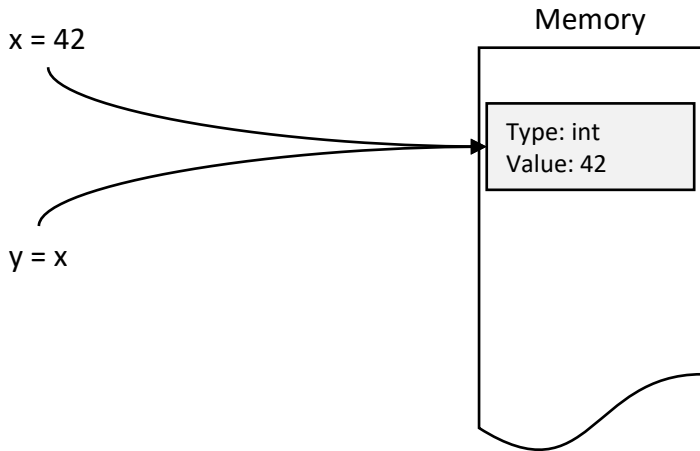
Memory



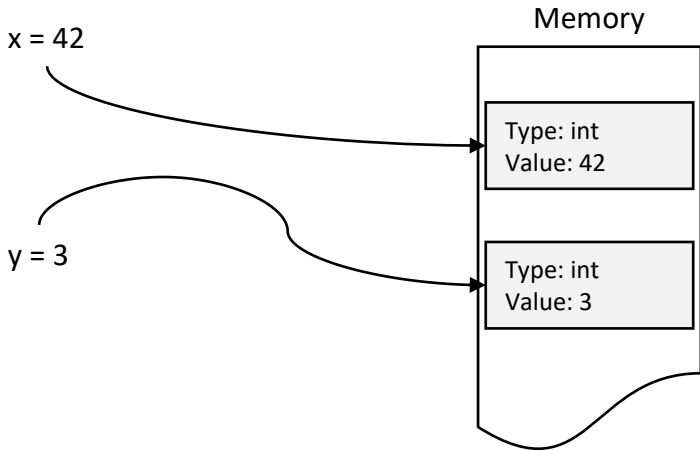
Example Visualization



Example Visualization



Example Visualization



CONSOLE



Console Input and Output

- The **console** is the primary and default **input** and **output device**
- Output (printing something to the console):

```
print("Hello, World!") # Prints "Hello, World!"  
print(123) # Prints "123"
```

- Input (reading user input from the console):

```
var = input("Please enter something: ")
```

The read input will be a string, so data type conversion must be done manually → see the accompanying code file for more details