PROGRAMMING IN PYTHON I

Comments, Variables, Console

Andreas Schörgenhumer
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

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

- Python program code is stored in text files
- Standard filename suffix indicating a Python file: `.py`
  - Example filename: `myfile.py`
- See file `01_comments_variables_console.py` for an example Python file
- We will use Python in version $\geq$ 3.9

Python Code Execution

- Python program code is executed **line by line (from first line to last line)**
- **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/`
- If you follow the style presented in the lecture/exercise, you will be on the right track

Comments

- Parts of the program code which are **not executed**
- Used for **documenting code**
- Start with **hashtag character** #
- Have no effect on the behavior of the program
- **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

- We can use a group of bits to encode a value
- There are different ways to encode values as bits (=**data types**)
- Our main data types will be

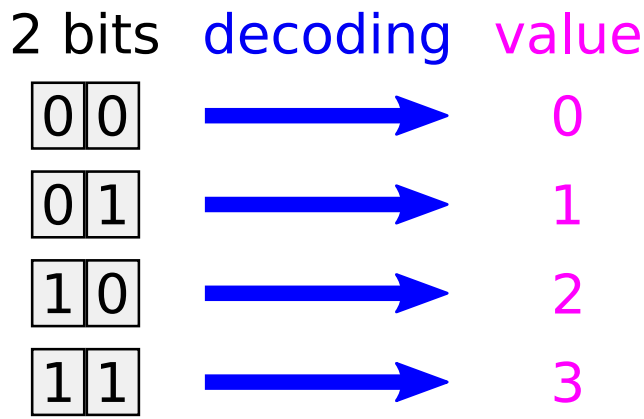
bool	Boolean – Either False or True
int	Integer – Integral numbers
float	Float – Floating point numbers
str	String – (String of) characters

Data Types: Boolean (Example: `var = True`)

- Often, we want to check conditions $\rightarrow$ 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 (Example: `var = 3`)

- **Integer** data type assigns one bit pattern to one value
- **Precise**, there is no information lost
- **Only integral numbers** in certain range[1]



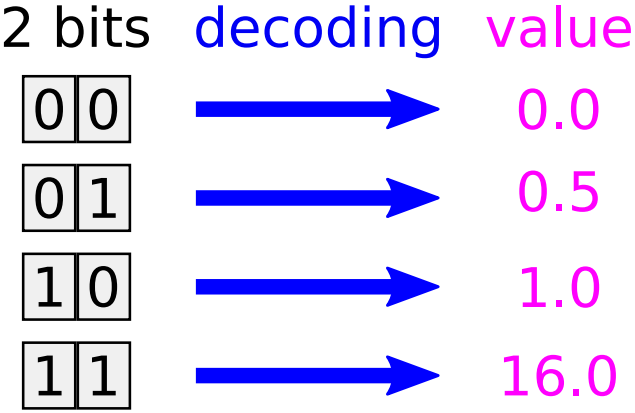
[1] In Python, integers are actually not limited by a certain fixed size (e.g., 32 bits) but are variable-length objects of arbitrary size.

Data Types: Float (Example: `var = 0.5`)

- **Float** data type uses the formula

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

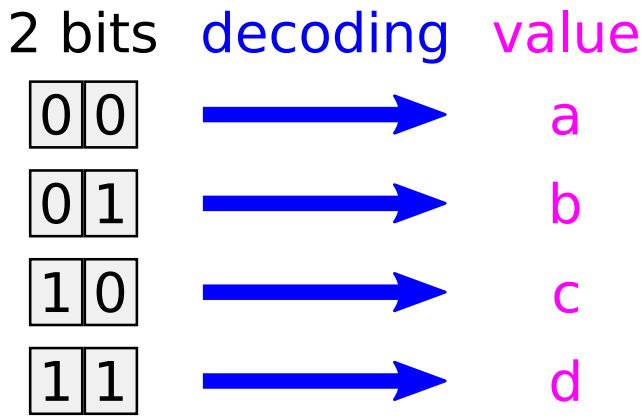
  where $significand$ and $exponent$ are integers extracted from the bit pattern, and $base$ is fixed (e.g., 2)
- **Not precise** because values are approximated
- Allows for **floating point numbers** in very large range[2]



[2] In Python, we 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 (Example: `var = "a"`)

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



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

Variables

- A **variable** is something that **can hold a changeable value** and has a **name** for accessing this value
- We can **store (assign), access and modify** the information in the variable
- Example in Python:
  - Assign integer `5` to variable `var`:

    ```
    var = 5
    ```

  - Modify content of existing variable `var`:

    ```
    var = 6
    ```

  - Assigning to variable `var2` by accessing/reading `var`:

    ```
    var2 = var - 5
    ```

- How do we know what type of data is stored in a variable?

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 that are generated, stored and managed automatically in the background, i.e., they are essentially only **names** or **identifiers** that are **bound to objects** – they themselves do not store any information
- These actual **objects** hold information on the **data type** and what kind of data is stored in your memory
- Example: `var = 5`
  - Variable/name/identifier `var` is bound to `5`
  - `5` is a Python integer object with numerical value 5
- What is an object in Python? **Everything!**

Variables in Python

- Assigning to a variable that does not exist yet, creates this variable (name is bound to the object)
- 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://docs.python.org/3/reference/lexical_analysis.html#keywords

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.

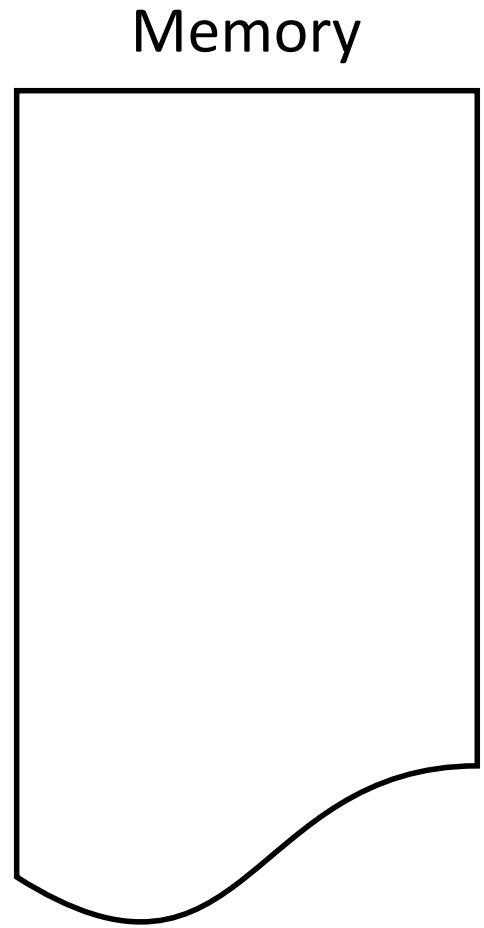
- Next we do:

```
y = 3
```

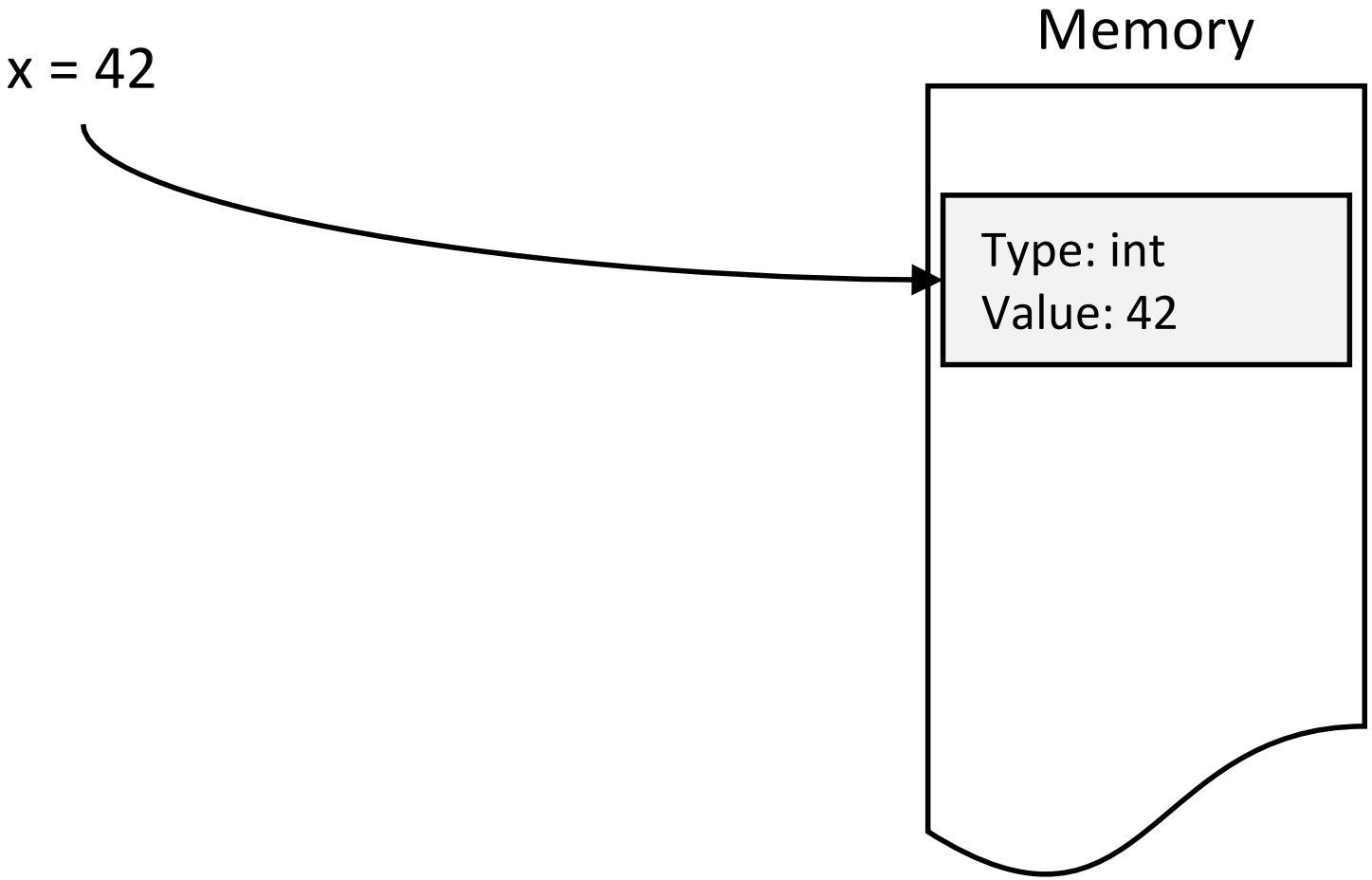
  What is the value of `x` now? Still 42! If a value is assigned to a variable, it refers to a new object, i.e., it does not overwrite the object it referred to before the assignment.

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

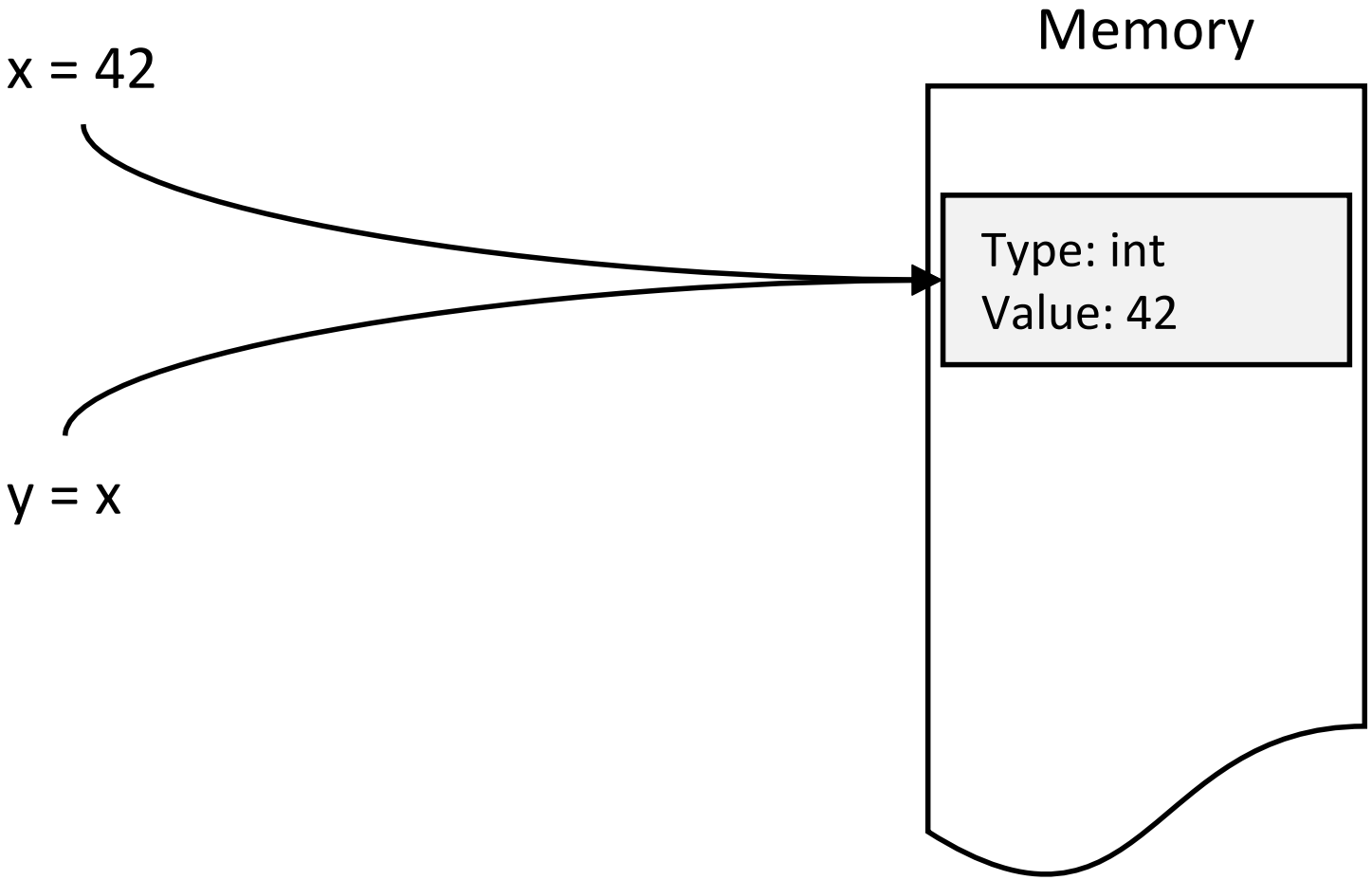
Example Visualization



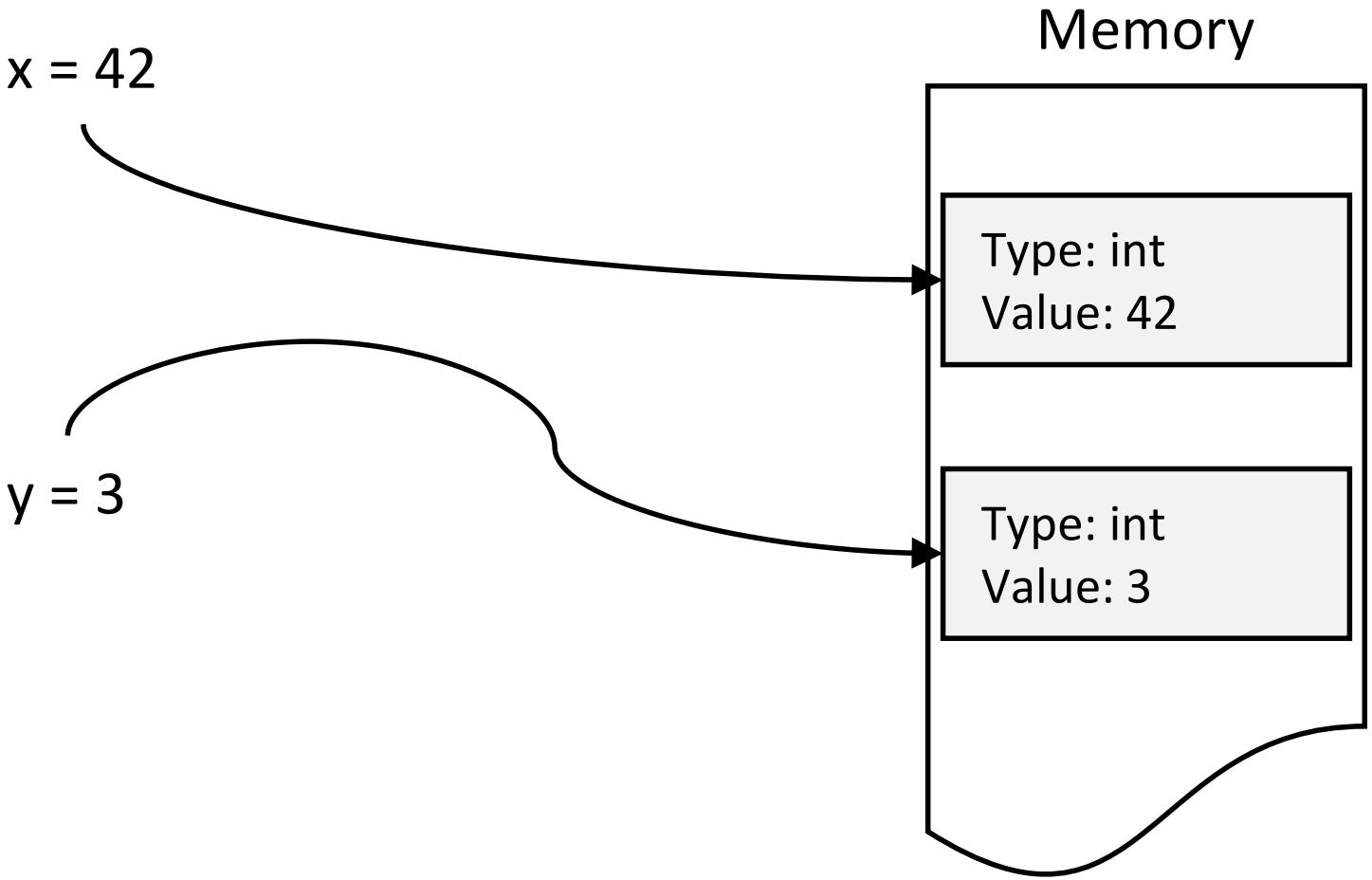
Example Visualization



Example Visualization



Example Visualization



Console Input and Output

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

```
print("Hello, World!")
print(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 $\rightarrow$ see the accompanying code file for more details