ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Python in practice

Lesson 1: Introduction

Semester 2021/22/2

Subject matter

1. Introduction
2. Variables, operators
3. Sequence, selection, iteration
4. Programming theses
5. Strings
6. Regular expressions
7. Files
8. Object-oriented programming
9. Multithread applications
10. GUI with tkinter
11. Communications
12. SQL basics
13. SQL queries
14. Example application

Requirements for the completion of the subject

- A 10-minute-long test from the theoretical lesson on every week.
- A 30-minute-long programming assignment on every week.
- Both tasks above must have at least 60% passed. If one of the above is failed, there is only one opportunity to repeat the task but in a merged form.
- The performance will be evaluated according to the following table :

0-59%	1	✗
60-69%	2	✓
70-79%	3	
80-89%	4	
90-100%	5	

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Feel free to ask your questions

Sándor Tamás	**sandor.tamas@uni-obuda.hu**
Borsos Döníz	**borsos.doniz@uni-obuda.hu**
Fodor Attila	**fodor.attila@stud.uni-obuda.hu**
Héjja Bertold János	**hejja.bertold@stud.uni-obuda.hu**

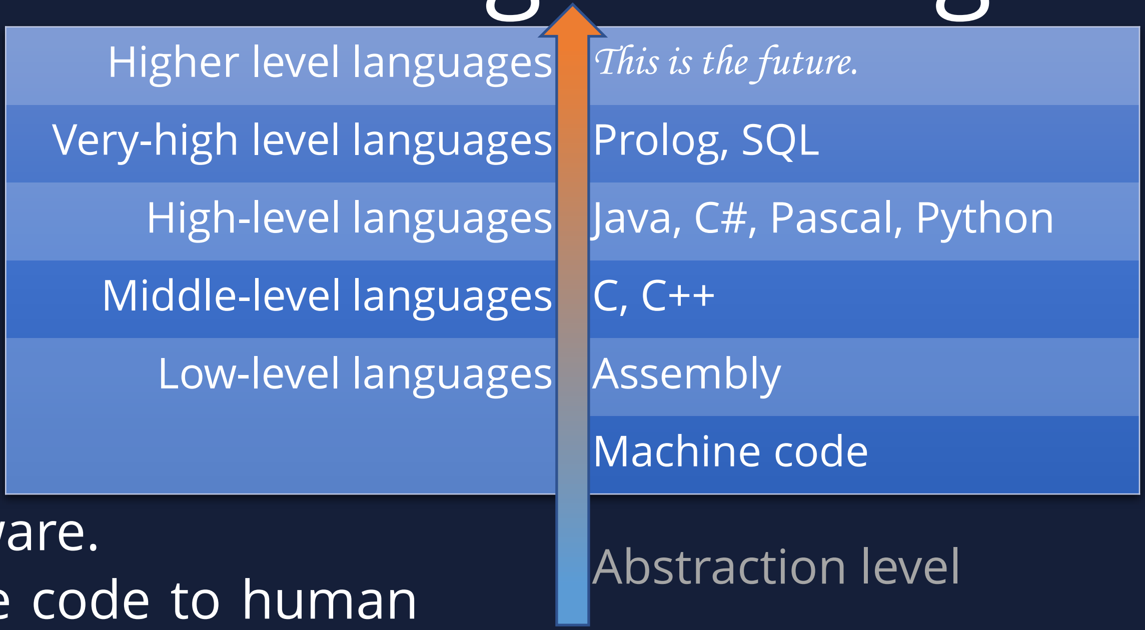
ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Python in practice

Basics of Programming

Basics of Programming

- Computer: black box
  - INPUT → OUTPUT
- Hardware: physical parts of a computer
- Software: controls the hardware
- Firmware, operating system
- A software can run other software.
- Software can control or create new software.
  - Abstract the language from machine code to human close languages
  - While programming in Python, we will use English keywords.
- Abstraction level
  - Higher: worst performance, easier to use
  - Lower: better performance, require more knowledge
- Source code: written with a programming language

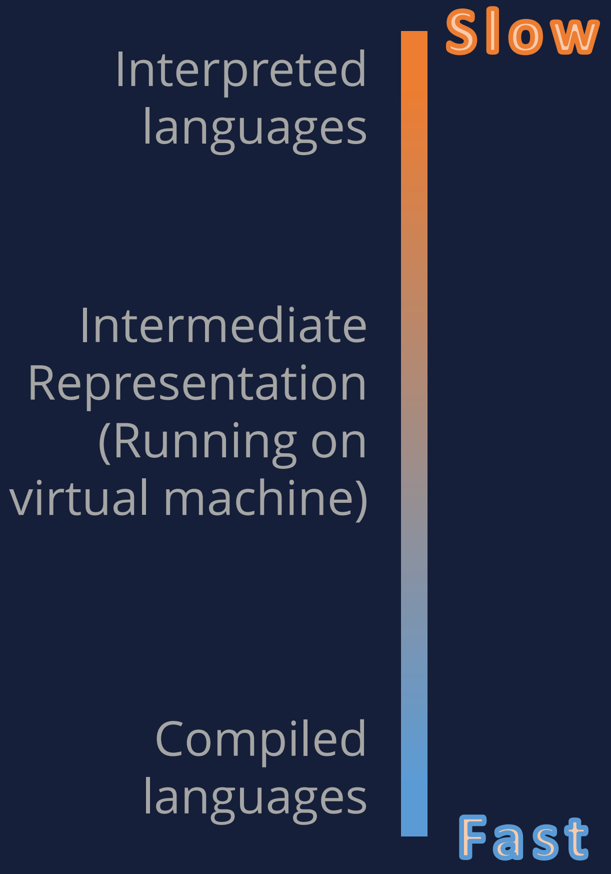


Higher level languages	*This is the future.*
Very-high level languages	Prolog, SQL
High-level languages	Java, C#, Pascal, Python
Middle-level languages	C, C++
Low-level languages	Assembly
	Machine code

Basics of Programming

- Compiler: translates source code into intermediate or machine code
  - Used by C-like languages (C++, C#, Java) etc.
- Interpreter: evaluates the program line by line
  - Used by **Python**, JavaScript etc.

- IDE (Integrated Development Environment):
  - Used for creating programs.
  - Useful tools (toolchain)
    - Syntax highlighting
    - Autocompletion
    - Error detection
    - CLI (Console Line Interface)
  - Example: **PyCharm**, Visual Studio



Slow

Interpreted languages

Intermediate Representation (Running on virtual machine)

Compiled languages

Fast

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Naming conventions

- Used while creating labels
- Makes the code more readable
- Shows the label’s function.
- A-Z; a-z; 0-9; _ (cannot start with a number !!)

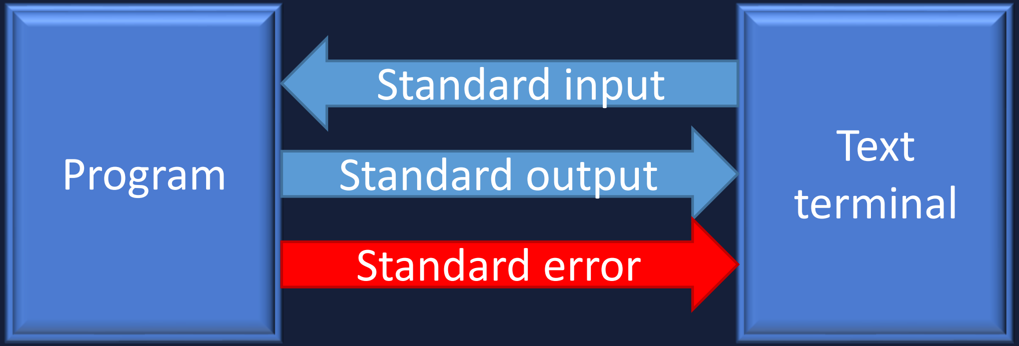
- Restricted keywords (35 pcs)
  - https://realpython.com/python-keywords/
  -
    ```
    False  await  else   import pass   None   break  except in     raise
     True  class  finally       is     return and    continue      for
     lambda try   as     def    from   nonlocal      while  assert del
     global not   with   async  elif   if     or     yield
    ```
- snake_case
- https://www.python.org/dev/peps/pep-0008/

Standard streams

- Stream: sequence of data elements
  - Can be thought of as a conveyor belt.
  - Communication channels
  - Program ↔ environment (CLI)

- 3 types:
  - `sys.stdin`
  - `sys.stdout`
  - `sys.stderr`
- Input: the program reads with `input()`
- Output: `print()` and the prompt of `input()`
- Error: interpreter’s own prompts and its error messages

- Streams can be redirected (we will use it later)


Program
Standard input
Standard output
Standard error
Text terminal