

# COMP110: Principles of Computing

## 4: LaTeX



# Worksheet 3



# Converting types



# Weak vs strong typing

# Weak vs strong typing

- In **weakly typed** languages, a variable can hold a value of any type

# Weak vs strong typing

- ▶ In **weakly typed** languages, a variable can hold a value of any type
  - ▶ Examples: Python, JavaScript

# Weak vs strong typing

- ▶ In **weakly typed** languages, a variable can hold a value of any type
  - ▶ Examples: Python, JavaScript
- ▶ In **strongly typed** languages, the type of a variable must be **declared**

# Weak vs strong typing

- ▶ In **weakly typed** languages, a variable can hold a value of any type
  - ▶ Examples: Python, JavaScript
- ▶ In **strongly typed** languages, the type of a variable must be **declared**
  - ▶ Examples: C#, C++, Java



# Weak typing (example in Python)

```
x = 7  
# Now x has type int  
  
x = "hello"  
# Now x has type string
```

# Strong typing (example in C#)

```
int x = 7;  
// x is declared with type int  
  
x = "hello";  
// Compile error: cannot convert type "string" to "int"
```

# Type casting

# Type casting

- It is often useful to **cast**, or **convert**, a value from one type to another

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → 17.0

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → `17.0`
  - ▶ `int(3.14)` → `3`

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → `17.0`
  - ▶ `int(3.14)` → `3`
  - ▶ `str(3.14)` → `"3.14"`



# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → `17.0`
  - ▶ `int(3.14)` → `3`
  - ▶ `str(3.14)` → `"3.14"`
  - ▶ `str(1 + 1 == 2)` → `"True"`

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → `17.0`
  - ▶ `int(3.14)` → `3`
  - ▶ `str(3.14)` → `"3.14"`
  - ▶ `str(1 + 1 == 2)` → `"True"`
  - ▶ `int("123")` → `123`

# Type casting

- ▶ It is often useful to **cast**, or **convert**, a value from one type to another
- ▶ In Python, this is done by calling the type as if it were a function
  - ▶ `float(17)` → `17.0`
  - ▶ `int(3.14)` → `3`
  - ▶ `str(3.14)` → `"3.14"`
  - ▶ `str(1 + 1 == 2)` → `"True"`
  - ▶ `int("123")` → `123`
  - ▶ `int("five")` gives an error

# Operations on types

# Operations on types

- ▶ Certain operations can only be done on certain types of values

# Operations on types

- ▶ Certain operations can only be done on certain types of values
- ▶ Can add two ints:  $2 + 3 \rightarrow 5$

# Operations on types

- ▶ Certain operations can only be done on certain types of values
- ▶ Can add two ints:  $2 + 3 \rightarrow 5$
- ▶ Can add int and float:  $2 + 3.1 \rightarrow 5.1$

# Operations on types

- ▶ Certain operations can only be done on certain types of values
- ▶ Can add two ints:  $2 + 3 \rightarrow 5$
- ▶ Can add int and float:  $2 + 3.1 \rightarrow 5.1$
- ▶ Can add two strings: `"COMP" + "110" → "COMP110"`



# Operations on types

- ▶ Certain operations can only be done on certain types of values
- ▶ Can add two ints:  $2 + 3 \rightarrow 5$
- ▶ Can add int and float:  $2 + 3.1 \rightarrow 5.1$
- ▶ Can add two strings: `"COMP" + "110" → "COMP110"`
- ▶ Can't add string and int: `"COMP" + 110 → error`

# Implicit type conversion

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**
- ▶ Some languages (not Python) can perform **implicit** type casts to make operations work

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**
- ▶ Some languages (not Python) can perform **implicit** type casts to make operations work
- ▶ Sometimes called **type coercion**

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**
- ▶ Some languages (not Python) can perform **implicit** type casts to make operations work
- ▶ Sometimes called **type coercion**
- ▶ E.g. in JavaScript, `"COMP" + 110`  $\rightarrow$  `"COMP110"`

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**
- ▶ Some languages (not Python) can perform **implicit** type casts to make operations work
- ▶ Sometimes called **type coercion**
- ▶ E.g. in JavaScript, `"COMP" + 110`  $\rightarrow$  `"COMP110"`
- ▶ The integer `110` is implicitly converted to a string `"110"` to make the addition work

# Implicit type conversion

- ▶ The type casts we saw a few slides ago are **explicit**
- ▶ Some languages (not Python) can perform **implicit** type casts to make operations work
- ▶ Sometimes called **type coercion**
- ▶ E.g. in JavaScript, `"COMP" + 110`  $\rightarrow$  `"COMP110"`
- ▶ The integer `110` is implicitly converted to a string `"110"` to make the addition work
- ▶ Equivalent in Python with explicit casts:  
`"COMP" + str(110)`



# Dangers of implicit type conversion

# Dangers of implicit type conversion

- ▶ Rules for implicit type conversion can sometimes be confusing

# Dangers of implicit type conversion

- ▶ Rules for implicit type conversion can sometimes be confusing
- ▶ E.g. in JavaScript:

# Dangers of implicit type conversion

- ▶ Rules for implicit type conversion can sometimes be confusing
- ▶ E.g. in JavaScript:

▶ `"5" + 3` → `"53"`

# Dangers of implicit type conversion

- ▶ Rules for implicit type conversion can sometimes be confusing
- ▶ E.g. in JavaScript:
  - ▶ `"5" + 3` → `"53"`
  - ▶ `"5" - 3` → `2`

# Introducing LaTeX



# What is LaTeX?

# What is LaTeX?

- ▶ A **typesetting** system



# What is LaTeX?

- ▶ A **typesetting** system
- ▶ A **markup language** (like HTML or Markdown)

# What is LaTeX?

- ▶ A **typesetting** system
- ▶ A **markup language** (like HTML or Markdown)
- ▶ **Not** a WYSIWYG system

# Why LaTeX?

# Why LaTeX?

- ▶ Plain text format

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting



# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS
  - ▶ Unlike most WYSIWYG systems

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS
  - ▶ Unlike most WYSIWYG systems
- ▶ Produces professional-looking papers, reports, theses, books, slideshows, ...

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS
  - ▶ Unlike most WYSIWYG systems
- ▶ Produces professional-looking papers, reports, theses, books, slideshows, ...
- ▶ Excellent facilities for typesetting mathematical equations, pseudocode, source code listings etc.

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS
  - ▶ Unlike most WYSIWYG systems
- ▶ Produces professional-looking papers, reports, theses, books, slideshows, ...
- ▶ Excellent facilities for typesetting mathematical equations, pseudocode, source code listings etc.
- ▶ Automatically handles cross-referencing of sections, figures etc.

# Why LaTeX?

- ▶ Plain text format
  - ▶ Can use any text editor
  - ▶ Can use version control (e.g. Git)
  - ▶ Can use online editors (e.g. Overleaf)
- ▶ Separates content from formatting
  - ▶ Similar to HTML and CSS
  - ▶ Unlike most WYSIWYG systems
- ▶ Produces professional-looking papers, reports, theses, books, slideshows, ...
- ▶ Excellent facilities for typesetting mathematical equations, pseudocode, source code listings etc.
- ▶ Automatically handles cross-referencing of sections, figures etc.
- ▶ Automatic tools for managing bibliographies (BibTeX)

# Getting LaTeX

# Getting LaTeX

- ▶ LaTeX is **free open source software**



# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:

# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:
  - ▶ Several **executables** (pdflatex, bibtex, makeindex, ...)

# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:
  - ▶ Several **executables** (pdflatex, bibtex, makeindex, ...)
  - ▶ A large library of **packages**

# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:
  - ▶ Several **executables** (pdflatex, bibtex, makeindex, ...)
  - ▶ A large library of **packages**
  - ▶ An **integrated development environment (IDE)** (optional)

# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:
  - ▶ Several **executables** (pdflatex, bibtex, makeindex, ...)
  - ▶ A large library of **packages**
  - ▶ An **integrated development environment (IDE)** (optional)
- ▶ Distributions available for all major OSes
  - ▶ Windows: MikTeX
  - ▶ MacOS: MacTeX
  - ▶ Linux: TeXLive

# Getting LaTeX

- ▶ LaTeX is **free open source software**
- ▶ Consists of:
  - ▶ Several **executables** (pdflatex, bibtex, makeindex, ...)
  - ▶ A large library of **packages**
  - ▶ An **integrated development environment (IDE)** (optional)
- ▶ Distributions available for all major OSes
  - ▶ Windows: MikTeX
  - ▶ MacOS: MacTeX
  - ▶ Linux: TeXLive
- ▶ Online services e.g. Overleaf (should also work on iPad / Android)

# Workshop Activity

- ▶ Go to <https://www.overleaf.com> and sign up for a free account
- ▶ Go to <https://www.latex-tutorial.com/tutorials/> and work through the tutorials
- ▶ Please prioritise the following tutorials (look at the others afterwards if you have time):
  - ▶ 01 Your first document
  - ▶ 02 Document structure (sections and paragraphs)
  - ▶ 03 Packages
  - ▶ 05 Adding pictures
  - ▶ 07 Bibliography
  - ▶ 13 Source code highlighting
  - ▶ 16 Hyperlinks
  - ▶ 17 Lists