

Course Software

Rob Hackman

Fall 2025

University of Alberta

Table of Contents

What are programs?

Operating Systems

Using The Shell and File Systems

Installing Python

Definition of a program

A program is a sequence of instructions for your computer to execute.

Definition of a program

A program is a sequence of instructions for your computer to execute.

- When a program is loaded to be executed the running instance of the program is called a *process*

Definition of a program

A program is a sequence of instructions for your computer to execute.

- When a program is loaded to be executed the running instance of the program is called a *process*
- Programs are typically written in high-level programming languages which our computers do not understand

Definition of a program

A program is a sequence of instructions for your computer to execute.

- When a program is loaded to be executed the running instance of the program is called a *process*
- Programs are typically written in high-level programming languages which our computers do not understand
- In order for a program written in a high-level language to be executed by a computer the equivalent instructions in that computers *machine language* must be executed

Machine code

In order for programs to be executed by your computer they must be written in the language your physical hardware understands — this is called your computers *machine code*.

Machine code

In order for programs to be executed by your computer they must be written in the language your physical hardware understands — this is called your computers *machine code*.

Machine code is not very human-readable. For this reason we created high-level programming languages, like Python that we will learn, to help us write programs more easily.

Machine code

In order for programs to be executed by your computer they must be written in the language your physical hardware understands — this is called your computers *machine code*.

Machine code is not very human-readable. For this reason we created high-level programming languages, like Python that we will learn, to help us write programs more easily.

In order to execute high-level programs on our computer, programmers write special programs to ultimately run the machine code that produces the behaviour desired by the high-level program you've written.

A compiler is one such program that facilitates the execution of a high-level program by our computers. A compiler takes your written program and creates a new translated copy of it in the machine code your computer understands. That new copy is the program that can be executed on your computer.

A compiler is one such program that facilitates the execution of a high-level program by our computers. A compiler takes your written program and creates a new translated copy of it in the machine code your computer understands. That new copy is the program that can be executed on your computer.

Once the *compiled* copy of your program has been created it can be executed as many times as you like — you now have a copy of your program that is runnable by your computer.

Compilers

A compiler is one such program that facilitates the execution of a high-level program by our computers. A compiler takes your written program and creates a new translated copy of it in the machine code your computer understands. That new copy is the program that can be executed on your computer.

Once the *compiled* copy of your program has been created it can be executed as many times as you like — you now have a copy of your program that is runnable by your computer.

You will learn more about compiled languages in CMPUT 275 when you learn C, a compiled language.

An interpreter is like a live translator. Instead of your program being translated into a program the computer can understand, the interpreter is the program that runs. The translator reads your code and performs the behaviour desired by your code.

An interpreter is like a live translator. Instead of your program being translated into a program the computer can understand, the interpreter is the program that runs. The translator reads your code and performs the behaviour desired by your code.

Since the interpreter is the program is actually running on your computer, your program can only be executed so long as you have the interpreter to run it.

Interpreters

An interpreter is like a live translator. Instead of your program being translated into a program the computer can understand, the interpreter is the program that runs. The translator reads your code and performs the behaviour desired by your code.

Since the interpreter is the program is actually running on your computer, your program can only be executed so long as you have the interpreter to run it.

The language we will learn in this class, Python, is an interpreted language.

The Python interpreter

In order to work in this class we will need to learn how to ask our computer to execute particular programs for us.

Of course, one of the programs we will need to execute will be the Python interpreter, as it is needed to see our programs run!

Table of Contents

What are programs?

Operating Systems

Using The Shell and File Systems

Installing Python

You've likely heard of Mac or Windows before, and likely used a computer that runs on one of these operating systems. But what exactly are the main purposes of an operating system?

You've likely heard of Mac or Windows before, and likely used a computer that runs on one of these operating systems. But what exactly are the main purposes of an operating system?

- Manage and provide interfaces for access to your resources (physical hardware)

You've likely heard of Mac or Windows before, and likely used a computer that runs on one of these operating systems. But what exactly are the main purposes of an operating system?

- Manage and provide interfaces for access to your resources (physical hardware)
- Provide environment to run programs safely and securely

You've likely heard of Mac or Windows before, and likely used a computer that runs on one of these operating systems. But what exactly are the main purposes of an operating system?

- Manage and provide interfaces for access to your resources (physical hardware)
- Provide environment to run programs safely and securely
- Maintain and provide access to files stored on your computer

Asking your operating system to do things

The most common task a user would ask their operating system to perform for them is to execute a program. You are most likely familiar with interacting with your operating system through its *graphical user interface* (GUI), however there is another way to talk to your operating system.

Operating systems also will come with a textual interface for requesting they perform tasks for you. These textual interfaces are called *shells* or *command lines*.

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

- Run a program

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

- Run a program
- Create, read, or delete a file

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

- Run a program
- Create, read, or delete a file
- Terminate a program

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

- Run a program
- Create, read, or delete a file
- Terminate a program

In this course we'll learn the absolute basics to allow us to use a shell to assist us in using the software necessary for this course.

What is a Shell?

A shell is a textual interface to allow users to ask their operating system to perform tasks, some common tasks are

- Run a program
- Create, read, or delete a file
- Terminate a program

In this course we'll learn the absolute basics to allow us to use a shell to assist us in using the software necessary for this course.

We'll learn more about using the shell effectively in CMPUT275!

In this class there is one option each for Mac and Windows users that is officially supported. Officially supported means course staff can offer help should students run into installation or configuration issues.

Students who are familiar with the software used in this class as well as interacting with programs may choose to use whatever solution works best for them, however should they run into issues due to their custom setup course staff is not guaranteed to be able to offer support.

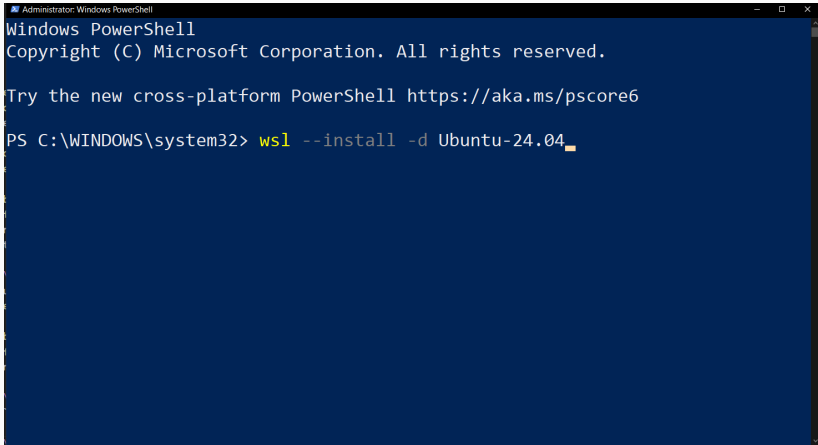
Instructions for Windows users

In CMPUT274 Windows users should install Windows Subsystem for Linux (WSL), which by default will install an emulated version of the Ubuntu operating system for use.

In order to install WSL students should search for “Windows PowerShell” in their start menu. They should then right-click on Windows PowerShell and select “Run as administrator”.

This will pop up a dark blue window which the user can type text into, this is your shell. Then the exact text `wsl --install -d Ubuntu-24.04` should be typed and the enter key hit. This will begin the installation of WSL.

Installing WSL Ubuntu visualization

A screenshot of a Windows PowerShell terminal window. The window title is "Administrator: Windows PowerShell". The text inside the terminal shows the standard PowerShell startup messages: "Windows PowerShell" and "Copyright (C) Microsoft Corporation. All rights reserved." followed by a suggestion to try the new cross-platform PowerShell at "https://aka.ms/pscore6". The prompt "PS C:\WINDOWS\system32>" is followed by the command "wsl --install -d Ubuntu-24.04" with a cursor at the end.

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> wsl --install -d Ubuntu-24.04
```

Accessing your Ubuntu shell

Once you have installed Ubuntu, you should be able to find the program by searching Ubuntu in your start bar. When executing the program you should see a screen similar to the following:

A screenshot of a terminal window. The title bar at the top reads "rob@DESKTOP-5VCTMEP: ~". The terminal content shows a green prompt "rob@DESKTOP-5VCTMEP: ~\$" followed by a blue cursor. The background is black, and the text is green and blue. The window has standard Ubuntu window controls (minimize, maximize, close) in the top right corner.

```
rob@DESKTOP-5VCTMEP: ~$
```

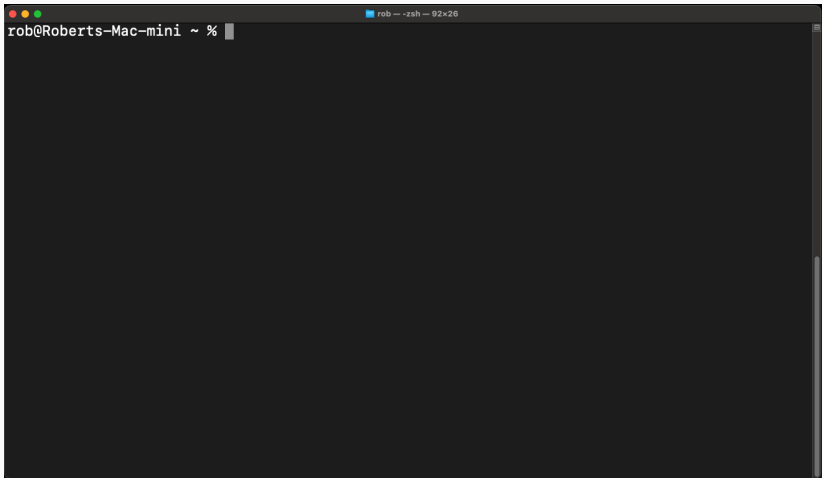

Note on WSL installation of Ubuntu

While students may get by in this course without installing WSL and Ubuntu it is necessary for CMPUT 275.

So, it is in your best interest to install it now and start getting familiar with it throughout the term.

Instructions for Mac users

Mac users already have a shell that is appropriate for CMPUT 274. They simply have to execute the program on their computer called “Terminal”, which should pop up a window that looks like this:



```
rob@Roberts-Mac-mini ~ %
```

Table of Contents

What are programs?

Operating Systems

Using The Shell and File Systems

Installing Python

In order to tackle some of the tasks we will be undertaking this semester, we will need some basic computer literacy to aid us both in using the software we need to, as well provide understanding of topics to come.

In order to tackle some of the tasks we will be undertaking this semester, we will need some basic computer literacy to aid us both in using the software we need to, as well provide understanding of topics to come.

The shell will allow us to execute programs by typing the names of the programs we'd like to execute, followed by additional information we'd like to pass to that program. Each instance of typing a request for the shell to execute a program is called a *command*.

Shell basics

In order to tackle some of the tasks we will be undertaking this semester, we will need some basic computer literacy to aid us both in using the software we need to, as well provide understanding of topics to come.

The shell will allow us to execute programs by typing the names of the programs we'd like to execute, followed by additional information we'd like to pass to that program. Each instance of typing a request for the shell to execute a program is called a *command*.

Programs that we interact with mainly through the shell, as opposed to our graphical interface, are said to have a *command-line interface* (CLI).

Sample command

```
$ cat /usr/share/dict/words
```

Sample command

```
$ cat /usr/share/dict/words
```

- cat is the program name - we are asking Ubuntu to run the cat program

Sample command

```
$ cat /usr/share/dict/words
```

- `cat` is the program name - we are asking Ubuntu to run the `cat` program
- `/usr/share/dict/words` is a *command-line argument*, additional data we provide which the program expects

Sample command

```
$ cat /usr/share/dict/words
```

- `cat` is the program name - we are asking Ubuntu to run the `cat` program
- `/usr/share/dict/words` is a *command-line argument*, additional data we provide which the program expects
 - The operating system passes this command-line argument to the program for use when it executes the program
 - Many programs we will need to execute will expect command-line arguments provided with them

Sample command

```
$ cat /usr/share/dict/words
```

- `cat` is the program name - we are asking Ubuntu to run the `cat` program
- `/usr/share/dict/words` is a *command-line argument*, additional data we provide which the program expects
 - The operating system passes this command-line argument to the program for use when it executes the program
 - Many programs we will need to execute will expect command-line arguments provided with them

The following command would be the same

```
$ cat "/usr/share/dict/words"
```

Sample command

```
$ cat /usr/share/dict/words
```

What does the text `/usr/share/dict/words` represent?

Sample command

```
$ cat /usr/share/dict/words
```

What does the text `/usr/share/dict/words` represent? This is a filepath - which specifies exactly where a file exists in our operating systems file system.

Sample command

```
$ cat /usr/share/dict/words
```

What does the text `/usr/share/dict/words` represent? This is a filepath - which specifies exactly where a file exists in our operating systems file system.

A filepath is like directions telling the operating system where to find a file within our *file system*.

File systems

A *file system* is the portion of your operating system that manages the storage and retrieval of your files.

Mainly file systems consist of two types of files

- *Regular files*
- *Directories* (also known as folders)

Directories are files that do not have any content themselves, rather they simply aggregate other files within themselves to help with organization of your files.

Regular files are all other files, such as pictures, movies, text documents, programs, etc.

```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory


```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory
- Beginning our path with / specifies that we are starting from that root directory - this is called an *absolute path*

Sample command

```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory
- Beginning our path with / specifies that we are starting from that root directory - this is called an *absolute path*
- /usr specifies the file `usr` that is immediately within the root directory

Sample command

```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory
- Beginning our path with / specifies that we are starting from that root directory - this is called an *absolute path*
- /usr specifies the file `usr` that is immediately within the root directory
- /usr/share then specifies the file `share` that is immediately within the `usr` directory

Sample command

```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory
- Beginning our path with / specifies that we are starting from that root directory - this is called an *absolute path*
- /usr specifies the file `usr` that is immediately within the root directory
- /usr/share then specifies the file `share` that is immediately within the `usr` directory
- /usr/share/dict specifies the file `dict` that is immediately within the `share` directory

Sample command

```
$ cat /usr/share/dict/words
```

- In both Ubuntu's and Mac's file system there is one *root* directory - each file resides somewhere within this directory
- Beginning our path with / specifies that we are starting from that root directory - this is called an *absolute path*
- /usr specifies the file usr that is immediately within the root directory
- /usr/share then specifies the file share that is immediately within the usr directory
- /usr/share/dict specifies the file dict that is immediately within the share directory
- /usr/share/dict/words specifies the file words that is immediately within the dict directory

Current working directory

A directory (or folder) is a special type of file that we use to organize other files. In our example on the previous slide `usr`, `share`, and `dict` were all directories.

Current working directory

A directory (or folder) is a special type of file that we use to organize other files. In our example on the previous slide `usr`, `share`, and `dict` were all directories.

Whenever any program, including your shell is running, it has a *current working directory*. For our shell this is particularly important as the commands we execute will take place in our current working directory.

Current working directory

A directory (or folder) is a special type of file that we use to organize other files. In our example on the previous slide `usr`, `share`, and `dict` were all directories.

Whenever any program, including your shell is running, it has a *current working directory*. For our shell this is particularly important as the commands we execute will take place in our current working directory.

- The command `pwd` prints out the current working directory
- The command `cd` changes the current working directory
 - If given no arguments sets the current working directory to the logged in users home directory
 - If given a command line argument sets the current working directory to the one specified by the argument

Absolute and relative paths

The path `/usr/share/dict/words` was called an absolute path because it started from our root directory.

Any path that does not begin with a forward slash is a relative path and is considered relative to the current working directory

Relative paths

- `$ cat foo.txt` here `foo.txt` is a relative path to the file with that name in the current working directory
- `$ cat ./foo.txt` the signifier `./` expands to the current directory in a path - so this example is the same as the above
- `$ cat ../foo.txt` the signifier `../` expands to the directory that contains the current directory in a path, so this would be the final `foo.txt` in the directory which contains our current working directory

If the current working directory is

`/usr/home/rob/cmp274/slides` can you write the absolute paths each that would be the same as the above relative paths?

Knowledge Check: If the current working directory is `/usr/home/rob/slides` then what is the absolute path relative to each of the following?

- `../..`
- `../cheese/bar.txt`
- `.././foo.txt`
- `../../../../blah.txt`

Knowledge Check: If the current working directory is `/usr/home/rob/slides` then what is the absolute path relative to each of the following?

- `../..` \rightarrow `/usr/home`
- `../cheese/bar.txt`
- `.././foo.txt`
- `../../../../blah.txt`

Knowledge Check: If the current working directory is `/usr/home/rob/slides` then what is the absolute path relative to each of the following?

- `../..` \rightarrow `/usr/home`
- `../cheese/bar.txt` \rightarrow `/usr/home/rob/cheese/foo.txt`
- `.././foo.txt`
- `../../../.././../blah.txt`

Knowledge Check: If the current working directory is `/usr/home/rob/slides` then what is the absolute path relative to each of the following?

- `../..` \rightarrow `/usr/home`
- `../cheese/bar.txt` \rightarrow `/usr/home/rob/cheese/foo.txt`
- `.././foo.txt` \rightarrow `/usr/home/rob/foo.txt`
- `../../../../../blah.txt`

Knowledge Check: If the current working directory is `/usr/home/rob/slides` then what is the absolute path relative to each of the following?

- `../..` \rightarrow `/usr/home`
- `../cheese/bar.txt` \rightarrow `/usr/home/rob/cheese/foo.txt`
- `.././foo.txt` \rightarrow `/usr/home/rob/foo.txt`
- `../../../../blah.txt` \rightarrow `/usr/blah.txt`

More shell commands for navigating the file system

In addition to `pwd` and `cd` some useful commands for navigating our file-system are

- `ls` — this command will display all the files in your current directory.

More shell commands for navigating the file system

In addition to `pwd` and `cd` some useful commands for navigating our file-system are

- `mkdir dirName` — this command will make the directory named `dirName` in the current working directory (or whatever path is referred to by the argument `dirName`).

More shell commands for navigating the file system

In addition to `pwd` and `cd` some useful commands for navigating our file-system are

- `cp src dest` — this command will make a copy of the file located `src` at the location of `dest`. Remember, a file's full path is really its name, so `src` and `src` both represent paths even if they are just names (then they are relative paths!).
CAREFUL: if there already exists a file at `dest` then it will be *overwritten* meaning it is deleted and gone forever!

More shell commands for navigating the file system

In addition to `pwd` and `cd` some useful commands for navigating our file-system are

- `mv src dest` — this command will move the file located at `src` to the location of `dst`. **CAREFUL:** Just as with `cp`, if a file already exists at `dest` then it will be overwritten and destroyed forever!

More shell commands for navigating the file system

In addition to `pwd` and `cd` some useful commands for navigating our file-system are

- `rm file` — this command will delete a file forever! Unlike what you may be familiar with on Mac and Windows there will not be a trash or recycling can to potentially recover the file from. Please be careful!

Table of Contents

What are programs?

Operating Systems

Using The Shell and File Systems

Installing Python

Windows users that are using WSL Ubuntu should execute the following commands in order to install the Python interpreter they need.

```
sudo apt update  
sudo apt upgrade  
sudo apt-get install python3
```

Mac users should install Python by downloading the installer at <https://www.python.org/downloads/>.

Once the installer is downloaded you should execute the installer and follow the instructions it provides.

Running the Python interpreter

You should now be able to invoke the Python interpreter by running the command `python3` in your shell (regardless of Mac or Windows).

We'll learn more about using the Python interpreter shortly. If you want you can play around with it for now by entering arithmetic expressions using numbers, and the symbols `+`, `-`, `*`, and `/`.

We're going to put Python away for a moment however and discuss instead a concept from mathematics that will help us with the approach we're going to take to learning programming.