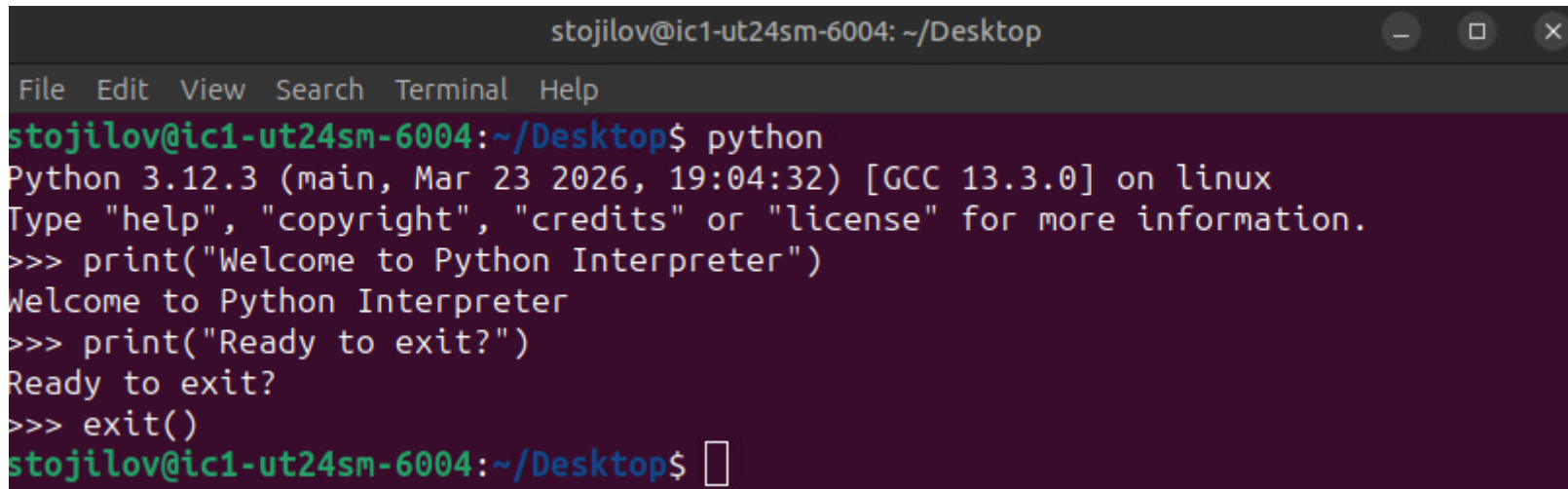


Information, Computation, Communication

Learning Python

Interactive Interpreter

A screenshot of a terminal window with a dark background. The window title is 'stojilov@ic1-ut24sm-6004: ~/Desktop'. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the execution of the 'python' command, which starts the Python 3.12.3 interpreter. The user enters 'print("Welcome to Python Interpreter")' and 'print("Ready to exit?")', which are printed to the screen. Finally, the user enters 'exit()', and the terminal returns to the shell prompt 'stojilov@ic1-ut24sm-6004:~/Desktop\$'.

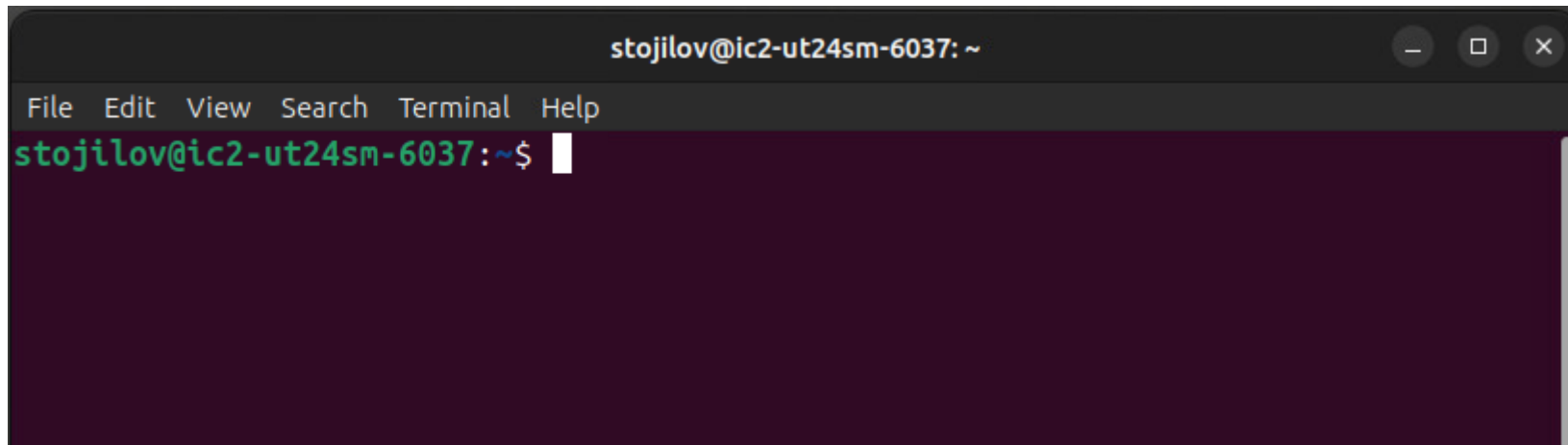
```
stojilov@ic1-ut24sm-6004: ~/Desktop
File Edit View Search Terminal Help
stojilov@ic1-ut24sm-6004:~/Desktop$ python
Python 3.12.3 (main, Mar 23 2026, 19:04:32) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Welcome to Python Interpreter")
Welcome to Python Interpreter
>>> print("Ready to exit?")
Ready to exit?
>>> exit()
stojilov@ic1-ut24sm-6004:~/Desktop$
```

Agenda

- [Launching Python interpreter](#)
- [Printing messages inside terminal \(console\)](#)
- [Computing](#)
- [Arithmetic and text](#)
- [Asking for input](#)
- [Exiting](#)

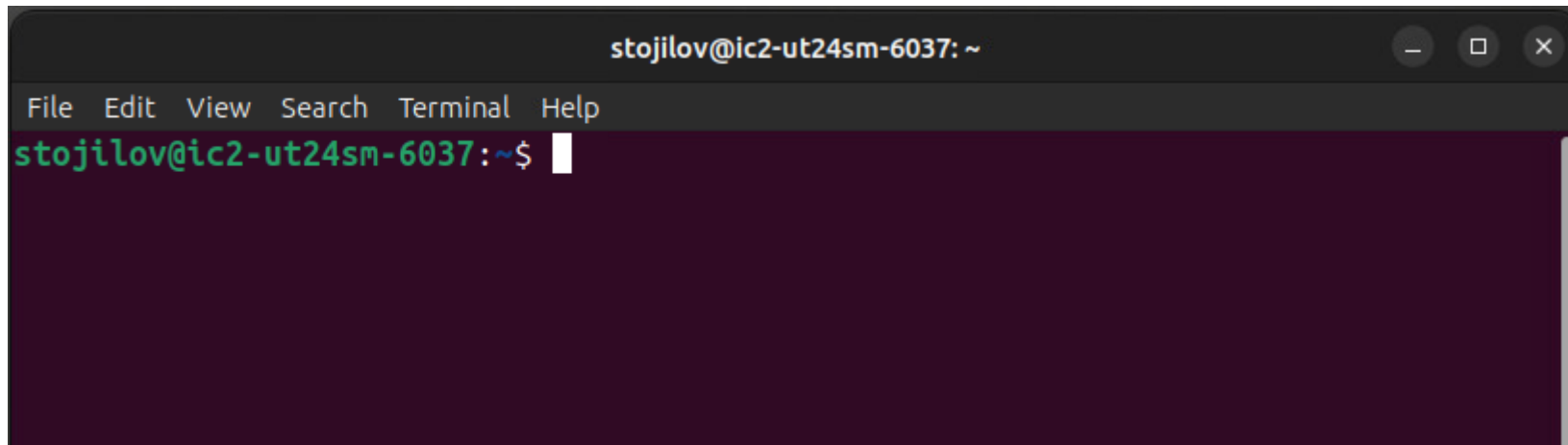
Before Python: What is a Terminal?

- A terminal is a text window for “talking” to the computer
- You type commands, press **ENTER**, and read the answer
- Terminal can start many programs; today we use it to start Python

A screenshot of a terminal window. The title bar at the top reads "stojilov@ic2-ut24sm-6037: ~" and includes standard window control buttons (minimize, maximize, close). Below the title bar is a menu bar with the options "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal is dark purple and contains a green prompt "stojilov@ic2-ut24sm-6037:~\$" followed by a white cursor. The terminal window has a vertical scrollbar on the right side.

Opening a Terminal

- **Windows:** open Command Prompt, PowerShell, or Windows Terminal
- **macOS:** open Terminal using Spotlight
- **Linux:** open your system's Terminal application;
on many distributions, **Ctrl + Alt + T** is a shortcut

A screenshot of a Linux terminal window. The title bar at the top reads "stojilov@ic2-ut24sm-6037: ~" and includes standard window control buttons (minimize, maximize, close). Below the title bar is a menu bar with the options "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal has a dark purple background. The prompt "stojilov@ic2-ut24sm-6037:~\$" is displayed in green text, followed by a white cursor. The rest of the terminal area is empty.



Python Interpreter: Launching from Terminal

To start interactive interpreter, type **python** in a terminal:

```
C:\> python
```

C:\> means the terminal is waiting

Interactive Interpreter: Launching

To start the interactive interpreter, type **python** in a terminal

```
C:\> python
```

```
Python 3.12.3 (main, Mar 23 2026, 19:04:32)
```

```
[GCC 13.3.0] on linux
```

```
Type "help", "copyright", "credits" or "license" for  
more information.
```

```
>>>
```

>>> means Python is waiting



Interactive Interpreter: Printing Messages (1)

Once the interpreter is active, try writing this

```
>>> print("How do you do?")
```

Interactive Interpreter: Printing Messages (2)

Once the interpreter is active, try writing this

```
>>> print("How do you do?")
```

How do you do?

This is what you'll see being printed
out in the terminal.

Strings (words inside quotes)
represent text which gets printed
in the terminal window



Interactive Interpreter: Computing (1)

Let's introduce variables (symbols) and do some computation

```
>>> x = 2
```

Creating a variable whose name is x and value is 2
This variable is an integer

Interactive Interpreter: Computing (2)

Let's introduce variables (symbols) and do some computation

```
>>> x = 2
```

Creating a variable whose name is x and value is 2
This variable is an *integer*

```
>>> x * 7
```

Let's multiply x by 7

Interactive Interpreter: Computing (3)

Let's introduce variables (symbols) and do some computation

```
>>> x = 2
```

Creating a variable whose name is x and value is 2
This variable is an *integer*

```
>>> x * 7
```

Let's multiply x by 7

```
14
```

It works!



Interactive Interpreter: Arithmetic and text? (1)

Try replacing numbers with strings

```
>>> x = "Are you ready "
```

Besides numbers, variables
can be strings (i.e., text)

Interactive Interpreter: Arithmetic and text? (2)

Try replacing numbers with strings

```
>>> x = "Are you ready "
```

Besides numbers, variables
can be strings (i.e., text)

```
>>> x + "to learn Python?"
```

What will happen if we use
+ (addition) with strings?

Interactive Interpreter: Arithmetic and text? (3)

Try replacing numbers with strings

```
>>> x = "Are you ready "
```

Besides numbers, variables
can be strings (i.e., text)

```
>>> x + "to learn Python?"
```

What will happen if we use
+ (addition) with strings?

```
'Are you ready to learn Python?'
```

Interactive Interpreter: Arithmetic and text? (4)

Try replacing numbers with strings

```
>>> x = "Are you ready "
```

Besides numbers, variables can be strings (i.e., text)

```
>>> x + "to learn Python?"
```

What will happen if we use + (addition) with strings?

```
'Are you ready to learn Python?'
```

+ (addition) will combine (*concatenate*) two strings into one! The whole expression will be evaluated, and the result will be printed.

Interactive Interpreter: Arithmetic and text? (5)

Try replacing numbers with strings

```
>>> x = "Are you ready "
```

Besides numbers, variables can be strings (i.e., text)

```
>>> x + "to learn Python?"
```

What will happen if we use + (addition) with strings?

```
'Are you ready to learn Python?'
```

In Python, strings can be enclosed between *apostrophes* or *double quotes*



Interactive Interpreter: Asking for Input (1)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

Interactive Interpreter: Asking for Input (2)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

What value do you want x to have?



Your message to the user
appears in the terminal
and the program is waiting
...zzzzzzzzzz.... on you to
answer!

Interactive Interpreter: Asking for Input (3)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

What value do you want x to have?

So let's answer.
For example...

77.045

Interactive Interpreter: Asking for Input (4)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

What value do you want x to have? 77.045

```
>>> print(x)
```

Did it work?
Let's print x to check

Interactive Interpreter: Asking for Input (5)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

What value do you want x to have? 77.045

```
>>> print(x)
```

It worked!

77.045

Interactive Interpreter: Asking for Input (6)

Ask the user (well, the user is **YOU** now) to provide
input to your program

```
>>> x = input("What value do you want x to have?")
```

What value do you want x to have? 77.045

```
>>> print(x)
```

77.045

It worked!

Printing can be even simpler. Try: **x**
Result: **'77.045'**



Interactive Interpreter: Asking for Input (7)

You can take text as input as well

```
>>> x = input("Say something...")
```

Say something...

Interactive Interpreter: Asking for Input (8)

You can take text as input as well

```
>>> x = input("Say something...")
```

Say something...

Let's answer.
For example...



Interactive Interpreter: Asking for Input (9)

You can take text as input as well

```
>>> x = input("Say something...")
```

Say something...Happy new semester!

What happened to x?
Let's print it to check



Interactive Interpreter: Asking for Input (10)

You can take text as input as well

```
>>> x = input("Say something...")
```

```
Say something...Happy new semester!
```

```
>>> print(x)
```

```
Happy new semester!
```

Interactive Interpreter: Asking for Input (11)

You can take text as input as well

```
>>> x = input("Say something...")
```

Say something...Happy new semester!

```
>>> print("You told me:", x)
```

You told me: Happy new semester!

Print can be used in many different ways!



Interactive Interpreter: Exiting

To close interactive interpreter, type `exit()` in the terminal
or press `ctrl` together with `z`

```
>>> exit()
```

- Interactive interpreter is great, but...All we type is **temporary** and **lost** once the interpreter is closed
- **Better alternative:** write code in **a file** (also known as **scripting**), which can be saved and reused



Next topic:

Numbers, Operators, Booleans