# CSCI 141

Command Line Arguments

## Goals

- Know how to pass user input to a program via command line arguments.
- Know how to access those arguments inside a program.

# User Input

command line arguments

(sometimes also known as)

program arguments

## User Input

- Getting input from the user is often useful.

command line arguments

(sometimes also known as)

program arguments

## User Input

- Getting input from the user is often useful.
- So far, we've seen:

  `input()` or `input(prompt)`

command line arguments

(sometimes also known as)

program arguments

# User Input

- Getting input from the user is often useful.
- So far, we've seen:

recall that `prompt` is an *argument* to the function

`input()` or `input(prompt)`

command line arguments

(sometimes also known as)

program arguments

# User Input

- Getting input from the user is often useful.
- So far, we've seen:

recall that `prompt` is an *argument* to the function

`input()` or `input(prompt)`

- Another approach is called

command line arguments

(sometimes also known as)

program arguments

## User Input:
## Command Line Arguments

## User Input: Command Line Arguments

To use command line arguments, we need to do two things:

## User Input:
## Command Line Arguments

To use command line arguments, we need to do two things:

1. **Pass** arguments to the program when running it.

## User Input: Command Line Arguments

To use command line arguments, we need to do two things:

1. **Pass** arguments to the program when running it.

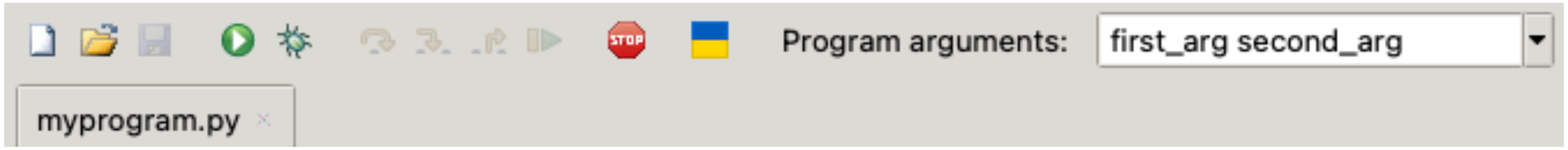
2. **Access** the arguments in the program's code.

# 1. Passing Command Line Arguments

This depends on how you run your program. For Thonny:

1. Enable the Program arguments box by checking **Program arguments** in the **View** menu. You should only need to do this once per Thonny installation.

2. Type the arguments in the Program arguments box, then click the Friendly Green Run Button



On a command line:

Add the arguments to your command, separated by spaces:

```
python3 myprogram.py first_arg second_arg
```

## 2. **Accessing** Command Line Arguments

This uses syntax that we won't see for a while; for now, here's the incantation:

**Important**: command line arguments always have type `str`!

## 2. **Accessing** Command Line Arguments

This uses syntax that we won't see for a while; for now, here's the incantation:

```
import sys # at the top of your program
```

**Important**: command line arguments always have type `str`!

## 2. Accessing Command Line Arguments

This uses syntax that we won't see for a while; for now, here's the incantation:

```
import sys # at the top of your program

first_argument = sys.argv[1]
```

**Important**: command line arguments always have type `str`!

## 2. Accessing Command Line Arguments

This uses syntax that we won't see for a while; for now, here's the incantation:

```
import sys # at the top of your program

first_argument = sys.argv[1]
second_argument = sys.argv[2]
```

**Important**: command line arguments always have type `str`!

## 2. Accessing Command Line Arguments

This uses syntax that we won't see for a while; for now, here's the incantation:

```
import sys # at the top of your program

first_argument = sys.argv[1]
second_argument = sys.argv[2]
# and so on...
```

**Important**: command line arguments always have type `str`!

# Demo

- Run a program that takes and prints two command line arguments in Thonny.
- Run the same program from the command line.
- What happens if you try to read an argument that the user didn't supply?
- What happens if you never use an argument that *was* supplied to the program?

# `input` vs command line args

Why use one over the other?

# Demo

- add2.py: write a program that takes two integers as command line arguments and prints their sum
    - Notice that we need to convert them to integers!
    - What happens if the provided arguments do not look like integers?