Managing Large Code Projects

Learning Goals

- Use the input command and try/except structures to handle direct user input in code

- Implement and use helper functions in code to break up large problems into solvable subtasks

- Install external modules with the pip command

- Read documentation to learn how to use a new module

CS as a Tool

Our next section focuses on how computer science can be used to benefit other domains.

We'll investigate three different applications of computer science: data analysis, simulation, and machine learning.

These three applications share a core idea in common: all three organize data to help people answer questions.

Before we get into these topics, we need to learn a bit about how to approach coding large projects instead of individual functions.

User Input

Input-Output Cycle

When you use a program (like your internet browser, or MS word), you're communicating with the computer. You give the computer input on what you want to do, and it produces output based on your requests.

When we write programs, we can capture input directly from the user. When combined with output (printed strings), we can make programs that a interact with a user!

input

output
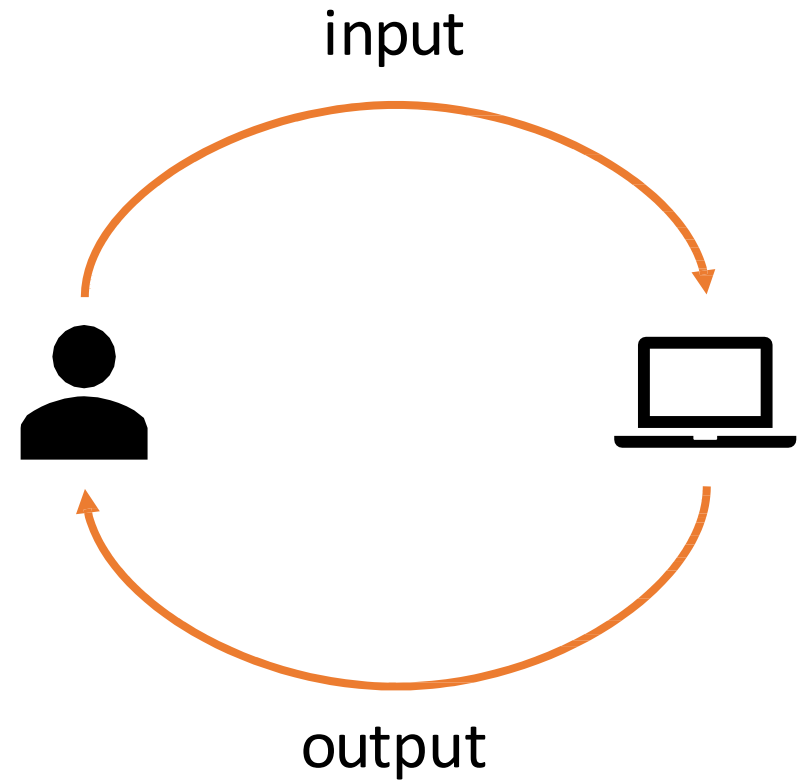
Getting Input from the User

Up until now, we've written programs that get their input solely from the function arguments we provide.   Alternatively, we can write programs that ask the user to enter information while the program is running.

The built-in function input(msg)displays a message in the interpreter, lets the user type a response in the interpreter, then returns the response as a string when the user presses enter.

```
name = input("Enter your name: ")
print("Hello, " + name + "!")
```

input() Returns a String

input() will always return a string. If we want to use a user's response as a number, we need to use type-casting to change it.

age = int(input("Enter your age: "))

print("You'll be", age + 1, "next year")

Note: users sometimes enter unexpected whitespace at the beginning or end of a response. The built-in function s.strip() may prove useful! It removes extra whitespace from the beginning/end of a string.

Handle User Errors with Try-Except Statements

What happens if we ask the user to enter a number and try to convert their text to a number, but they enter a non-number instead?

The code will throw a ValueError when it tries to convert the text to an int. This is not great, because users get frustrated if the program crashes each time they make a mistake.

In order to make a program robust against human errors, we can use a try-except control structure to recover from such errors.

Try-Except Statements

A Try-Except statement looks like this:

try:

      <try-block>

except:

      <what to do if the try code throws an error>

This works a bit like an if-else statement. Go to the try block first. If the code in the try block runs correctly, the except block is skipped. Alternatively, if Python encounters a runtime error in the try block, it immediately exits that block and jumps to the beginning of the except block.

Example: Inputting a Number

Let's try our age-entering program again, this time with error handling.

```
try:
        age = int(input("Enter your age:"))
        print("You'll be", age + 1, "next year")
except:
        print("That's not a real age!")
```

Note that the first print statement does not run if the user enters a non-number into the input.

Activity: Write error-catching code

You do: write a short snippet of code that asks the user to enter two numbers (with two separate input() calls), then prints the result of multiplying those two numbers. If at least one of the inputs isn't a number, print an error message using an except block.

Test your code by trying good inputs (two inputs) and bad inputs of different kinds.

User Input Loops Ensure Correct Inputs

Sometimes you might need to try several times to get the user to input a valid option into the program.

When you need to get a real input from a user, use a loop to continue asking them for input until they get it right.

What's the loop control variable? You could use the variable you're setting based on the user's entry. You could also use while True, then break when you get the right input.

Example: Entering y/n

For example, let's write a simple program that requires the user to respond with either y (yes) or n (no).

```
answer = ""
while True:
        answer = input("Do you like ice cream? [y/n]:")
        if answer == "y" or answer == "n":
                break
        else:
                print("Seriously, answer the question.")
if answer == "y":
        print("Me too!")
else:
        print("Lactose intolerance sucks :(")
```

Helper Functions

Helper Functions

In your assignments, the code you write will be bigger than a single function. You'll often need to write many functions that work together to solve a larger problem.

We call a function that solves part of a larger problem this way a helper function. By breaking up a large problem into multiple smaller problems and solving those problems with helper functions, we can make complicated tasks more approachable.

We've used helper functions in class before- to calculate how much to pay for a meal (helper function calculated tip), in selection sort (to swap two items in a list), and in merge sort (to merge two lists).

Designing Helper Functions

Try to identify subtasks that are repeated or are separate from the main goal; break down the problem into smaller parts. Have one subtask per function to keep things simple.

Example: Connected4

Consider the game connected4 in the assignment. It seems simple, but it involves multiple parts to play through a whole game.

Python Modules

Python Modules

The Python programming language comes with a large set of built-in functions that cover a range of different purposes. However, it would take too long to load all these functions every time we want to run a program.

Python organizes its different functions into modules. When you run Python, it loads only a small set of functions from the built-in module. To use any other functions, you must import them.

Built-in Modules

We've already used a few core modules for homework assignments - mainly
math and multiprocessing.

For a full list of python libraries, look here:
https://docs.python.org/3/library/index.html

External Modules

There are many other libraries that have been built by developers outside of the core Python team to add additional functionality to the language. These modules don't come as part of the Python language, but can be added in. We call these external modules.

In order to use an external module, you must first install it on your machine. To install, you'll need to download the files from the internet to your computer, then integrate them with the main Python library so that the language knows where the module is located.

Finding Useful Modules

One of the main strengths of Python as a language is that there are thousands of external modules available, which means that you can start many projects based on work others have done instead of starting from scratch.

You can find a list of popular modules here: wiki.python.org/moin/UsefulModules And a more complete list of pip-installable modules here: pypi.org

There are bonus slides on the course website that introduce Python modules that are popular among CMU students

pip

It is usually possible to install modules manually, but this process can be a major pain. Luckily, Python also gives us a streamlined approach for installing modules – the pip module! This feature can locate modules that are indexed in the Python Package Index (a list of commonly-used modules), download them, and attempt to install them.

Traditionally, programmers run pip from the terminal. This is a command interface that lets you make changes directly to your computer. But in this class, we'll just run pip in Colab.

Running pip

To run pip in Colab, use this command in the interpreter

pip install module-name

This will identify the module and your version of Python and start the download and installation process. It may run into a dependency error if the module needs a second module to already be installed – in general, installing that module and then running pip again will fix the problem.

Using an Installed Module

Once you've successfully installed a module, you should be able to put

import module-name

at the top of a Python file, and it will load the module the same way it would load a built-in library.

Note: this may fail if you have multiple versions of Python installed on your machine and you install in the terminal. Make sure to use the pip associated with the version of Python you're using in your editor. You can check your editor's version in PyCharm with Shell > Edit Shell Configurations (check the value in exe), then call pip using

pythonversion-number   -m pip install module-name

Documentation

Learning how a Module Works

Once a new module is installed, you're still left with a major question: how do you use it?

This varies by module, but the best answer is to read the documentation. Most external modules have official documentation or APIs that describe which functions exist and how to use the module.

For example: how can we run a T-test on two datasets using scipy?

T-test Documentation

Searching "python scipy t-test" gets us this link from scipy's official documentation.

The page describes what the function does. It also outlines the required arguments and what it returns.

It's common for functions to also have optional keyword parameters, like tkinter's fill parameter; we can choose to include these or leave them to the default value.

scipy.stats.ttest_ind

scipy.stats.**ttest_ind**(*a*, *b*, *axis=0*, *equal_var=True*, *nan_policy='propagate'*, *alternative='two-sided'*)   [source]

Calculate the T-test for the means of *two independent* samples of scores.

This is a two-sided test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

Parameters: **a, b** : *array_like*

The arrays must have the same shape, except in the dimension corresponding to *axis* (the first, by default).

axis : *int or None, optional*

Axis along which to compute test. If None, compute over the whole arrays, *a*, and *b*.

equal_var : *bool, optional*

If True (default), perform a standard independent 2 sample test that assumes equal population variances [1]. If False, perform Welch's t-test, which does not assume equal population variance [2].
New in version 0.11.0.

Returns: **statistic** : *float or array*

The calculated t-statistic.

pvalue : *float or array*

The two-tailed p-value.

Documentation with Examples

Really good documentation may also contain examples showing how to use the function!

If you want to learn how to use a function, try copying the example code into your editor and running it. Then try changing some things to see how the results are affected.

If you find that you're completely lost, it's likely that the module organizes things differently from what we've learned in class. Search the documentation site for a Getting Started
/ Introduction page to learn the basics.

Examples

```
>>> from scipy import stats
>>> np.random.seed(12345678)
```

Test with sample with identical means:

```
>>> rvs1 = stats.norm.rvs(loc=5,scale=10,size=500)
>>> rvs2 = stats.norm.rvs(loc=5,scale=10,size=500)
>>> stats.ttest_ind(rvs1,rvs2)
(0.26833823296239279, 0.78849443369564776)
>>> stats.ttest_ind(rvs1,rvs2, equal_var = False)
(0.26833823296239279, 0.78849452749500748)
```

ttest_ind underestimates p for unequal variances:

```
>>> rvs3 = stats.norm.rvs(loc=5, scale=20, size=500)
>>> stats.ttest_ind(rvs1, rvs3)
(-0.46580283298287162, 0.64145827413436174)
>>> stats.ttest_ind(rvs1, rvs3, equal_var = False)
(-0.46580283298287162, 0.64149646246569292)
```

When n1 != n2, the equal variance t-statistic is no longer equal to the unequal variance t-statistic:

```
>>> rvs4 = stats.norm.rvs(loc=5, scale=20, size=100)
>>> stats.ttest_ind(rvs1, rvs4)
(-0.99882539442782481, 0.3182832709103896)
```

Other Resources

It can also be helpful to search online for other projects that have used the same module, to find examples of how to set it up. Many people have written helpful tutorials for this exact purpose.

Two standard resources for finding help are StackOverflow, a site where people can ask questions about code and get answers from other developers, and GitHub, a site where people post open-source projects for others to use and contribute to.

IMPORTANT: whenever you copy code from online, make sure to cite it the same way you would cite a paragraph of text in an essay. You can do this by putting a comment above the copied code that includes a link to the URL you got the code from.