T01: Choose Your Own Adventure

- Assignment T01 should be done in pairs and in-class.
- You are not expected to work on teamworks outside of class.
- Much of T01 will require software to be installed; refer back to HW00 if you are missing something!

Learning Objectives:

- Explore your PyCharm IDE environment.
- Understand Python files.
- Take a first dive into Python code.
- Use some Python commands and understand what they do.
- Work with conditionals to make decisions in Python.
- Practice some creative writing and storytelling.
- Understand some different data types and how to use them to make decisions.

How to Start

- To begin, make a copy of this document by going to File >> Make a Copy...
- Share the copied document with all members of your team. You can share this document by hitting the blue button in the top right of the document, then entering the email addresses of all members in the bottom input field.
- Change the file name of this document to **username1, username2 - T01: Choose Your Own Adventure** (for example, **heggens, wilborned - T01: Choose Your Own Adventure**). To do this, click the label in the top left corner of your browser.
- Next, go to the **Github classroom for T01**.
- You will be asked to create a team:
  - One of you will create the team.
  - Then, the other will join the team created by the first partner.
- Paste the link to your team's Github repo here:

Github Repo Link:	https://github.com/Berea-College-CSC-226/t01-choose-your-own-adventure-beststudents

Working with a Partner

In each teamwork, you will work with at least one partner, sometimes two. At the beginning of each day in class, enter your name in the following table.

Team assignment T01 should be explored in class with a partner. In the future, we will work as "pair programmers", where two programmers work together simultaneously solving the same problem **on a single computer**. Today, we will be doing a different practice called "partner programming", where you work on the same problem in parallel **on your own computers** because we want everyone to get their laptops set up with all the tools we will be using in this course. Work in parallel with your partner to answer questions in this document, but be sure to follow all steps on your own computer so you're ready for future classes as well.

There will be a lot of steps in this teamwork assignment which may assume some understanding of your Windows operating system. If you find yourself confused by an instruction, first ask your partner for help. If you are both unable to determine what it means, you should feel encouraged to raise your hand. One of the instructors or a TA will help as soon as we're able.

Both partners should be on the same step throughout this assignment. Do not race ahead, or let yourself fall behind. Communication is key! I should hear a lot of talking amongst partners throughout the class period! I should not see you on the internet browsing, and I DEFINITELY should not see your phone!

Step 1: The PyCharm IDE

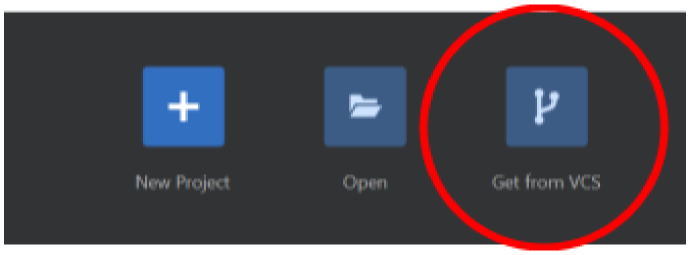
According to Wikipedia[1], "An integrated development environment (IDE) is a software application that helps edit and debug code." In other words, it is a program used to create other programs. Our IDE is PyCharm. Open up PyCharm and we'll start exploring. Some important notes:

- HW00: Task 5 asks you to download and install PyCharm. If you haven't done that yet, now is a good time to switch over to that document, and complete the installation process.
- You'll also need your Github account and Git installed on your computer. If you haven't completed HW00: Task 6 and Task 7, do those now.
- PyCharm is a large IDE, and often takes a long time to load. When you come to class in the future, start PyCharm right away so you are not wasting time waiting on it.

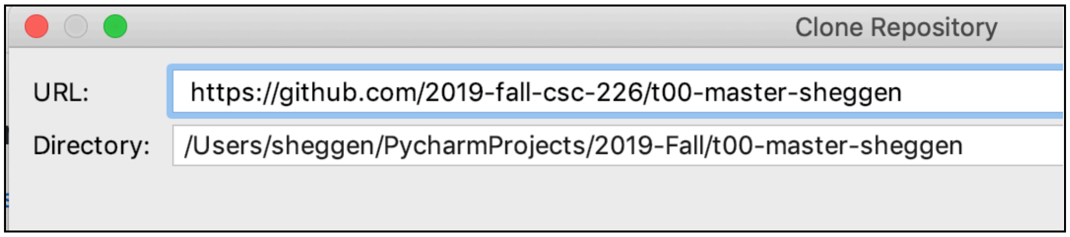
Help your partner get Step 1 complete. Once you are both done, you can move on to Step 2. Do not run ahead of your partner, that is bad teamwork! Do ask for help if you both get stuck.

Step 2: The Projects Page

As PyCharm opens, you get a Welcome screen. As time goes on, this screen will fill up with all the work you are doing. Right now, it's probably pretty empty.



1. To begin, click either the **Get from VCS** or **Clone Repository** button**.**
2. Put the following information in the two boxes that appear, in order:
    - Git Repository URL:
        - Copy the link you pasted above and paste it here.
    - Directory:
        - Leave this to the default location, which is likely your PycharmProjects folder:



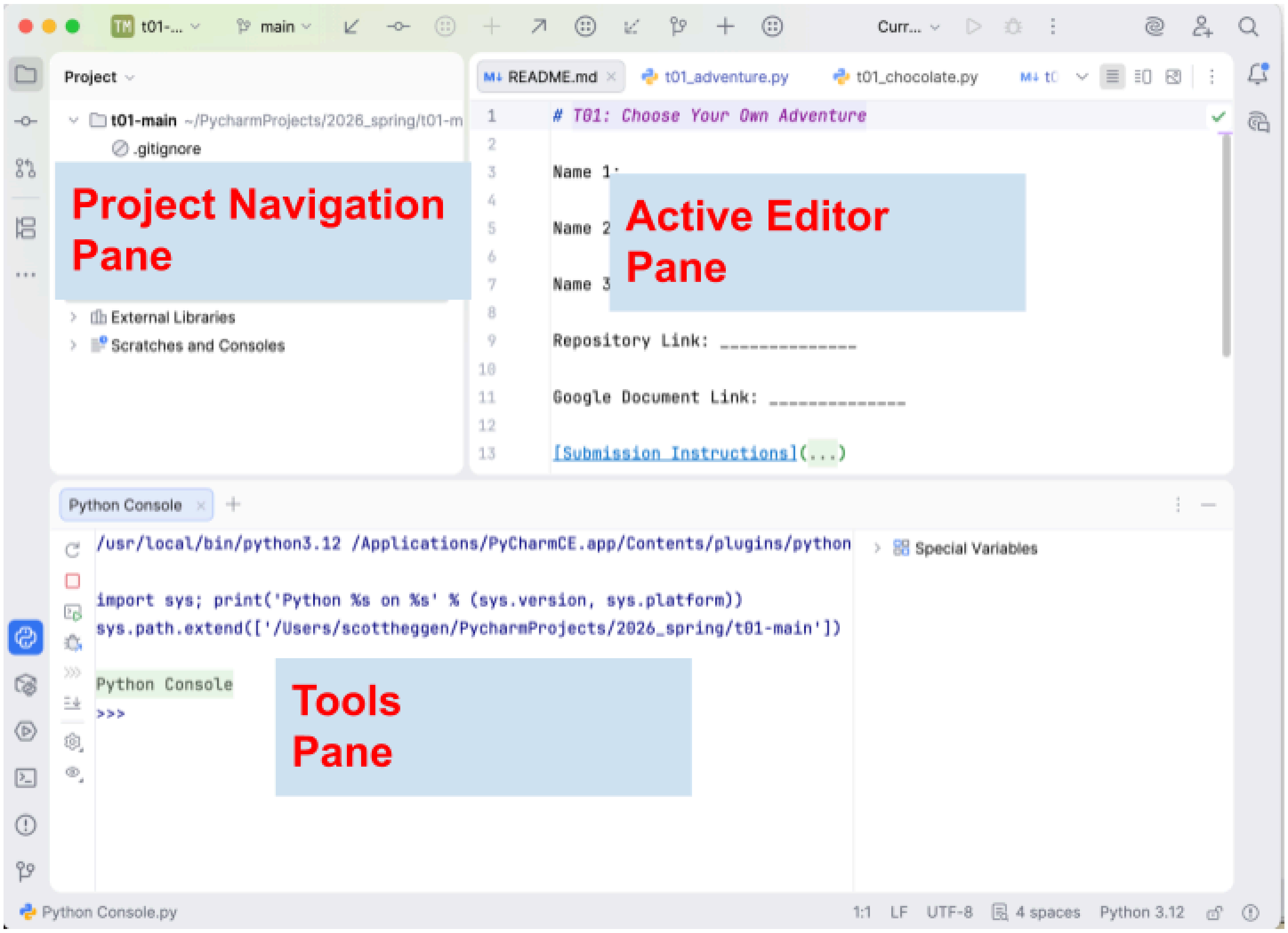
URL: https://github.com/2019-fall-csc-226/t00-master-sheggen
Directory: /Users/sheggen/PycharmProjects/2019-Fall/t00-master-sheggen

[1] https://en.wikipedia.org/wiki/Integrated_development_environment

3. Click **Clone**. Click **Yes** if it asks you to open the directory. This will take you into the main PyCharm programming environment. You may also need to put in your Github username and password.
4. In the PyCharm menu bar, go to **Git >> New Branch...**
5. Name the branch your usernames (e.g., **heggens_wilborned**). Click **Create**.

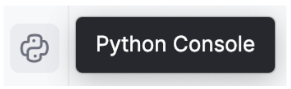
Step 3: The Panes of PyCharm

PyCharm has three main areas you'll be working in: the Project Navigation pane, the Active Editor pane, and the Tools pane:



```
# T01: Choose Your Own Adventure

Name 1:

Name 2

Name 3

Repository Link: ______________

Google Document Link: ______________

[Submission Instructions](...)
```

```
/usr/local/bin/python3.12 /Applications/PyCharmCE.app/Contents/plugins/python

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/scottheggen/PycharmProjects/2026_spring/t01-main'])

Python Console
>>>
```

First, click the 

button in the Tools Pane. It is an interpreter window which you can think of as a scratch pad. Hence, you can type commands directly into it, and they'll execute.

The Active Editor pane is where you will be doing the majority of your work. This is where you'll edit a file. Think of it like the area where you edit a Word document; you'll edit code in the Active Editor pane.

The Project Navigation pane is where you'll find all of your files related to a project. You'll see a folder with files in there. The folder is named the same as your directory name from the step above. Inside the folder are multiple files, downloaded from the Github repository when you started.

In the PyCharm Project Navigation Pane, find the **t01_questions.md** file and open it. Answer the question in **SECTION 1**, then come back to this document to continue the assignment.

Step 4: Play With Some Code

At this point, it may make sense for one partner to keep the **t01_questions.md** file open, and the other partner to enter commands into PyCharm. Though, you should both take turns entering commands (i.e., switch roles, and switch often) so you both become comfortable using the IDE.

Follow the instructions below, and record what is happening. For the first few questions, you can use the Python Console, as you did before.

Go to **t01_questions.md** and answer **SECTION 2**, then return to this document to continue the assignment

Step 5: Choose your Own Adventure

The Cave of Time, by Edward Packard (1979), is one of the first and best known "Choose-your-own-adventure" books. A choose-your-own-adventure book asks the reader to make decisions at key points in the book. For example, the reader may have to make a choice between following the strange mystic man or waiting at a bar for his friend. Depending on his choice, the story could end, or the adventure could continue. Typically, the story has multiple, very different endings. In fact, The Cave of Time book has 36 unique endings, depending on your choices. You can see the book analyzed here.

Your Task

Your team will be writing a chapter of our own Choose Your Own Adventure tale using Python!

Open the starter code **t01_adventure.py**. Most of the instructions are in the file, but you should read below to make sure you don't miss anything important.

Code Structure

Note that the code is broken up into sections:
Lines 1 - 8: Header block, which contains basic information
Lines 9 - 14: Acknowledgement block, which contains space for citing sources used to solve this assignment
Lines 15 - 24: Beginning of the code, which has some required imports and variables that will be used later.
Lines 25 - 38: More code, which is the start of the Choose your own adventure story
Lines 39 - 67: An example story, which your code will mimic
Lines 68 - 77: Space for you to write your code
Lines 78 - 81: The ending of the story

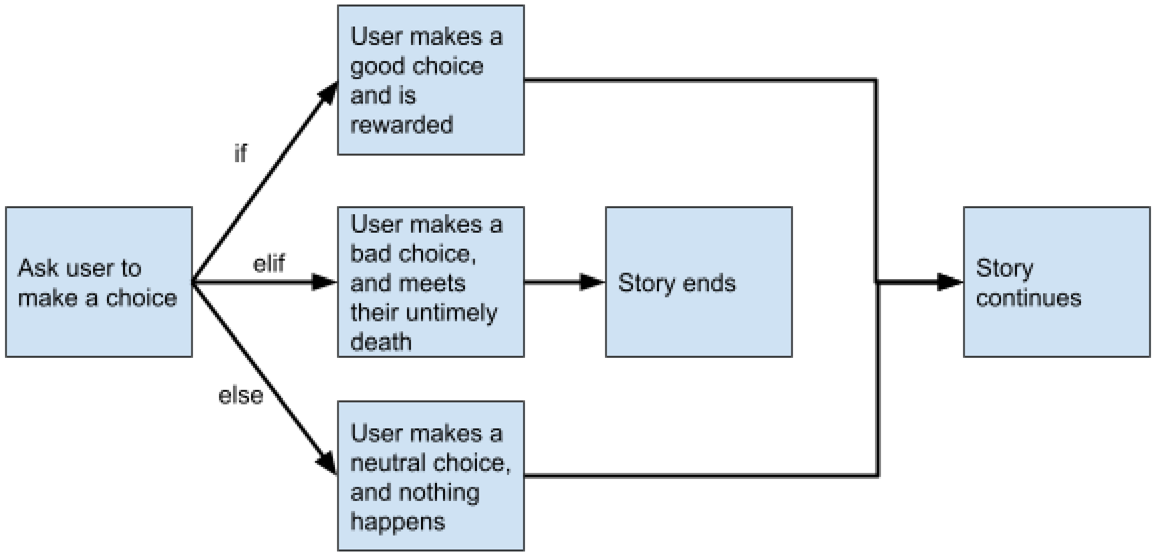
The only section you need to edit is between lines 68 - 77!

Creating the Branching Structure

Use the following rules and hints to create your chapter of the story:

- Your chapter should ask the user a question, and the user will input a response to that question.
- You will need to decide which potential user input constitutes a “good” choice and a “bad” choice.
- Your chapter will then make one of three decisions based on the user’s choice:
    a. The user made the right choice, is rewarded with an encouraging message (e.g., *You found a golden egg!*), and the story will advance.
    b. The user made the wrong choice, and the story ends with an untimely death (e.g., *You found a rotten egg, and ate it. Oops, you're dead.*).
    c. The user’s choice was not “death worthy”. The story moves on, but the user gets no reward (e.g., *You enter a new room. There's lots of doors...*). Use the `else` statement to catch these cases.

The following flowchart represents the requirements listed above.


User makes a good choice and is rewarded
if
Ask user to make a choice
elif
User makes a bad choice, and meets their untimely death
Story ends
Story continues
else
User makes a neutral choice, and nothing happens

- Try not to copy and paste the starter code. You will learn more and better by writing it from scratch. However, the starter code is there to help you when you get stuck.
- Be creative in your writing. Have fun with it!

After you have completed the code to solve the problem above, go to **t01_questions.md** and answer **SECTION 3**, then return to this document to continue the assignment

Step 6: Enhancing the Branching Structure [OPTIONAL if time allows]

- Modify your code so that if the user picked the wrong choice (step b. above), give them one more opportunity to *not* die. Ask the user to make another decision. This time, make sure the user is inputting an integer (e.g., *The Eggmeister asks you for a number between 1 and 20...*):
    a. Create another conditional statement which evaluates using the less than, less than or equal to, greater than, or greater than or equal to operators (`<`, `<=`, `>`, or `>=`).

i. If they choose wisely, let the story progress.
ii. If they chose poorly, give them a murder most foul.
iii. You and your partner decide what “wisely” and “poorly” mean. Remember, you are the authors of this adventure book.

Go to **t01_questions.md** and answer **SECTION 4**, then return to this document to continue the assignment

Submission Instructions

At the end of every assignment, I will include these instructions. They do change on occasion, so be sure you check them each assignment to ensure no special instructions were added.

Follow the submission instructions by next Friday at 11:55PM.

Backup code:

```
# T01: Choose Your Own Adventure

## Instructions

To begin, click the button at the top of PyCharm which says `main`. Click `Create a new branch`. Name your new branch
`t01_username1_username2`, replacing the two usernames with you and your partner's usernames.

**Replace each instance of `**Replace This With Your Answer**` with your answer**

___

## SECTION 1

1.a. In the top right corner of PyCharm are three buttons. Click the first button which looks like three horizontal
    lines (it may already be selected).

    Hold your mouse over the button. What markdown mode are you in?

```
   Editor
```

1.b. Now select the third button, which looks like a picture of a mountain and the sun.

    Hold your mouse over the button. What markdown mode are you in?
    Describe how the interface looks different than the last mode.

```
  Preview; the last mode allowed us to edit (thus the interface held colored text that looked unanimous) while this new mode, the preview, allows us to simply view the end markdown created from the editor (thus looks fancier)
```

1.c. Now select the middle button.

    Hold your mouse over the button. What markdown mode are you in?
    Describe how the interface looks different than the last mode.

```
  Editor and Preview; the last mode held no opportunity to edit, only allowing for one to view the resulting markdown of the editor while this one, with both Editor and Preview, allows for us to write within the editor and immediately view the markdown we created.
```

1.d. Now look in the bottom left corner of Pycharm. Click the top button, which opens the Python Console.

    In the Python console, try typing some arithmetic statements like:
```

~~~
       20 + 23

   Describe the result:

```
  The result of "20+23" being typed into the Python Console was the integer form of 43,
having completed the addition of our desired numbers
```

_Return to the Google Doc to continue this assignment._

___

## SECTION 2

2.a. Use the Python Console for arithmetic. In the space to the right, report what happens
(generally) for each operator.
    Is it what you expect, in all cases? What cases did something unexpected?

   At a minimum, try each of the following operators: +, -, *, and / using positive and
negative integers (e.g., -5, 13)
   as well as positive and negative non-integer real numbers (e.g., -2.35).

```
2+5 = 7
-7-9 = -16
-9.67* 66.2 = -640.154
2.98/ -2 = -1.49
```

2.b. Use the Python Console to determine what // (two backslashes) does when used as an
operator with integers.
    Try enough examples to determine precisely what it does. Explain.

```
66//3 = 22
66/3 = 22.0

5//4 = 1
5/4 = 1.25

Based on this, the // operator does integer division that ignores the decimal point and
returns an integer.
```

2.c. Use the Python console to determine what ** does when used with integers. Try enough
examples to determine
    precisely what it does. Explain.

```
~~~

~~~
2 ** 4 = 16
3**12 = 531441
2**4 = 16

It raises the initial integer to the power of the second integer
```


___

2.d. Use the Python console with ^ and integers to prove that it does not do the same thing as **. Report what you tried
    and what you got. Try the same numbers with ** and report what you got. Do not worry about trying to figure out
    what ^ does quite yet -- we give you some additional tools to figure that out with the next bullet.
    Just report on what you got and whether or not they are the same.

```
2^4 = 6
2**4 = 16

They are not the same.
```

Type `help('^')` in PyCharm's Python console and look through to see what it says about the ^ operator. Feel free
to spend some time discussing this and Googling if you are interested in what this is. We mostly wanted you to know
about the `help` feature.

2.e. Do not Google this one at first. Instead, just guess. Use the Python console to see if you can figure out how to
    convert a non-integer (like 3.4 or 6.78) to an integer number (like 3 or 6) in Python. Report how you do it and
    also how many guesses it took until you figured it out. Note that there are different ways. You only need to find
    one of them, but please try enough examples so you can describe precisely what it does.

```
x_float = 3.45
x_int = int(x_float)

it took us one try as we both have previous experience.
```

2.f. Type `x = 5` in the PyCharm's Python console. Then type `print(x)`. Report what happens.

```
  It prints the value stored in x.
```
~~~

~~~
2.g. Do these steps in order:

   i. Type x == 10. Note the number of equal signs. Report what happens.

```
  It prints False.
```

   ii. Next type x == 5. Report what happens.

```
  It prints True.
```

   iii. Next type x = 10. Report what happens.

```
  Nothing printed to the console, but the value on x in memory got updated to 10.
```

   iv. Next type x == 10 again. Report what happens.

```
  The console returns True.
```

2.h. Explore the use of "=" and "==" until you understand how they differ. Then precisely
describe their differences.

```
  **=** :  Assigns a value (it can be of any datatype) to a variable so that it can store it
in memory.
  ****==**** :  It compared the left side to the right side and prints TRUE when they are
both the same and prints FALSE when they are not equal.
```

_Return to the Google Doc to continue this assignment._

___

## SECTION 3

3.a. Based on your code, why did we use else in step c? In other words, what happens if the
user puts in an option you
    didn’t expect (like “down” in my example code)?

```
  **Replace This With Your Answer**
```
~~~

~~~
3.b. When you get the above code working, test it multiple times to see if it makes any
mistakes. Note any bugs you
    created that you cannot figure out how to fix:

```
  **Replace This With Your Answer**
```

_Return to the Google Doc to continue this assignment._

___

## SECTION 4

4.a. Does your conditional handle all possible inputs from the user, so long as the input is
a number? What happens if
    the user puts in a float (e.g., 3.14)?

```
  **Replace This With Your Answer**
```

4.b. Was the logic for this step more, less or no more difficult to write than the
conditional statements you wrote to
    satisfy the flowchart above? Explain.

```
  **Replace This With Your Answer**
```

4.c. What happens if the user inputs a string (e.g., "cat") in the previous step?

```
  **Replace This With Your Answer**
```

Finally, head to our [#csc226 Slack channel](https://bereacs.slack.com/archives/C3QACGH8R).
Either a) leave a question about something you found confusing in T01, or
b) summarize the most challenging part of the assignment in a paragraph or so.


_Return to the Google Doc to continue this assignment._

___
~~~