TECHNOPython

Teacher Guide

Lessons for Middle and High School Students

Technology Project using

Python

Become a programmer.

In this project, students are introduced to Python, a text-based programming language. They complete coding missions to develop the characteristics most valued in a programmer. To start, they ignite their curiosity by exploring scripts to discover how they are put together. Next, they create a series of games including Pet Monster Rescue, Guess It, and Adventure Quest. These foster logical thinking, persistence, and creativity, and are ideal for beginners. Upon completion, students share their favorite Python program in a coding presentation to build strong communication skills. Fun activities highlight the importance of being a team player. Do more than copy lines of code. Have your students develop original scripts using variables, loops, functions, and conditionals.

TECHNOKids®

Contents

TechnoPython Overview

Introduction to TechnoPython

In this project, students are introduced to Python, a text-based programming language. They complete coding missions to develop the characteristics most valued in a programmer. To start, they ignite their curiosity by exploring scripts to discover how they are put together. Next, they create a series of games including Pet Monster Rescue, Guess It, and Adventure Quest. These foster logical thinking, persistence, and creativity, and are ideal for beginners. Upon completion, students share their favorite Python program in a coding presentation to build strong communication skills. Fun activities highlight the importance of being a team player. Do more than copy lines of code. Have your students develop original scripts using variables, loops, functions, and conditionals.

Students complete the following tasks:

- In session 1, students explore the Coding Jungle. The goal of this mission is to learn about Python, which is a text-based programming language. To start, the explorers are introduced to terminology by experimenting with code. Once familiar with the role of a programmer, they play a Python Hunt game and then edit the program to discover how it works. Afterwards, they break code in the Catch the Bugs game to develop essential debugging skills. Successful completion of the four-part mission requires curiosity, which is a highly valued trait in a programmer.
- In session 2, students create a program for the Pet Monster Rescue, which is a group that finds loving homes for monsters. To prepare for the programming mission, students learn about strings, integers, and variables. They apply this knowledge to personalize the adoption process. To pair a pet owner to their monster, the programmers write code that ask questions. The answers are used to match people to their ideal pet. This is done by combining logical operators, if and else statements, and a variable that changes from True to False. Throughout the four-part mission, an emphasis is placed upon thinking logically, which is an ability every successful programmer needs.
- In session 3, students design a guessing game in which the player must correctly pick a number before they run out of chances. Clues tell them if their answer is too high or low. This programming mission has six parts. To prepare, students first explore how to code while and for loops. Once familiar with how to repeat a set of instructions, they start to build Guess It. To guide development, the Python programmers sequence steps into algorithms. These flowcharts provide a framework for constructing each part of the program. Fun challenges encourage students to build a unique game. Interwoven throughout all tasks is a focus upon being methodical. This skill helps programmers test different cases to solve problems within the code.
- In session 4, students develop a text-based adventure game. It is a quest that has players overcome challenges to earn rewards. To prepare for this programming mission, students learn techniques to standardize data entry. Next, they apply these skills to build the first part of their game. It will allow players to pick a direction to explore. It will also include a challenge whereby the player can win coins when they travel North. To complete the task, students must be persistent. What will happen in this strange land?

- In session 5, students complete their text-based adventure game. They develop a treasure hunt that has players travel East to collect objects. They must avoid danger, or risk losing it all! To prepare for this part of the programming mission, students learn about lists. They add, remove, sort, and count items. Once this skill is mastered, they apply it to their quest. Throughout the activities, an emphasis is placed upon creativity. This trait is essential as it allows programmers to design original programs.
- In session 6, students share their favorite Python program in a coding presentation. They demonstrate how the game works and explain the code. This provides an opportunity to develop strong communication skills, which help programmers do their job.

Implementation and Technology Integration Ideas

TechnoPython introduces the text-based Python programming language to middle and high school students. They develop programs including the Pet Monster Rescue, Guess It, and Adventure Quest games. It is an ideal project for Grades 6 and up.

Ideas for Implementation

Have your students become programmers using the fun activities in the TechnoPython project. Easy to follow instructions gradually introduce computer science concepts. The activities are suitable for any teaching situation. Select the option that works best for you and your students:

- *STEM Project and a Computer Science Class*: This option is best suited to teachers that have a chunk of instructional time allocated to STEM. TechnoPython has 24 assignments that systematically develop programming skills. The young programmers will learn how to divide a task into steps using an algorithm, follow a plan to sequence instructions, and debug code. In addition, they become aware of how to create variables, manipulate data, format output, import libraries, loop instructions, apply conditional logic, manage lists, and build functions. TechnoPython provides a solid foundation for mastering text-based programming languages in the future.
- *Coding Unit for Beginners*: Assignments in Sessions 1-3 are ideal for students new to text-based programming. The step-by-step instructions explain how to build simple games. Experimentation is encouraged to help young programmers understand Python. As well, there are templates to jump start learning. Session 6 includes a coding presentation activity. It can be completed whether students have built one program or many.
- *Coding Unit for Advanced Beginners:* Once students understand the basics of programming, they extend their learning in Sessions 4-5. The programming task of creating an Adventure Quest is more complicated. The instructions require students to customize the code to suit their storyline. Moreover, many of the scripts include fill-in-the-blanks and do not tell exactly where to place the code. This is done to encourage logical thinking. These assignments are ideal for students that understand the fundamentals and are ready for a challenge. Session 6 includes a coding presentation activity. It can be completed whether students have built one program or many.
- *Hour of Code*: If you only have one class to teach the Python programming language there are many assignments in TechnoPython that can be used for this purpose. For example, if your students are beginners, they can experiment with editing scripts in Session 1 to develop a basic understanding. Or another option is to complete the exploratory assignments such as Assignment 5, 9, 10, 15, and 20.
- *Makerspace Workshop Series*: If you are running a workshop series as part of an after-school program or community event, then you will need to select assignments that fit the number of classes offered. When selecting a mission consider instructional time. For example, Pet Monster Rescue and Guess It can be completed in fewer classes than Adventure Quest. In addition, think about the age range, reading level, keyboarding skills, and programming abilities of students. Activities gain in complexity.

Technology Integration Suggestions

The TechnoPython project is primarily a STEM project that teaches programming skills. However, the activities also integrate into other areas of curriculum including mathematics, language arts, or social studies.

- *Mathematics Problem Solving Unit*
  Connect mathematics to programming. TechnoPython has many activities that combine computational thinking with mathematical reasoning. For example, students apply logical thinking to organize their ideas into a flowchart, that outlines the steps in a program. As well, many of the programs require conditional logic to trigger actions.
- *Language Arts*: Include Python as part of a unique language arts unit. The Adventure Quest has a storyline that uses first person narrative to describe events. The programmer tells the player what they see, how they feel, and what they do as they explore a strange land.
- *Social Studies*: Transfer map making skills to a new task. The Session 5 Extension Activity has students draw a map of a strange land using PowerPoint or Google Slides.
- *Computer Science*: TechnoPython is an introduction to text-based Python programming. The project includes a Summary of Skills that details computer science learning outcomes. Students learn how to manipulate variables, calculate values, trigger actions using logic, format output, define functions, manage lists, and more!

Preparing to Teach TechnoPython

Complete the following:

Step 1: Read the "Get Started in 5 Easy Steps" Instructions
Step 2: Download Python
Step 3: Install PDF App
Step 4: Share the Python folder with Students
Step 5: View Sample Programs (Optional)
Step 6: Select Assessment Tools (Optional)
Step 7: Monitor Progress (Optional)
Step 8: Print Handouts (Optional)

Step 1: Read the "Get Started in 5 Easy Steps" Instructions

Visit the *Getting Started* page to jump start your use of TechnoKids projects. It has everything you need to know about accessing files, installing PDF viewers, viewing a project folder, and sharing resources with students.

https://www.technokids.com/support/getting-started.aspx

Step 2: Download Python

You must have the latest version of Python on your device. The instructions are written for 3.8.5. If you have an earlier version, you must update it.

1. Visit https://www.python.org/downloads/
2. Before you click the *Download* button you may want to click on the link below that lists your operating system. This is advisable if you are using a 64-bit version of Windows.

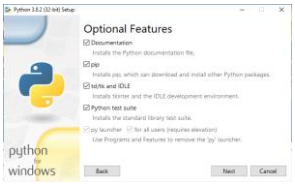
How to Check Windows Version:
1. Click *Start*.
2. Type About your PC.
3. Select the *System Settings* option.
4. Refer to device specifications.

If you selected your operating system instead of the *Download* button, find your version from the list. Select the *executable installer* link. Save the file.

3. Install Python. Your window will look something like this:

Make sure that the download includes IDLE and pip.

If you pick *Customize installation* select ALL the options.



Step 3: Install PDF App

The teacher guide and student workbook are in PDF format. The files have security settings and are best viewed using Adobe Reader or a recommended PDF viewer.

Each assignment in the workbook is available as an individual worksheet. Students can read the instructions and type answers into the worksheets. Adobe Acrobat Reader is recommended for desktop devices. Or there are several free PDF reader options such as Kami that can be used with Google Drive.

Follow Step 2 on the *Getting Started* page.

https://www.technokids.com/support/getting-started.aspx#step2

Step 4: Share the Python folder with Students

The *Python* folder contains templates of Python programs and resource materials. This folder can be placed on a local computer, memory stick, school server, or private web-based folder (not available to the public).

1. Open the *TechnoPython Resources* folder.
2. Copy the *Python folder*.
3. Paste it in a desired share location or upload the folder contents to a private web-based location.

Step 5: View Sample Programs (Optional)

Example scripts are available. Use them as a source of inspiration.

1. Open the *TechnoPython Resources* folder.
2. Open the *Samples* folder. They are organized by session.
3. Right click on a file and choose *Edit with IDLE > Edit with IDLE…*
4. When the program opens, from the Run menu choose *Run Module,* or press F5.

Step 6: Select Assessment Tools (Optional)

TechnoPython has several tools for assessing and evaluating student work. Review the contents of the *TechnoPython Resources \ Assessment* folder. It includes marking sheets, peer reviews, reviews, and skill reviews. Files are customizable.

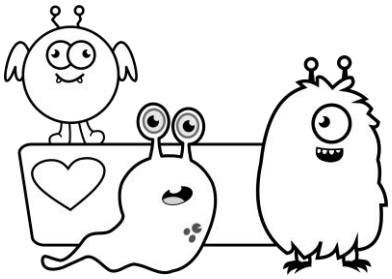
Step 7: Monitor Progress (Optional)

TechnoPython has several tools for tracking progress. Each mission has a map and task list. Students can use them to self-monitor and recognize accomplishments. These files are in the *TechnoPython Resources \ Missions* folder.

Step 8: Print Handouts (Optional)

At the end of the project you can send home a parent letter or certificate to celebrate learning. These files are in the *TechnoPython Resources \ Handouts* folder.

Assignment 5 Prepare for the Pet Monster Rescue Mission



It is time for a new programming mission. You are going to create a program for the Pet Monster Rescue. Pet Monster Rescue is a group that finds loving homes for monsters.

The program will match people to their ideal pet. It will ask questions, store answers, and use logic to determine if there is a monster that meets the person's needs.

To successfully complete the mission, you must think logically. This is a valued trait in a programmer. People who are logical:

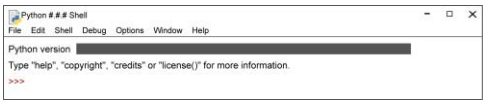
- carefully watch what is happening
- pay attention to details
- outline ideas clearly by breaking them down into parts
- study facts to determine if a statement is True or False

To complete the task, aside from being logical, you also need to know more about Python. Follow the instructions to write code to learn about strings, integers, and variables.

Open the Python Shell

1. ▷ Open IDLE (Python).

   ▷ View the Python Shell:



What is a String?

A string is text. It can be a word, phrase, or sentence. In Python, str is short for string.
To learn about strings, type each line of code and then press ENTER to study the output.

2. ▷ A string must have brackets and quotes around it:

```
>>> print('a string is text')
```

▷ A string can have single or double quotes around it:

```
>>> print("it must have quotes")
```

▷ Use double quotes if you want to use an apostrophe in the string:

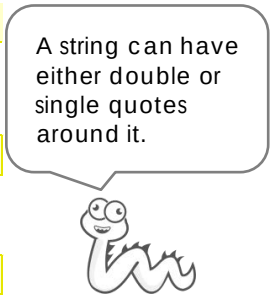
```
>>> print("let's code")
```

▷ Use single quotes if you want to show someone talking in the string:

```
>>> print('he said "okay"')
```

▷ If you want both, you need to put a slash before the apostrophe:

```
>>> print('she said "let\'s start now"')
```



▷ Be logical. Study the above code to determine how to complete each task:

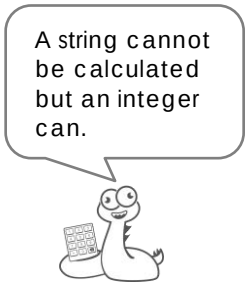
☐ it's fun ☐ I said "wow" ☐ I said "wow, it's fun"

What is an Integer?

An integer is a whole number such as 10 or 42. In Python, int is short for integer.
To learn about integers, type each line of code and then press ENTER to study the output.

3. ▷ An integer must have brackets around it:

```
>>> print(5)
5
```



▷ An integer can be used to calculate values:

```
>>> print(6+4)
10
```

▷ If you put quotes around a number, it becomes a string:

```
>>> print('6+4')
6+4
```

▷ Think logically! Study the above code to determine how to complete each task:

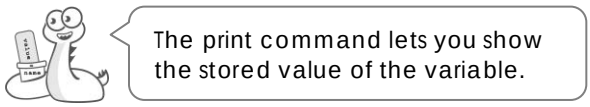
☐ show `10` ☐ calculate `3+2` ☐ show the equation `3+2=`

What is a Variable?

A variable stores a value that can change. It can be text, a number, or a list of items.
To learn about variables, type each line of code and then press ENTER to study the output.

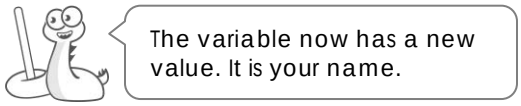
4. ▷ A variable has two parts – name and value:

```
>>> name=('value')
>>> print(name)
value
```



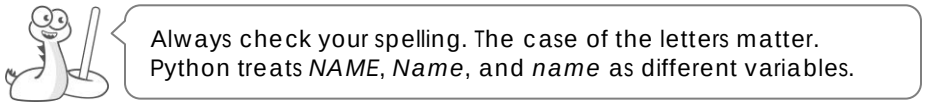
▷ The variable value can change:

```
>>> name=('Student Name')
>>> print(name)
Student Name
```



▷ The variable is case specific. Type a capital N in the variable name:

```
>>> print(Name)
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print(Name)
NameError: name 'Name' is not defined
```



▷ A variable can store a string or integer. Think logically! Show the value using `print`:

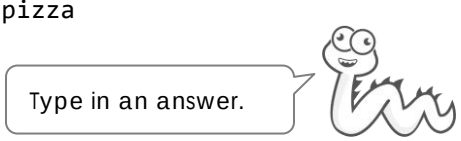
☐ `weather=('sunny')` ☐ `mood=('happy')` ☐ `count=(5)`

Input a Variable Value

Sometimes the programmer will input a value for a variable. However, other times the user can input a value. This is done using the `input` command.

5. ▷ Ask the user a question:

```
>>>food=input('What food do you like?')
What food do you like?pizza
```



▷ The input is stored by the computer:

```
>>> print(food)
pizza
```

▷ The input can be used to have the program talk to the user:

```
>>> print('I also like', food)
I also like pizza
```

▷ Think logically! Study the above code to determine how to ask each question:

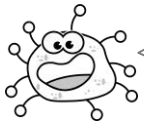
☐ What music do you like? ☐ What is your age? ☐ Do you like sports?

Name a Variable

A variable name must be meaningful. It should describe its purpose. There are rules you must follow when assigning a variable name. To learn them, type each line of code and then press ENTER. Does the variable name follow the rules, or do you get an error?

6. ▷ A variable name must start with a letter or underscore, but not a symbol or number.
Put a checkmark ✓ beside the variable names that work:

☐ `color=('red')` ☐ `2color=('red')` ☐ `_color=('red')` ☐ `!color=('red')`

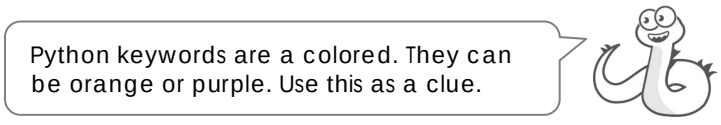


▷ A variable name must be one word with no spaces.
Put a checkmark ✓ beside the variable names that work:

☐ `game_score=(0)` ☐ `game score=(0)` ☐ `gamescore=0`

▷ A variable name cannot be a Python keyword. Keywords are color-coded.
Put a checkmark ✓ beside the variable name that works:

☐ `answer=('yes')` ☐ `True=('yes')` ☐ `import=('yes')`



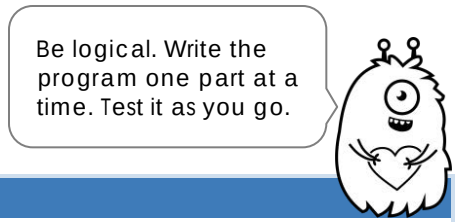
Close the Python Shell

Assignment 6 About the Pet Monster Rescue Mission

In this programming mission, you design a program for the Pet Monster Rescue. It is a group that helps people adopt pet monsters in need of loving homes.

To start, you will apply what you know about strings and variables to inform others about the Pet Monster Rescue. To personalize the adoption process, you will ask questions.

To interact with the user, you will use the following code:



CODE	PURPOSE
`print('Type text here.')`	show text
`print("Let's use an apostrophe.")`	use double quotes around text to show an apostrophe
name`=('value')`	create a variable and set the value to text
name`=input('What is the question?')`	create a variable that has the user input the value
`print('Type text',` *name*`)`	make a sentence that includes the variable value
`print('Type text',` *name*`+'.')`	remove the space between the variable value and text

---

Open IDLE and Create a New File

1. ▷ Open IDLE (Python).

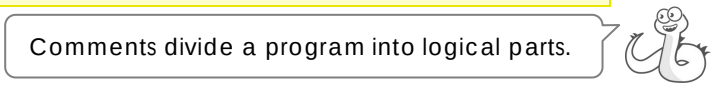
   ▷ From the File menu, select *New File*.

   ▷ From the File menu, select *Save*.
   Type monster as the file name. Click *Save*.

Inform Others About the Pet Monster Rescue

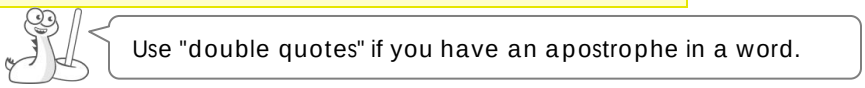
2. ▷ A comment is a note to the programmer. Organize your code:

```
#about
```



   ▷ Tell others about the Pet Monster Rescue program:

```
#about
print('Answer questions to find a pet.')
print("Let's get started.")
```



3. ▷ From the File menu, select *Save* or press CTRL + S.

   ▷ From the Run menu, select *Run Module*.

   ▷ Read the text:

```
Answer questions to find a pet.
Let's get started.
>>>
```

   ▷ Close the Python Shell.

Create a Variable to Store the Value 'Pet Monster Rescue'

4. ▷ At the top of the program add a section for variables:

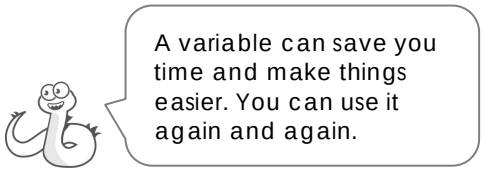
```
#variables

#about
print('Answer questions to find a pet.')
print("Let's get started.")
```

▷ Create a variable to store the name of the place:

```
#variables
place=('Pet Monster Rescue')
```

▷ Use the variable to show a title in the *about* section:

```
#variables
place=('Pet Monster Rescue')

#about
print(place)
print('Answer questions to find a pet.')
print("Let's get started.")
```



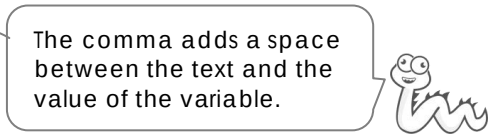
▷ Save the changes. From the Run menu, select *Run Module*.

▷ When done, close the Python Shell.

Use the Variable Value in Sentences

5. ▷ You can include the value of a variable in a sentence:

```
#about
print(place)
print('Welcome to', place)
print('Answer questions to find a pet.')
print("Let's get started.")
```

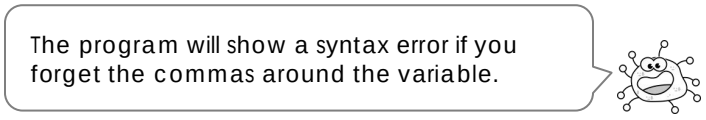


▷ Apply your skills to view the changes.

```
Pet Monster Rescue
Welcome to Pet Monster Rescue
Answer questions to find a pet.
Let's get started.
>>>
```

6. ▷ You can include the value of a variable in the middle of a sentence:

```
print("Let's get started.")
print('Thanks for visiting', place, 'where pets are family.')
```



▷ Apply your skills to view the changes.

Ask a Question to Get to Know the Pet Owner

7. ▷ You can have the user input the value of a variable. Ask the person's name using `input`:

```
#variables
place=('Pet Monster Rescue')

#about
print(place)
print('Welcome to', place)
print('Answer questions to find a pet.')
print("Let's get started.")
print('Thanks for visiting', place, 'where pets are family.')

#pet owner
name=input('What is your name?')
```

▷ The answer can be used to have the program talk to the user:

```
#pet owner
name=input('What is your name?')
print('Hello', name)
```

▷ Apply your skills to view the changes. Type the answer to the question:

```
Pet Monster Rescue
Welcome to Pet Monster Rescue
Answer questions to find a pet.
Let's get started.
Thanks for visiting Pet Monster Rescue where pets are family.
What is your name?Alex
Hello Alex
>>>
```

To make it easier to read, you need a space between the question and answer. You should also add a period after the name in the greeting.

Make the Text Easier to Read

8. ▷ Add a space between the question and the answer:

```
#pet owner
name=input('What is your name? ')
print('Hello', name)
```

add a space before the quote

▷ To add a period to a sentence you must use a plus sign:

```
#pet owner
name=input('What is your name? ')
print('Hello', name+'.')
```

add a plus sign + to remove space between a variable and text

A comma before or after a variable creates a space.
A plus sign before or after a variable removes any space.

▷ Apply your skills to view the changes.

Ask Another Question

9. ▷ Ask the person to describe their home:

```
#pet owner
name=input('What is your name? ')
print('Hello', name+'.')
home=input('Is your home calm or busy? ')
print('Okay. Your home is a', home,'place.')
```

▷ Apply your skills to view the changes. Type the answers to each question:

```
Pet Monster Rescue
Welcome to Pet Monster Rescue
Answer questions to find a pet.
Let's get started.
Thanks for visiting Pet Monster Rescue where pets are family.
What is your name? Alex
Hello Alex.
Is your home calm or busy? busy
Okay. Your home is a busy place.
>>>
```

Take the Coding Challenge

10. You know lots about how to show text on the screen, store variables, ask questions, and use the answers in a sentence. Pick a challenge to improve the program.

☐ Add a decorative line around the title in the about section. For example:

```
-*-*-*-*-*-*-*-*-*-
Pet Monster Rescue
-*-*-*-*-*-*-*-*-*-
```

HINT: You need to add two `print` lines. One above and one below the title.

☐ Add a sentence after the welcome greeting:

```
You're going to love your new pet.
```

HINT: It has an apostrophe. You need double quotes around the string.

☐ Add a comma after the variable value *Pet Monster Rescue*:

```
Thanks for visiting Pet Monster Rescue, where pets are family.
```

HINT: You need to use a plus sign + after the variable.

☐ Ask the question *How many pet monsters do you own?*
Reply *So you have _ .*

```
How many pet monsters do you own? 2
So, you have 2.
```

HINT: You need to make a variable. Name it and use the command `input` to ask the question.

When testing a program, you can kill or stop running it. To do this, click *Close* X. Click *OK.*

Close Python

Assignment 7 Be Logical – Organize Your Ideas

Your Pet Monster Rescue program is looking great!

Next, you need to write the part of the code that will help people find their ideal pet. You will do that by asking questions. If the answer is a match to a monster in the shelter, the person can take their new pet home. If there is no match, the program will continue to ask questions.

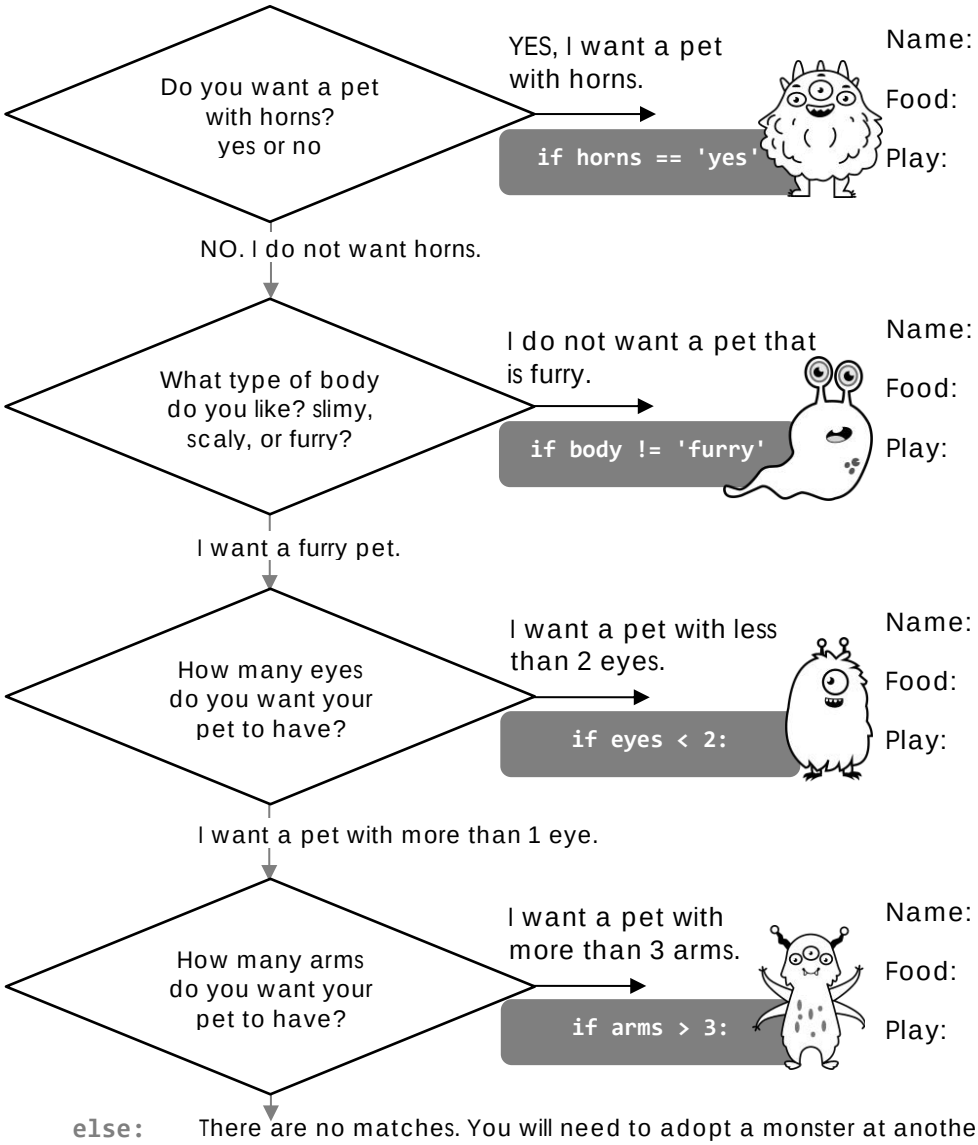
This part of the program will use logic to make decisions:

==	equal
!=	different
>	greater than
<	less than

if	A keyword used to trigger an action when a statement is True.
else	A keyword used to trigger an action only when all other statements above are False.

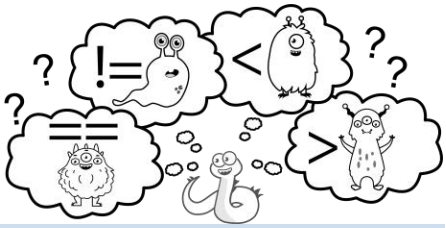
Before you start to write the code, you must form a plan. Programmers are logical. They often use a flowchart to make an algorithm. An algorithm is a description of what the program will do.

Complete the algorithm to describe each monster. What is their name? What food do they eat? What do they like to play?



Assignment 8 Build Decision Making into the Code

In this assignment, you complete the Pet Monster Rescue program using your plan from Assignment 7. You will write code that asks questions. The answers will be used to match a person to their ideal pet.



The program will make decisions using the following code:

CODE	PURPOSE
`find=True` `find=False`	A variable that starts and stops the search for a pet. While `find` is set to True the person is asked a list of questions. When `find` changes to False the search ends.
`while find:`	A loop that contains a list of questions. The person will be asked each question in order. If no match is found to a pet, the search ends.
`if horns=='yes':`	A statement that finds a pet with horns if the person's answer is 'yes'.
`if body!='furry':`	A statement that finds a pet if the person's answer is not 'furry'.
`if num_eyes<2:`	A statement that finds a pet with one eye if the number is less than 2.
`if num_arms>3:`	A statement that finds a pet with 4 or more arms if the number is greater than 3.
`break`	A command that stops the loop. This happens when a person finds a pet and no longer needs to answer questions.
`else`	An action that happens if there are no matches after answering all the questions. `find` changes to False causing the search to end.

Open the Saved Monster File in IDLE

1. ▷ Open IDLE (Python).

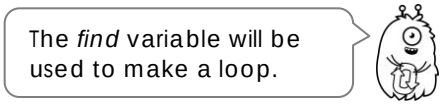
   ▷ From the File menu, select *Open*.

   ▷ Go to the place where you saved the monster file. Select it. Click *Open*.

Control the Search

2. ▷ A *find* variable will be used to start and stop the search. Add it to the variable section:

```
#variables
place=('Pet Monster Rescue')
find=True
```



   ▷ Create a loop that controls a list of questions:

```
#pet owner
name=input('What is your name? ')
print('Hello', name+'.')
home=input('Is your home calm or busy? ')
print('Okay. Your home is a', home, 'place.')
own=input('How many pet monsters do you own? ')
print('So you have', own+'.')

#find match
while find:
```

press ENTER

Find a Match if the Person Wants a Pet with Horns

3. ▷ Ask a question and store the answer as the variable *horns*:

```
#find match
while find:
    horns=input('Do you want a pet with horns? yes or no ')
```

The answer will be used to find a pet with horns.

▷ From the File menu, select *Save* or press CTRL + S.

▷ From the Run menu, select *Run Module*.

▷ The question repeats. This is because there is nothing to stop the loop yet:

```
Do you want a pet with horns? yes or no yes
Do you want a pet with horns? yes or no no
Do you want a pet with horns? yes or no yes
Do you want a pet with horns? yes or no
```

▷ Close the Python Shell.

You need to add code that will match a person to a pet if they answer yes.

4. ▷ Add an *if* statement that matches a person to a pet with horns:

```
#find match
while find:
    horns=input('Do you want a pet with horns? yes or no ')
    if horns=='yes':
        print('We have a match.')
        print('Describe the pet. Refer to Assignment 7.')
```

What is the pet's name? What does it eat and play?

Test the Code

5. ▷ Apply your skills to run the program.

▷ When you see the question you just added, type yes.

▷ Read about the matching pet monster:

```
Do you want a pet with horns? yes or no yes
We have a match.
It is Furhop. He likes to eat pizza and play ring toss.
Do you want a pet with horns? yes or no
```

▷ When you see the question again, type no.

The question keeps repeating itself. You need to add code that stops the loop.

▷ Close the Python Shell.

Stop the Loop to End the Search

6. ▷ If there is no match, the search needs to end. Set the *find* variable to *False*:

```
if horns=='yes':
    print('We have a match.')
    print('It is Furhop. He likes to eat pizza and play ring toss.')
else:
    find=False
    print('We do not have a matching pet.')
```

press BACKSPACE

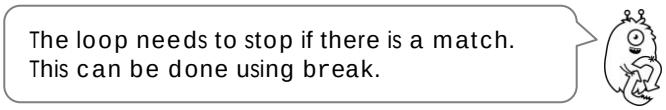
▷ Test the code. Run the program. When you see the question you just added, type no:

```
Do you want a pet with horns? yes or no no
We do not have a matching pet.
>>>
```

▷ Close the Python Shell.

7. ▷ Run the program again. When you see the question, type yes:

```
Do you want a pet with horns? yes or no yes
We have a match.
It is Furhop. He likes to eat pizza and play ring toss.
Do you want a pet with horns? yes or no
```



▷ Close the Python Shell.

▷ Add break to stop the loop:

```
if horns=='yes':
    print('We have a match.')
    print('It is Furhop. He likes to eat pizza and play ring toss.')
    break
else:
```

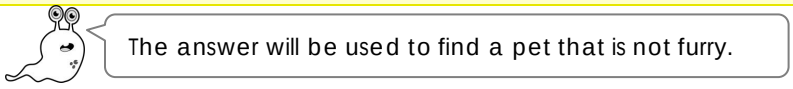
▷ Apply your skills to test that the loop ends when there is a match.

Make a Match if the Answer is Not Furry

8. ▷ Ask another question and store the answer as the variable *body*:

```
if horns=='yes':
    print('We have a match.')
    print('It is Furhop. He likes to eat pizza and play ring toss.')
    break
body=input('What type of body do you like? slimy, scaly, or furry? ')
else:
```

press ENTER



▷ Add an *if* statement that matches a person to a pet that is not furry:

```
body=input('What type of body do you like? slimy, scaly, or furry? ')
if body!='furry':
    print('You answered', body)
    print('We have a match. Describe the pet. Refer to Assignment 7.')
    break
```

Be Logical – Test One Value at a Time

9. ▷ Apply your skills to run the program.

   ▷ When you are asked if you want a pet with horns, type no.

```
Do you want a pet with horns? yes or no no
```

Since there is no match, the program will ask the next question.

   ▷ When you are asked about the body of the pet, type furry.

```
What type of body do you like? slimy, scaly, or furry? furry
We do not have a matching pet.
>>>
```

The program only finds a match if the answer is different from furry. Try it!

   ▷ Apply your skills to test the program again:

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? slimy
You answered slimy
We have a match. It is Slimo. He likes to eat bugs and swim in mud.
>>>
```

   ▷ Close the Python Shell.

Make a Match if the Person Wants Less than 2 Eyes

10. ▷ Ask another question and store the answer as the variable *eyes*. Add the logic:

```
    eyes=input('How many eyes do you want your pet to have? ')
    if eyes<2:
        print('You want a pet with', eyes, 'eye.')
        print('We have a match. Describe the pet. Refer to Assignment 7.')
        break
else:
```

   ▷ Apply your skills to test the program:

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? furry
How many eyes do you want your pet to have? 1
Traceback (most recent call last):
  File "C:\Users\Student\Documents\monster.py", line 36, in <module>
    if eyes<2:
TypeError: '<' not supported between instances of 'str' and 'int'
```

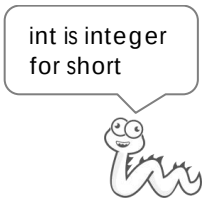
The answer to a question is stored as a string. Text is not a number value. You must change the variable *eyes* to an integer.

   ▷ Close the Python Shell.

Change the Variable Type so the Logic Can Work

11. ▷ Create a new variable to turn *eyes* into an integer. Edit the code:

```
eyes=input('How many eyes do you want your pet to have? ')
num_eyes=int(eyes)
if num_eyes<2:
    print('You want a pet with', num_eyes, 'eye.')
```



▷ Apply your skills to test the program again:

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? furry
How many eyes do you want your pet to have? 1
You want a pet with 1 eye.
We have a match. It is Cyclopee. He likes to eat worms and sleep.
```

▷ Apply your skills to test the program again. Pick a different value such as 2 or 3.

Make a Match if the Person Wants More than 3 Arms

12. ▷ Ask another question and store the answer as the variable *arms*. Add the logic:

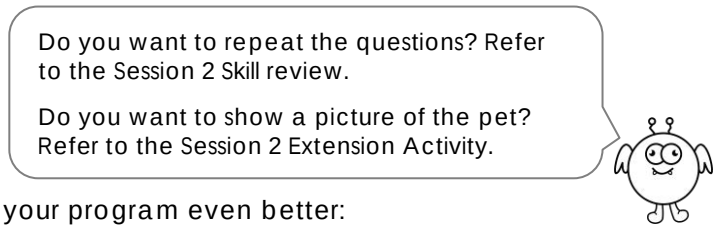
```
arms=input('How many arms do you want your pet to have? ')
num_arms=int(arms)
if num_arms>3:
    print('You want a pet with', num_arms, 'arms.')
    print('We have a match. Describe the pet. Refer to Assignment 7.')
    break
else:
```

▷ Apply your skills to test the program. Test it with 2 arms. Test it again with 4 arms.

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? furry
How many eyes do you want your pet to have? 3
How many arms do you want your pet to have? 2
We do not have a matching pet.
```

▷ Close the Python Shell.

Take the Coding Challenge



13. Pick from the options below to make your program even better:

- ☐ Add a period to complete the sentence about the body: *You answered slimy.*
- ☐ Add the text *Let's keep looking* before the body, eyes, and arms questions.
- ☐ Add the question, *Do you want a pet with wings?*
  If yes, the person should match to a pet.
- ☐ Add the question, *How many legs do you want your pet to have?*
  If the number is greater than 2, the person should match to a pet.

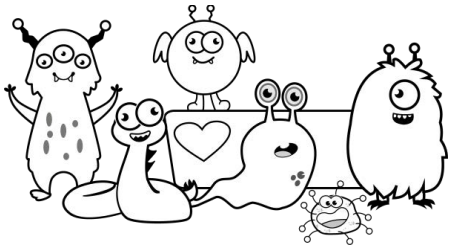
Close Python

Session 2 Peer Review: Rescue a Pet Monster

Invite a friend to adopt a pet monster.

1. Decide how to share the file. The Pet Rescue Mission program can be open on your device. You can also send the player the monster.py file by email or a file sharing service.

2. Get feedback! Ask the person to answer the questions below after they are done.



Pet Monster Rescue Feedback

Pet Owner:

Programmer Name:

1. Did you find a pet? ☐ yes ☐ no

2. If yes, describe your new pet.

   Name:

   Food:

   Play:

3. What ONE thing do you like about the program?

   ☐ It was easy to use.
   ☐ The questions made you feel as if the pet you wanted was important.
   ☐ The matching pet was one you would like to own.
   ☐ The pet was creative or funny.
   ☐ Other:

4. What ONE thing do you think the programmer should change to make it *even* better?

   ☐ Add a blank line between questions to make it easier to read.
   ☐ Add titles to divide the text such as ***Get to Know Pet Owner*** or ***Find a Pet***.
   ☐ Display a message that invites people to run the program again.
   ☐ Change a monster's name to _____

☐ Other:

Session 2 Review: About Strings, Integers, and Variables

1. Match the programming term to its meaning.

variable	integer	string	comment

a. string — Text such as a word, phrase, or sentence.

b. comment — Note to a programmer about the code.

c. integer — Whole number.

d. variable — A stored value that can change.

/4

---

2. Complete each string using the correct symbols.

"	\	'

a. print(' I can program ')

b. print(" let's code ")

c. print('I said, "let \'s code" ')

/3

Pick the correct code for the output.

3. **10**

a.
```
print('5+5')
```

b.
```
print(5+5)
```

c.
```
print("6+4")
```

4. **Hello Beth**

a.
```
name=('Beth')
print('Hello', name)
```

b.
```
name=('Beth')
print('Hello name')
```

c.
```
name=('Beth')
print('Hello'+name)
```

/2

Check if a statement about variables is true or false.

5. A variable name can start with a number. ☐ true ☐ false

6. The value of a variable can change. ☐ true ☐ false

7. A variable name can have a space. ☐ true ☐ false

8. The user can set the value of a variable. ☐ true ☐ false

9. A variable value can be a string or an integer. ☐ true ☐ false

/5

---

10. Complete the code with the correct Python command.

print	while	if	input

a. `name=` ___input___ `('What is your name?')`

b. ___print___ `('I like to code.')`

c. ___while___ `True:`

d. ___if___ `answer=='yes':`

/4

Turn each sentence into code with the correct logical operator. Pick from the options below:

==	!=	<	>

11. if the answer is yes, then show the word *great*

```
if answer == 'yes':
    print('great')
```

12. if the score is greater than 4, end the game

```
if score > 4:
    print('game over')
    break
```

/2

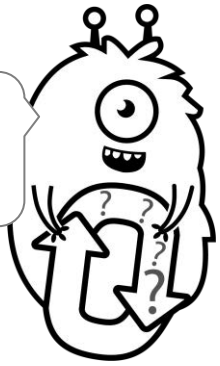
TOTAL: /20

Session 2 Skill Review: Loop the Questions to Try Again

The Pet Monster Rescue program helps people find their ideal pet. It does this by asking questions. If there is a match to a pet in the shelter, the person can take their new pet home. However, if there is no match, the person cannot adopt a pet.

Help people find a pet!

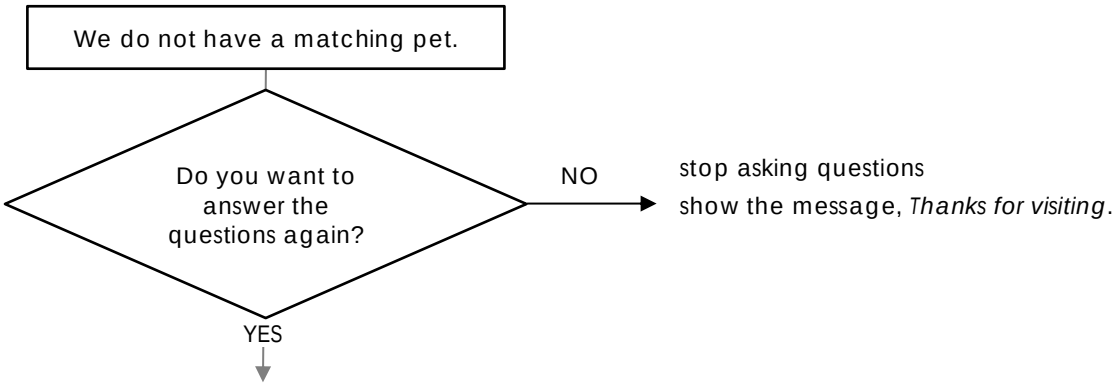
Create a loop that lets the person answer the questions again. They can change their mind about the horns, body, eyes, or arms. Now maybe they will find a match and get to take their new pet home.



You need to edit the logic in the *else* code.
Right now, it looks like this:

```
    else:
      find=False
      print('We do not have a matching pet.')
```

The *else* code needs to work like this instead:



Turn the Plan into Code

Complete the code to ask the person if they want to answer the questions again.

else	input	False	==

```
    else:
        print('We do not have a matching pet.')
        answer= input ('Do you want to answer the questions again? ')
        if answer == 'no':
            find= False
            print('Thanks for visiting.')
        else :
            print('We will help you find a pet.')
```

Loop the Questions

1. Open monster.py in IDLE (Python).

2. View *else* at the bottom of the program. Delete find=False.

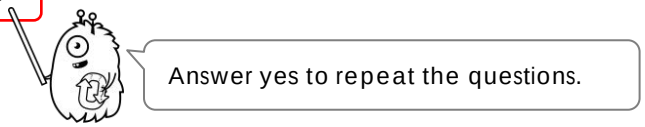
```
else:
  find=False
  print('We do not have a matching pet.')
```

delete

3. Refer to code on the previous page to edit the program.
   Ask the person if they want to answer the questions again.

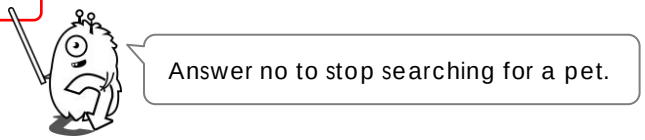
4. Test the program. Answer the questions so that there is NO MATCH.
   At the end, answer yes to have the questions asked again.

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? furry
How many eyes do you want your pet to have? 2
How many arms do you want your pet to have? 3
We do not have a matching pet.
Do you want to answer the questions again? yes


We will help you find a pet.
Do you want a pet with horns? yes or no
```



5. Test the program. Answer the questions so that there is NO MATCH.
   At the end, answer no to stop the loop.

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? furry
How many eyes do you want your pet to have? 2
How many arms do you want your pet to have? 3
We do not have a matching pet.
Do you want to answer the questions again? no


Thanks for visiting.
```



Answer Questions about the Program

1. Which line of code stops the program from looping?
   a. `else:`
   b. **`find=False`**
   c. `print('We do not have a matching pet.')`

2. Do you like the program more now that the person can answer the questions again?
   Why or why not?

Close Python

Session 2 Extension Activity: Open a Pet Monster Picture

The new pet owner may want to see a picture of their pet.

Python has libraries with special functions. The *webbrowser* library has commands to display web-based documents such as images or web pages on the Internet.

Follow the instructions to add code to open a picture of the pet when a person finds a match:

CODE	PURPOSE
`import webbrowser`	get commands from the webbrowser library
`webbrowser.open("https://www.website.com/image.png")`	open an image file on the Internet

Show a Pet Monster with Horns

1. Open monster.py in IDLE (Python).

2. Import the *webbrowser* library. Add the code to the first line of the program.

```
import webbrowser

#variables
place=('Pet Monster Rescue')
find=True
```

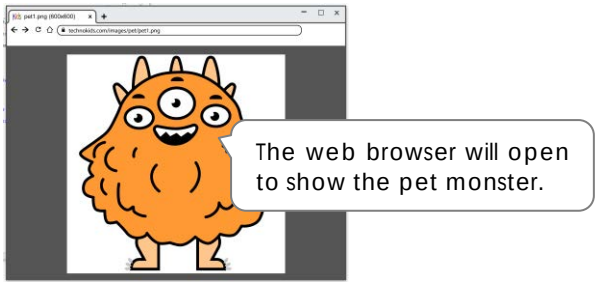
3. The first monster has horns. The web address with a picture of this pet is: https://www.technokids.com/images/pet/pet1.png

   Add the code: **`webbrowser.open("https://www.technokids.com/images/pet/pet1.png")`**

```
    horns=input('Do you want a pet with horns? yes or no ')
    if horns=='yes':
        print('We have a match.')
        print('It is Furhop. He likes to eat pizza and play ring toss.')
        webbrowser.open("https://www.technokids.com/images/pet/pet1.png")
        break
```

4. Test the program. When you are asked if you want a pet with horns, type yes.

```
Do you want a pet with horns? yes or no yes
We have a match.
It is Furhop. He likes to eat pizza and play ring toss.
```



5. Close the browser window. X

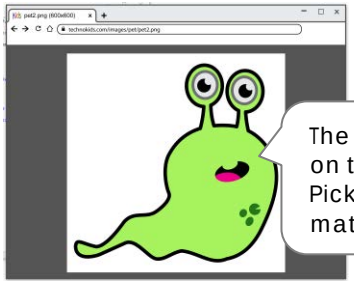
6. The second monster is not furry. The web address with a picture of this pet is: https://www.technokids.com/images/pet/pet2.png

   Add the code: **webbrowser.open("https://www.technokids.com/images/pet/pet2.png")**

```
body=input('What type of body do you like? slimy, scaly, or furry? ')
if body!='furry':
    print('You answered', body+'.')
    print('We have a match. It is Slimo. He likes to eat bugs and swim in mud.')
    webbrowser.open("https://www.technokids.com/images/pet/pet2.png")
    break
```

7. Test the program. When you are asked about the body, type slimy.

```
Do you want a pet with horns? yes or no no
What type of body do you like? slimy, scaly, or furry? slimy
You answered slimy.
We have a match. It is Slimo. He likes to eat bugs and swim in mud.
```



The pictures are in a *pet* folder on the TechnoKids website. Pick the ones you want to match your monsters.

8. Apply your skills to add a picture for each monster. Refer to the table below for the URL.

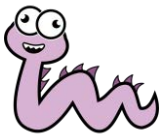

https://www.technokids.com/images/pet/pet1.png

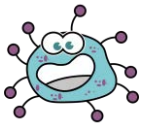

https://www.technokids.com/images/pet/pet2.png


https://www.technokids.com/images/pet/pet3.png

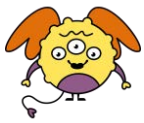

https://www.technokids.com/images/pet/pet4.png


https://www.technokids.com/images/pet/pet5.png


https://www.technokids.com/images/pet/pet6.png


https://www.technokids.com/images/pet/pet7.png


https://www.technokids.com/images/pet/pet8.png


https://www.technokids.com/images/pet/pet9.png


https://www.technokids.com/images/pet/pet10.png


https://www.technokids.com/images/pet/pet11.png


https://www.technokids.com/images/pet/pet12.png

Take the Challenge! Add a time delay between the description and picture.

```
import time
```

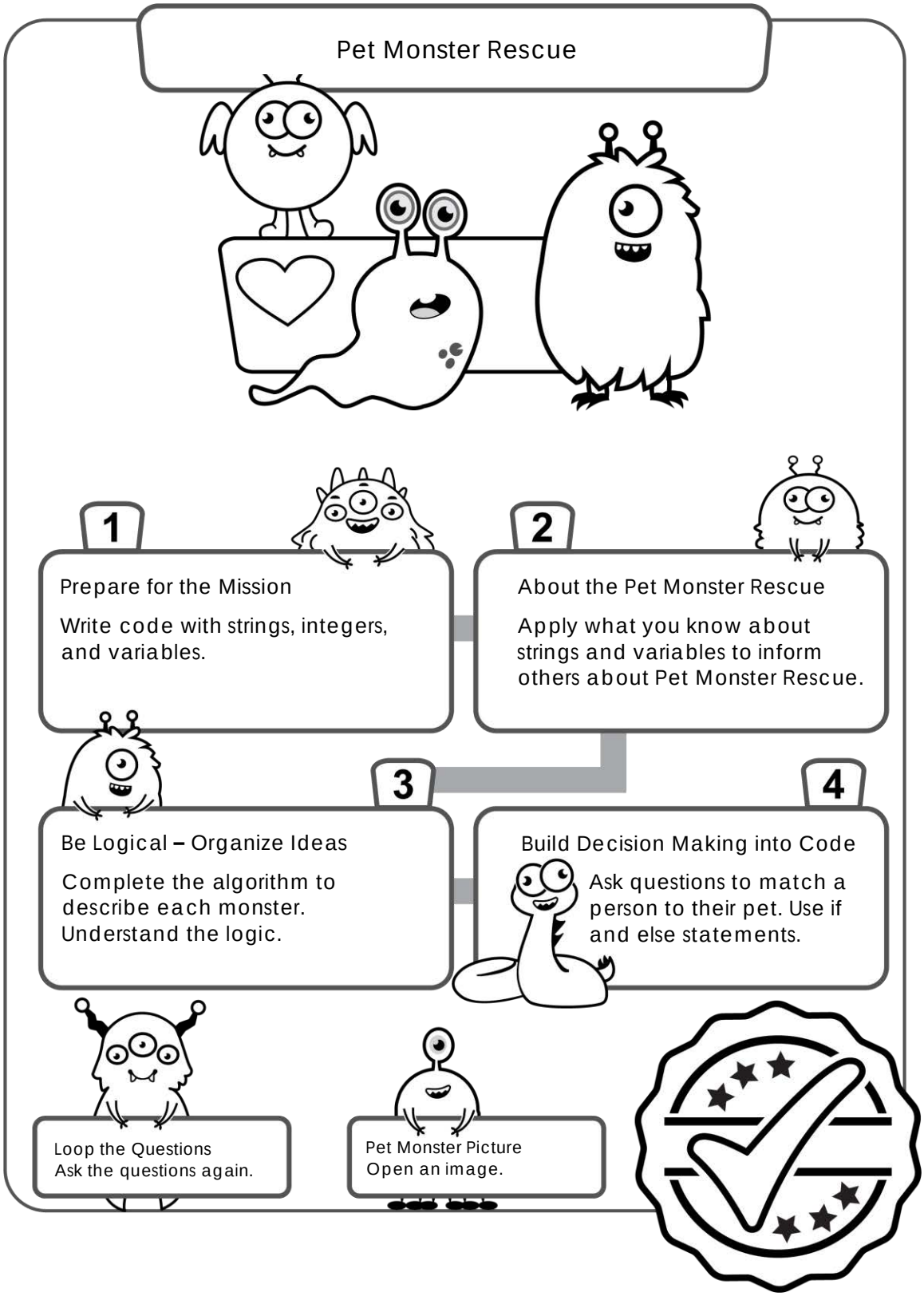
Where should this go? Start or end of the program?

```
time.sleep(5)
```

Add this line for each pet monster.

Session 2 Extension Activity: Pet Monster Rescue Mission

Great work! You have completed the Pet Monster Rescue Mission. To build the code was a four-part mission. Read each step to review the programming skills you have learned.



Programmers Are Logical

To complete the Pet Monster Rescue mission, you had to think logically. This is a valued trait in a programmer. Answer the questions to explain how being logical helped you finish the tasks.

1. A programmer carefully watches what is happening.

   How does testing a program to see the output help you to write better code?

2. A programmer pays attention to details.

   Pick one of the items from the list. Explain why it is important when writing code.

   ☐ spell each Python keyword correctly

   ☐ use the proper punctuation such as brackets, commas, or colons

   ☐ indent a block of instruction to group them together

   ☐ sequence instructions in the correct order

3. A programmer outlines their ideas clearly.

   In the Pet Monster Rescue Mission, you organized your ideas for each pet monster into a flowchart. How did your plan help you complete the program?

4. A programmer divides their ideas into smaller parts.

   When you wrote your program, you chunked the code into parts using comments. How will comments help others understand your program?

5. A programmer considers if a statement is true or false.

   You used logic in your program to match a person to a pet monster. Which type of logic did you find most difficult to understand.

   ☐ logical operators == != < >

   ☐ if and else statements

   ☐ instructions that run if a variable is True or False

   When did the logic become easy to understand? Or, are you still a bit confused?

Pet Monster Rescue Marking Sheet

Task: Design a program for the Pet Monster Rescue. Ask questions to match people to their ideal pet. Use logical operators to make decisions based upon the pet owner's answers.

Work with Strings and Variables	
User can input answers to questions that are stored as variables. Output combines text with variable values to create sentences. Each sentence has spaces between words and the proper punctuation.	/10
Control Actions using Logic	
Search for a pet is controlled using a while loop that switches from True to False. Program correctly finds a match using if statements and logical operators (==, !=, <, >). Questions continue if there is no match using an else statement.	/10
Produce an Original Program	
Text is easy to read. Program personalizes the pet adoption process by responding to user's input. Descriptions of pets are creative and original. (name, food, and play) Extra coding has been added to enrich the game design (formatted output, additional pet monsters, pictures of pets)	/10
TOTAL:	/30