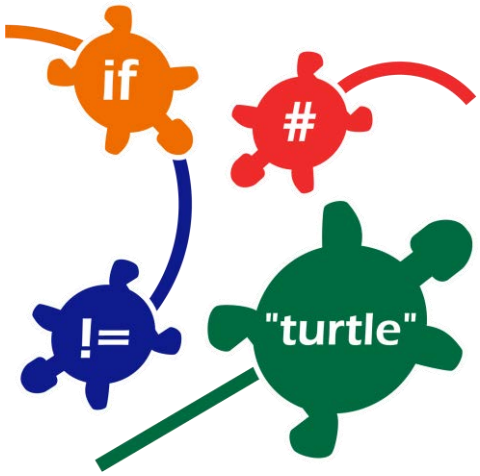
TECHNOTurtle

Teacher Guide

Lessons for Elementary School Students



Technology Course
using

Python

Become a game designer.

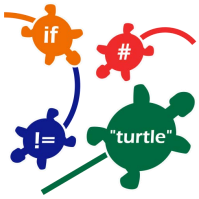
In this course, students become game designers. They use Python and the Turtle library to conquer mazes, paint pixel art, create a Mad Lib Generator, and build a Carnival Game. The fun begins when students edit code to gain an understanding of the structure of Python scripts. Once familiar with basic concepts, the young programmers are introduced to debugging, loops, variables, and conditional logic. Ignite an interest in programming with meaningful activities designed for beginners.

TECHNOKids®

Table of Contents

Session 4 Design Colorful Spirographs

Session 5 Create a Mad Lib Generator



Introduction

This section provides valuable information about teaching TechnoTurtle. It includes a description of the Teacher Guide, as well as an overview of the course. In addition, there are ideas for implementation and technology integration.

For additional guidance, open the course in TechnoHub and select Get Started to access preparatory steps, resource list, and scheduling timetable.

How to Use This Guide

This Teacher Guide contains the following:

Getting Started – This section contains a course description, as well as ideas for implementation.

Course Instructions – The course is comprised of six sessions, each focused on a problem-solving task that aligns with the project theme. Each session includes assignments that break down the task into manageable steps. The components of each session are as follows:

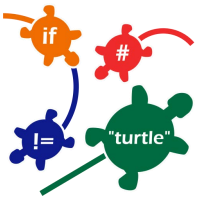
- Overview – An explanation of the session activities and their purpose.
- Materials – A list of handouts, sample files, templates, and teacher resource materials needed to teach the session.
- Teaching Strategies – Instructional methods recommended for teaching the activities.
- Lesson Plan – A detailed list of each step in the session.
- Learning Objectives – A summary of the content knowledge and technical skills taught throughout the session.
- Assignments – A session consists of assignments completed by students. Actions to be performed on the computer by the student are indicated with a triangle (▷). Background information is indicated with a dash (–).
- Review – A session review contains a list of fill-in-the-blank, multiple choice, or short-answer questions intended to review Python and Turtle commands (answers included).
- Skill Review – An additional assignment intended to review coding skills (includes completed sample).
- Extension Activity – An additional activity that relates to the problem-solving task presented in the session.

Appendices – This section contains additional information or materials including the following resources.

- Assessment Tools – Skill summary and marking sheets to evaluate Python projects.
- Reference Sheets – List of Turtle commands and enlarged image of Turtle canvas to support programming activities.
- Glossary – A definition of each term.
- Contact Information – How to contact TechnoKids Inc. for curriculum support.

TechnoTurtle Overview

Introduction to TechnoTurtle



In this course, students become game designers. They use Python and the Turtle library to conquer mazes, paint pixel art, create a Mad Lib Generator, and build a Carnival Game. The fun begins when students edit code to gain an understanding of the structure of Python scripts. Once familiar with basic concepts, the young programmers are introduced to debugging, loops, variables, and conditional logic. Ignite an interest in programming with meaningful activities designed for beginners.

Students complete the following tasks:

- In session 1, students become programmers. To start they learn how the programming language Python is used in daily life. Next, they visit the Turtle library to study the commands and make predictions about their function. They test their ideas by modifying a program to control what it draws. Once familiar with how to run a Python program, students add bugs to the code. This allows them to identify and fix common errors.
- In session 2, students control the movement of a Turtle through a series of mazes. The fun begins when the young programmers write their first script. It marches a Turtle around the screen by moving forwards, backwards, and turning. Once they have mastered this set of commands, students are challenged to develop a script that will guide a Turtle through a maze. Can they solve the puzzle?
- In session 3, students write code to draw pictures. To start, they learn how to plot a point on the canvas using x and y coordinates. They apply this knowledge to stamp a unique design. Next, the young programmers follow instructions to design a robot by combining lines, rectangles, circles, dots, and symbols. Once familiar with how to control the Turtle's drawing tools, students build their own program to draw a picture.
- In session 4, students paint stunning artwork. To start, they learn code that repeats a set of instructions forever or for a specific number of times. Next, they complete a series of exercises to discover how to construct looping geometric shapes called spirographs. Once students are familiar with designing patterns, they use the Random library to produce colorful creations.
- In session 5, students design a word game, called a Mad Lib. It has players provide a list of words that are used to complete a silly sentence or story. To prepare for this coding task, students learn about variables by chatting with the computer. Next, they edit a Mad Lib party invitation to discover how to join variables and text together to form sentences. Once familiar with the structure of the code, they program their own wacky word game.
- In session 6, students become game designers. They combine Python and Turtle programming commands to produce a Carnival Game. To start, they learn about if, elif, and else. Once familiar with conditional logic they invent a game that prompts the player to pick an option to win a prize. Optional challenges enrich the design such as looping a flashing message or showing a picture of their winnings. Get ready for fun. Step right up to win a prize!

Implementation and Technology Integration Ideas

TechnoTurtle introduces the text-based Python programming language to elementary and middle school students. They apply coding skills to solve puzzles, design artwork, and build games. It is an ideal course for Grades 3 and up.

Ideas for Implementation

Have your students become programmers using the fun activities in the TechnoTurtle course. 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:

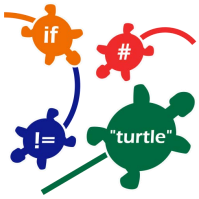
- *STEM Project and a Computer Science Class*: This option is best suited to teachers that have a chunk of instructional time allocated to STEM. TechnoTurtle has 30 assignments that systematically develop programming skills. The young programmers will learn how to divide a task into steps, sequence instructions, and debug code. In addition, they become aware of how to import libraries, loop instructions, store variables, and apply conditional logic to trigger actions. TechnoTurtle 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 sequence instructions to build simple scripts. Experimentation is encouraged to help young programmers understanding Python functions. As well, there are templates to jump start coding. The activities are perfect for Grades 3 and up.
- *Coding Unit for Advanced Beginners:* Once students understand the basics of programming, they extend their learning in Sessions 4-6. The programming tasks become more complicated. They include loops, variables, and conditional logic. The activities are ideal for students that understand the fundamentals and are ready for a challenge.
- *Hour of Code*: If you only have one class to teach the Python programming language there are many assignments in TechnoTurtle 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. If your students have existing knowledge, another option is the maze activity in Session 2.
- *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. Consider the age range and coding abilities of students. Pick activities they will enjoy. For example, complete a dot-to-dot, edit a Mad Lib party invitation, or design a Guess the Number game.

Technology Integration Suggestions

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

- *Mathematics Problem Solving Unit*
  Connect mathematics to programming. For example, you can teach angles and measurement by solving mazes. Graph ordered pairs by plotting objects on a canvas. Or, apply conditional logic to code a game that awards prizes. TechnoTurtle has many activities that combine computational thinking with mathematical reasoning.
- *Language Arts*: Include Python as part of a unique language arts unit. The Mad Lib Generator built in Session 5 is a word game. Players input answers that form silly sentences. This programming activity transforms parts of speech including noun, verbs, adjectives, and adverbs into variables.
- *Social Studies*: Transfer map making skills to a new task. The assignments in Session 2 have students apply knowledge of directions (left, right, up, and down) to move a robotic Turtle from one location to another.
- *Visual Arts*: Infuse programming into visual arts. The assignments in Session 3 experiment with the Turtle's drawing tools. Students explore how to combine lines, shapes, and symbols to draw pictures. The assignments in Session 4 expand upon this skillset when loops are used to produce colorful spirographs and random geometric patterns. These activities blend computer science with artistic expression.
- *Computer Science*: TechnoTurtle is an introduction to text-based Python programming. The course includes a Summary of Skills that details computer science learning outcomes. Students learn how to debug, loop a set of instructions, manipulate variables, and apply conditional logic.

This is a preview of the teacher guide.
Pages have been omitted.



Session 2
Conquer the Maze

In this session, students control the movement of a Turtle through a series of mazes. The fun begins when the young programmers write their first script. It marches a Turtle around the screen by moving forwards, backwards, and turning. Once they have mastered this set of commands, students are challenged to develop a script that will guide a Turtle through a maze. Can they solve the puzzle?

Assignment 6: Tell the Machine What to Do

Assignment 7: Ready, Set, Go!

Assignment 8: Create Line Drawings

Assignment 9: Move Through the Maze

Assignment 10: Solve the Puzzle

Session 2 Review: Pick the Command to Do the Job

Session 2 Skill Review: Program the Robot

Session 2 Extension Activity: Dot-to-Dot Fun

Session 2 Getting Started

Overview

In this session, students control the movement of a Turtle through a series of mazes. The fun begins when the young programmers write their first script. It marches a Turtle around the screen by moving forwards, backwards, and turning. Once they have mastered this set of commands, students are challenged to develop a script that will guide a Turtle through a maze. Can they solve the puzzle?

Materials

- IDLE Python 3
- Maze folder - Python background images: maze.gif, zoo.gif, zoo2.gif
- Python samples:
  - maze.gif, maze.py
  - zoo.gif, zoo.py or zoo2.gif, zoo2.py
- Turtle Reference Sheet (optional)
- Coding Journal Reflection (optional)
- Session 2 Review: Pick the Command to Do the Job
- Session 2 Skill Review: Program the Robot
  - farm.gif, farm_solution.py
- Session 2 Extension Activity: Dot-to-Dot Fun
  - gift.gif, gift_solution.py
  - kite.gif, kite_solution.py
  - ghost.gif, ghost_solution.py

Teacher Preparation

(Refer to the Preparing to Teach section of this guide for instructions)

- Make the files in the Turtle folder available to students. It has the *maze* folder inside it that is required for Assignments 9-10 and the session's Skill Review. Students must copy the *maze* folder to the place where they save their work. They will then save the Python files into their personal *maze* folder throughout the session.
- View the Python samples to gain an understanding of the completed mazes.

Teaching Strategy

In this session, students solve puzzles by moving a Turtle through a maze. Explain scenario:

> *In this session, you program a robotic Turtle to move through a maze. To start, you learn how moving robots are used to help people. Next, you write your first Python program using the Turtle library of commands. You discover how to maneuver forwards, backwards, left, and right. Once these coding skills are mastered, you insert a picture of a maze. You then sequence instructions to guide a Turtle along a path from the start to the end. Can you help the Turtle swim home? Or can you help the Turtle find his friend at the zoo? Yes, you can!*

Assignment 6: Tell the Machine What to Do

In this assignment, students learn how moving robots are used to help people. They read about programs that tell a machine how to do a repetitive task or respond to commands in real-time. Afterwards, they answer questions that require them to apply computational thinking. In one task they must sequence commands to paint a stripe. In the other task they use symbols to plan the flight path of a drone.

The goal is to have students begin to form a connection between the real-world application of programmable robots and moving a robotic Turtle using Python.

Introduce the following terminology:

- *algorithm*: A description of what a program should do using words or symbols.

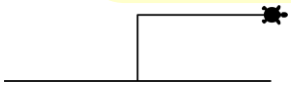
Assignment 7: Ready, Set, Go!

In this assignment, students write their first Python program. They import the Turtle library and then practice moving and turning the robotic Turtle around the screen. Encourage students to experiment with a different number of steps and various angles. This will allow them to gain an understanding of how to program movement.

Prior to beginning have students pretend to be the robotic Turtle. Give instructions to move around the classroom forwards, backwards, left, and right. The steps are like what they will code. At this time, you may want to review angles such as 45°, 90°, 180°, and 360°.

1. Walk forward 1 step.
2. Walk backward 4 steps.
3. Walk forward 2 steps.
4. Turn left or counterclockwise 90°.
5. Walk forward 1 step.
6. Turn right or clockwise 90°.
7. Walk forward 2 steps.

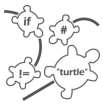
```
forward(100)
backward(400)
forward(200)
left(90)
forward(100)
right(90)
forward(200)
```



Have fun! You may want to have students change their speed from slow to fast.

Afterwards, introduce the following commands:

from turtle import *	Import all the commands in the Turtle library.
speed("slowest")	Adjust the drawing speed.
shape("turtle")	Change the shape of the symbol used to draw.
forward(steps)	Direct the Turtle symbol to move ahead a number of steps.
backward(steps)	Direct the Turtle symbol to move back a number of steps.
left(angle)	Turn the Turtle symbol counterclockwise by setting the degree.
right(angle)	Turn the Turtle symbol clockwise by setting the degree.



TIP: There are shorter ways to write the commands to move the Turtle. The complete words are used to emphasize that Python coding makes sense and is easy to read. However, if your students are struggling with their keyboarding skills you may wish to introduce the alternate commands. The shorter versions are introduced in the Session 2 Extension Activity:

`forward( )`	`backward( )`	`left( )`	`right( )`
or	or	or	or
`fd()`	`bk()`	`lt()`	`rt()`

Assignment 8: Create Line Drawings

In this assignment, students practice their programming skills. They write a sample program to see what letter it makes. Afterwards, they complete a program to draw a zig zag line. This is a great activity if your students need extra practice. If time is an issue, you may want to complete it as a group activity. Or, assign it to programmers that completed Assignment 7 quickly.

Assignment 9: Move Through the Maze

In this assignment, students program a robotic Turtle to move through a maze. The maze is an image file that is inserted onto the canvas. For this activity to work successfully the maze.gif file and the Python file must be in the SAME folder.

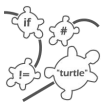
To build the code, students are introduced to *Trial and Error* as a debugging strategy. Trial and Error means that you test different ideas until you find one that works.

Offer the following Trial and Error tips:

- *Slow it down*: Set the speed to slowest. For example: `speed("slowest").` This will allow you to see each command in action. This will make it easier to pinpoint the problem.
- *Write one line of code at a time*: Do not try to write the entire program at once. Instead, program a line at a time – then test it. This will allow you to get each movement correct before programming the next action.
- *Make an educated guess*: Do not just randomly assign the number of steps or angle. For example: `forward(100) right(90).` Instead, be thoughtful. Look at the result of the number you picked. If the Turtle goes too far use a lower value. If it moves too little use a higher value.
- *Consider the size of the background image*: The maze is 600 pixels wide by 600 pixels tall. For this reason, the number of steps cannot be greater than 600 as the Turtle would then move off the maze. Use the image size as a guide when selecting values.
- *Notice the direction of the Turtle's head*: The turtle always moves forward in the direction it faces. When it first appears on the canvas, it faces right.

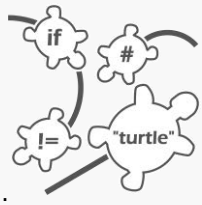
Introduce the following command:

bgpic("*picturename*.gif")	Apply an image to the background of the canvas. The name of the picture goes in quotes. The image must be a gif.



TIP: You may want students to take a screenshot of their solved maze and submit the image with their Python program.

Session 2 Skill Review: Program the Robot



In this activity students program a robot to do chores for the farmer. The program is like the drone algorithm written in Assignment 6. The instructions sequence movement from one location to another. This activity can be used prior to teaching Assignment 9 to demonstrate how to insert a background image. Or, it can be used as a review to practice coding.

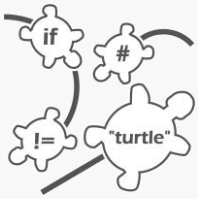
Assignment 10: Solve the Puzzle

In this assignment, students select a maze and program a solution. There are two options – zoo.gif is a bit easier than zoo2.gif due to the angles. Prior to beginning, a checklist of programming skills is listed. Have students reflect upon their learning. Afterwards, they should select a maze that matches their skillset.

Helpful Resources:

- Solutions are available in the Turtle Resources/samples/mazes folder.
- Distribute the *Turtle Reference Sheet* for a summary of Turtle movements.
- Use the *Coding Journal Reflection* sheet to have students reflect upon the coding experience.

Session 2 Extension Activity: Dot-to-Dot Fun



Want a fun challenge? Program a dot-do-dot.
There are three images. Each one has dots that must be connected to complete the picture. They are different levels of difficulty:

- gift.gif – easy – box made from four lines with 90° turns
- kite.gif – medium – diamond made from four lines with acute and obtuse angles
- ghost.gif – hard – bowtie made from five different sized lines and angles

Remind students that the image file and Python file must be in the SAME folder.

Lesson Plan

Assignment 6: Tell the Machine What to Do

- Read about programmable robots.
- Write an algorithm to help a robot to move:
  - Sequence the steps to complete the algorithm.
  - Plan the flight path of a drone using arrow symbols.

Assignment 7: Ready, Set, Go!

- Open IDLE and create a new Python file.
- Import the Turtle library.
- Move the Turtle forwards and backwards.
- Set the speed to "slowest" and the shape to "turtle".
- Turn the Turtle left and right. Explore using angles to change the shape of the line.
- Experiment with moving the Turtle.
- Close the program.

Assignment 8: Create Line Drawings

- Open IDLE and create a new Python file.
- Read the program. Guess the letter it will make.
- Build the program and run it to reveal the letter.
- Look at the line. Complete the program to replicate the shape.
- Close the program.

Assignment 9: Move Through the Maze

- Have EACH student copy the *maze* folder to the place where they save their work.
- Open IDLE. Create a new Python file. Save it in the copied *maze* folder.
- Start to build a program that imports the library, sets the speed, and Turtle shape.
- Apply the *maze.gif* background picture to the canvas.
- Program the Turtle to move to the end of the maze.
- Use Trial and Error to debug the code.
- Close the program.

Assignment 10: Solve the Puzzle

- Complete the checklist to recognize programming skills.
- Copy the *maze* folder if this step was not completed in Assignment 9.
- Open IDLE. Create a new Python file. Save it in the copied *maze* folder.
- Start to build a program that imports the library, sets the speed, and Turtle shape.
- Apply the *zoo.gif* or *zoo2.gif* background picture to the canvas.
- Program the Turtle to move to the end of the maze.
- Close the program.

Learning Objectives

Programming Basics

- recognize that programs are used to tell a machine how to do a task
- sequence the steps in an algorithm
- write an algorithm using symbols
- infer the output of a program after reading the code
- sequence the instructions in a program
- pick the correct command to do a task
- use the proper punctuation for each command
- apply Trial and Error as a debugging strategy
- design a program that achieves a specific goal

Python Basics

- create a new Python program
- interpret code using the Python Shell to generate a program's output

Turtle Functions

- import all the commands in the Turtle library
- view a drawing on the Turtle canvas
- move the Turtle forward and backward a specific distance
- turn a Turtle left or right by setting the angle of rotation
- adjust the drawing speed of the Turtle
- modify the shape of the Turtle
- save a Python file and picture file in the same location to display a background image
- apply an image to the background of the canvas
- set the pen size to control line thickness (optional)

Applied Technology

- experiment with different values to control the movement of a Turtle
- develop code that will draw a specific shape
- build a Python program to move a Turtle through a maze
- program the Turtle's movement to complete a dot-to-dot picture (optional)

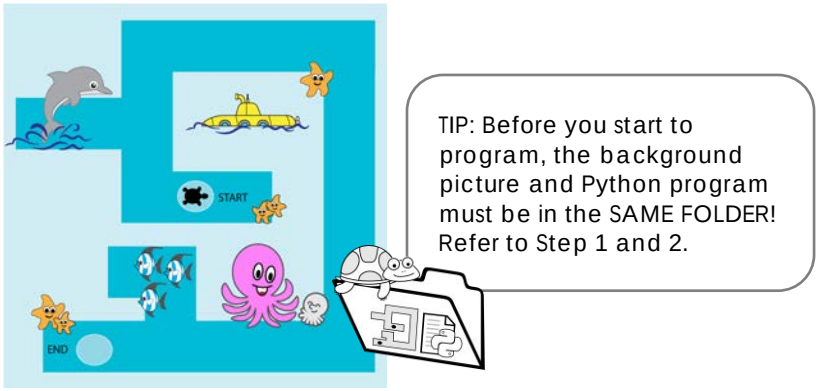
This is a preview of the teacher guide.
Pages have been omitted.

Assignment 9 Move Through the Maze

Programmers have a goal when they design a program. They know what they want the outcome to be.

You are a programmer! Your goal is to move the Turtle through the maze. You must write instructions to go from the start to the end point. The Turtle must stay inside the path!

Help the Turtle swim home.



This image is in the maze folder.

Copy the Maze Folder

The maze is a picture. You will add it to the background using `bgpic("picture.gif" )`. To have it display, the picture MUST BE in the same place as the maze Python file.

1. ▷ Open the *Turtle* folder. (Ask your teacher where this is located.)

   ▷ Copy the *maze* folder and paste it where you save your work.

Save a Python File into the Maze Folder

2. ▷ Open IDLE (Python).

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

   ▷ From the File menu, select *Save*.

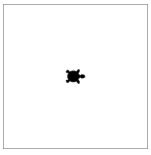
   ▷ Go the *maze folder* you copied in Step 1.
   Type swim as the file name. Click *Save*.

Start the Program

3. ▷ Import the Turtle library. Set the speed to slow. Make the symbol look like a Turtle:

```
from turtle import *
speed("slowest")
shape("turtle")
```

   ▷ Apply your skills to run the program.
   A Turtle should be on the canvas.

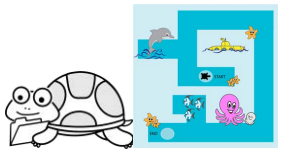


Apply a Background Picture to the Canvas

4. ▷ Add the maze image to the background using the command `bgpic("maze.gif")`:

```
from turtle import *
speed("slowest")
shape("turtle")
bgpic("maze.gif")
```

If you cannot see the maze, the Python file and picture are not in the same folder.



▷ Apply your skills to run the program.
The Turtle will be at the start of the maze.

Use Trial and Error as a Debugging Strategy

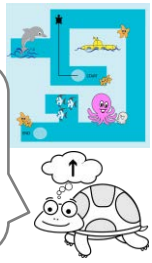
5. When you run a program, the result might not be what you want. One method to fix the problem is *Trial and Error*.

Trial and Error means that you test different ideas until you find one that works. After each *trial* you study the *error* you get. This lets you come up with a better idea to try next.

Start to move the Turtle through the maze.
How much do you need to move the Turtle forward? Use Trial and Error to find out!

```
from turtle import *
speed("slowest")
shape("turtle")
bgpic("maze.gif")
backward(80)
left(90)
forward(?)
```

If the turtle moves too far use a lower number. If it moves too little use a higher number. Keep trying numbers until the turtle moves forward to the top of the path.



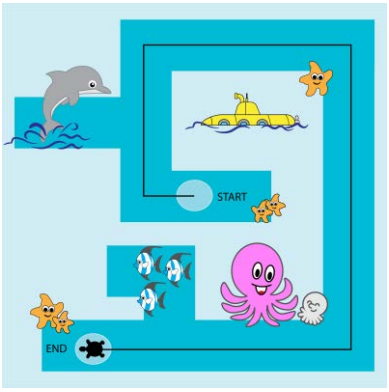
Program the Turtle to Move to the End of the Maze

6. Apply your skills to move the Turtle to the end of the maze. Use the following commands:

```
forward( )    backward( )      left( )    right( )
```

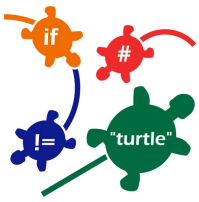
HINTS:

- The maze is 600 pixels wide by 600 pixels tall.
- The number you pick should always be less than 600.
- Compare the path to a part you have already done. Is the path shorter or longer? This will help you pick a number to use.
- Pretend your body is the turtle. Point it in the direction the turtle faces. Do you need to turn right or left to move along the path?
- Write one line of code at a time – then test it. This will allow you to focus on each movement.



Close the Program

This is a preview of the teacher guide.
Pages have been omitted.



Session 3
Draw Pictures

In this session, students write code to draw pictures. To start, they learn how to plot a point on the canvas using x and y coordinates. They apply this knowledge to stamp a unique design. Next, the young programmers follow instructions to design a robot by combining lines, rectangles, circles, dots, and symbols. Once familiar with how to control the Turtle's drawing tools, students build their own program to draw a picture.

Assignment 11: Get to Know the Canvas

Assignment 12: Stamp Around

Assignment 13: Draw a Robot

Assignment 14: Program a Picture

Session 3 Review: About the Turtle Canvas and Library

Session 3 Skill Review: Picture Perfect

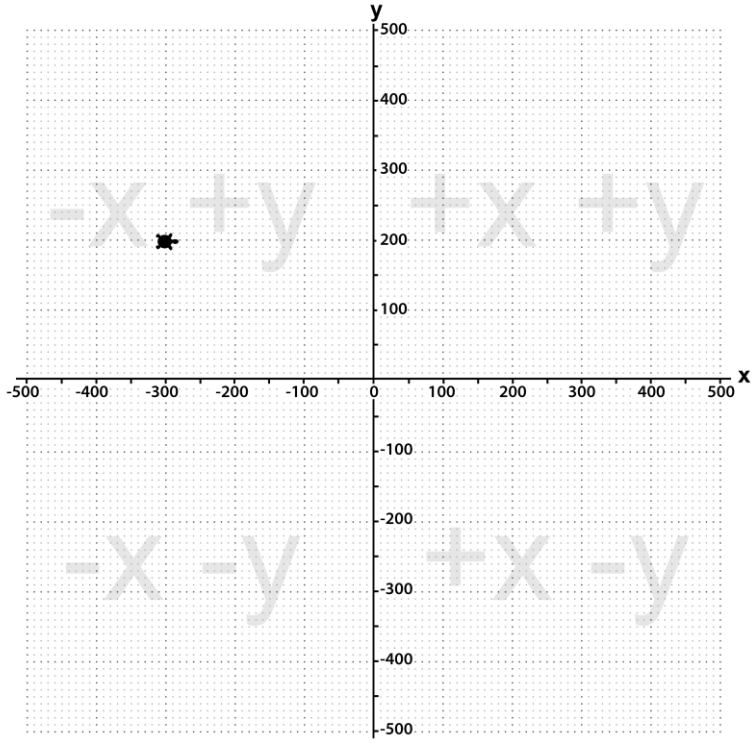
Session 3 Extension Activity: Customize the Stamp

This is a preview of the teacher guide.
Pages have been omitted.

Session 3 Review: About the Turtle Canvas and Library

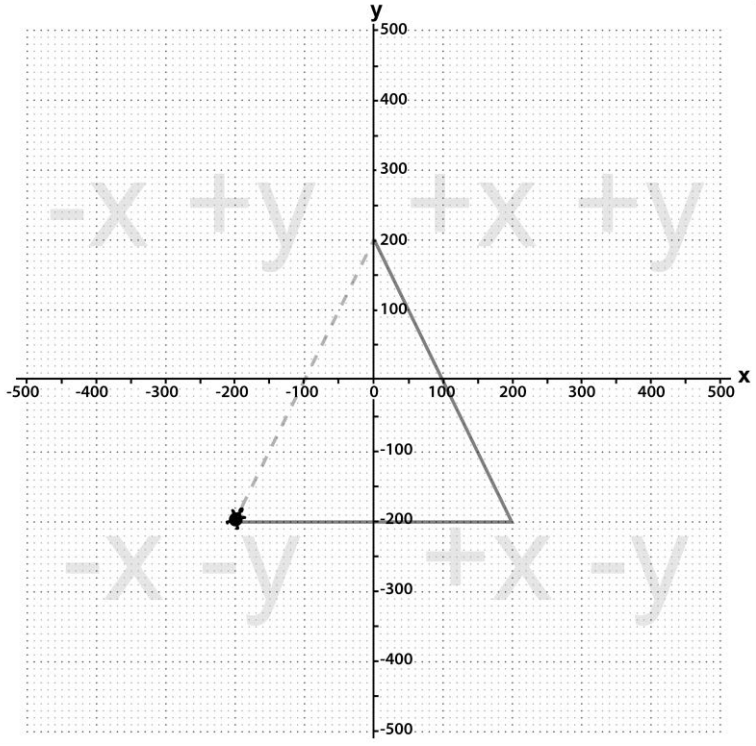
About the Turtle Canvas

1. Look at the Turtle on the canvas. Where is it?



- **x: -300 y: 200**
- x: 200 y: 300
- x: -300 y: -200

2. Move the Turtle to draw a triangle. What are the x and y values in the last line of code?




```
#triangle
penup()
goto(0, 200)
pendown()
goto(200, -200)
goto(-200, -200)
goto(x, y)
```

- goto(-100, 200)
- **goto(0, 200)**
- goto(0, 400)

Pick the Command That Will Complete the Task

3. You want to set the shape of the Turtle symbol:

   a. **`shape("circle")`**

   b. `pensize(10)`

   c. `dot()`

4. You want to make the outline of a shape red:

   a. `pendown()`

   b. `fillcolor("red")`

   c. **`pencolor("red")`**

5. You want to stop drawing a line:

   a. `goto(0, 0)`

   b. **`penup()`**

   c. `stamp()`

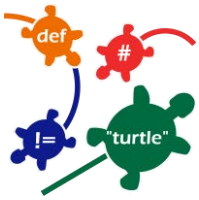
6. Write the command that does the task:

```
setheading(90)    pendown()    circle(50)    begin_fill()    goto(100, 50)
```

a.	**`goto(100, 50)`**	Move the Turtle to a specific spot.
b.	**`circle(50)`**	Create a circle.
c.	**`pendown()`**	Draw a line.
d.	**`begin_fill()`**	Tell the computer you are making a shape to fill.
e.	**`setheading(90)`**	Set the Turtle symbol to face up.

TOTAL: /10

This is a preview of the teacher guide.
Pages have been omitted.



Session 4
Design Colorful Spirographs

In this session, students paint stunning artwork. To start, they learn code that repeats a set of instructions forever or for a specific number of times. Next, they complete a series of exercises to discover how to construct looping geometric shapes called spirographs. Once students are familiar with designing patterns, they use the Random library to produce colorful creations.

Assignment 15: Explore Loops

Assignment 16: Loop to Paint Patterns

Assignment 17: Surprise the Viewer with Random Creations

Session 4 Review: About Loops and the Random Library

Session 4 Skill Review: Design a Spirograph

Session 4 Extension Activity: It is Raining Cats and Dogs

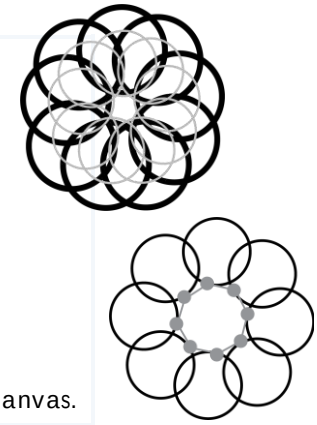
This is a preview of the teacher guide.
Pages have been omitted.

Assignment 16 Loop to Paint Patterns

You already know how to create lines, shapes, and symbols using the Turtle library of commands. In this assignment, you apply this knowledge to paint beautiful looping patterns.

Here are the drawing commands you will combine to create artwork:

circle(size)	Draw a circle that is a specific size.
forward(steps)	Move ahead a number of steps.
right(angle)	Turn clockwise, which is right.
pensize(thickness)	Set the size of the pen outline.
pencolor("color")	Set the color of the pen.
fillcolor ("color")	Set the fill color of a stamp.
shape("turtle")	Change the shape of the symbol.
stamp ( )	Put a mark of the Turtle symbol on the canvas.



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 pattern as the file name. Click *Save*.

2. Import the Turtle library and set the screen size:

```
from turtle import *
screensize(2000, 2000)
```

Draw Circles Forever

3. ▷ Draw circles non-stop:

```
#draw a pattern
while True:
    circle(25)
```

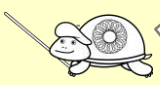
   ▷ Save the program. From the Run menu, select *Run Module* to see it loop.

   ▷ Close the canvas and Python Shell.

Draw Circles in a Pattern

4. ▷ Make circles around the canvas by moving forward and setting the angle:

```
#draw a pattern
while True:
    circle(25)
    forward(5)
    right(25)
```



Experiment with the values. What happens if forward is 10 or right is 62?

Set the Pen Size and Color

5. Make the pattern look fancy. Set the pen size. Pick a color from the list:

```
#draw a pattern
while True:
    pensize(5)
    pencolor("blueviolet")
    circle(25)
    forward(5)
    right(25)
```

orangered	aqua
violet	deepskyblue
blueviolet	lightpink

Find more color names:
https://www.w3schools.com/colors/colors_names.asp

▷ Run the program again.

Add a Stamp to the Pattern

6. Create a unique design by setting the stamp shape and fill color:

```
#draw a pattern
while True:
    pensize(5)
    pencolor("blueviolet")
    circle(25)
    forward(5)
    right(25)
    fillcolor("aqua")
    shape("turtle")
    stamp()
```

triangle	Turtle
classic	square
arrow	circle

▷ Run the program again.

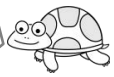
Get Creative!

7. Pick an idea to create a colorful pattern:

Pick the pen up and move, to create a wider circle pattern:

```
#draw a pattern
while True:
    pensize(5)
    pencolor("blueviolet")
    circle(25)
    forward(5)
    right(25)
    fillcolor("aqua")
    shape("turtle")
    stamp()
    penup()
    forward(50)
    pendown()
```

Near the top of the code, set speed: speed("fastest")



Copy and paste the loop instructions. Change the values:

```
#draw a pattern
while True:
    pensize(5)
    pencolor("blueviolet")
    circle(25)
    forward(5)
    right(25)
    fillcolor("aqua")
    shape("turtle")
    stamp()
    pensize(2)
    pencolor("navy")
    circle(20)
    forward(45)
    right(62)
    fillcolor("magenta")
    shape("circle")
    stamp()
```

Edit the code to set the number of loops. Go to a new spot to paint:

```
#draw a pattern
for shape in range(15):
    pensize(5)
    pencolor("blueviolet")
    circle(25)
    forward(5)
    right(25)

penup()
goto(100, 300)
pendown()

#draw a new pattern
for newshape in range(36):
    pensize(3)
    pencolor("seagreen")
    circle(40)
    forward(25)
    right(62)
```

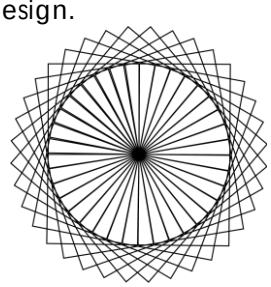
press BACKSPACE to stop indenting

Close the Program

This is a preview of the teacher guide.
Pages have been omitted.

Session 4 Skill Review: Design a Spirograph

You can create a spirograph using squares. You will use loops to make the design.



1. a. Open IDLE (Python).

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

   c. From the File menu, select *Save*.

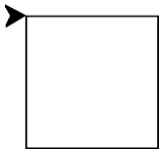
   d. Type spiro as the file name. Click *Save*.

2. Import the Turtle library. Set the speed and screen size.

```
from turtle import *
speed("fast")
screensize (2000, 2000)
```

3. Use a *for* loop to repeat instructions to draw each line in a square. Run the program.

```
#draw square
for line in range(4):
    forward(100)
    right(90)
```



4. Draw squares forever using a *while* loop.

```
#loop forever
while True:

#draw square
for line in range(4):
     forward(100)
     right(90)
```

5. Indent the square code to make it part of the *while* loop:

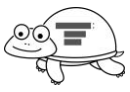
   a. Select the square code:

```
#loop forever
while True:

#draw square
for line in range(4):
    forward(100)
    right(90)
```

   b. Press TAB on the keyboard to move it over. Run the program.

```
while True:

    #draw square
    for line in range(4):
        forward(100)
        right(90)
```

Indent the code below `while True:` so that it becomes part of the forever loop.

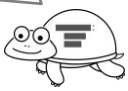


6. To make the spirograph, angle the pen, before drawing a new square:

   a. Place the cursor at the end of the *while True* line. Press ENTER on the keyboard.

```
while True:
    |

    #draw square
    for line in range(4):
        forward(100)
        right(90)
```

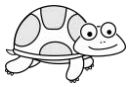
The cursor must be at the same indent level as `for line in range`. This will make the command part of the *while* loop, not the square loop.



   b. Turn right to change the angle. Run the program.

```
while True:
    right(62)
    #draw square
    for line in range(4):
        forward(100)
        right(90)
```

The angle should be a value that cannot be divided evenly into 360. For example, 19, 38, 47, or 62.



7. Import the Random library:

```
from turtle import *
import random
speed("fast")
```

8. Above the *while* loop, create a *linecolor* variable with a list of colors:

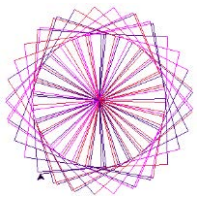
```
#random colors
linecolor=("crimson", "fuchsia", "indigo")

#loop forever
while True:
    right(62)
```

olive	turquoise
teal	orange
tomato	lime

Find more color names:

9. Make each line a different color using `random.choice` and the *linecolor* variable name:

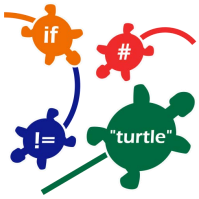
```
#random values
linecolor=("orange", "plum", "teal")

while True:
    right(62)
    #draw square
    for line in range(4):
        pencolor(random.choice(linecolor))
        forward(100)
        right(90)
```



10. Run the program to see each line change color.

11. Close the program.

This is a preview of the teacher guide.
Pages have been omitted.

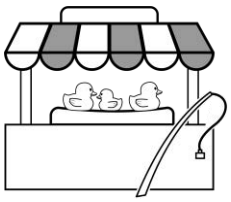


Session 6
Invent a Carnival Game

In this session, students become game designers. They combine Python and Turtle programming commands to produce a Carnival Game. To start, they learn about if, elif, and else. Once familiar with conditional logic they invent a game that prompts the player to pick an option to win a prize. Optional challenges enrich the design such as looping a flashing message or showing a picture of their winnings. Get ready for fun. Step right up to win a prize!

This is a preview of the teacher guide.
Pages have been omitted.

Assignment 27 Plan a Carnival Game



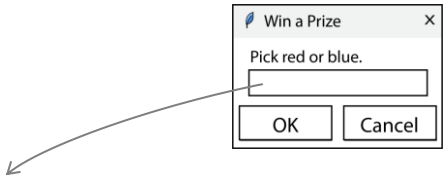
You are going to become a game designer. Read to learn how logic will be used to award the player a prize. Afterwards answer the questions to form a plan for your own Carnival Game. Will it be a ring toss, dart throw, bottle stand, or spin-the-wheel? You decide!

About the Carnival Game

To start, the player will read about the game that they are playing. For example:

At the Carnival
You are playing the Duck Pond game.
Which duck will you fish from the pond to win?

The player will be given two choices. The choice will be stored as a variable:




```
choice=textinput("Win a Prize", "Pick red or blue.")
```

About the Logic

The program will use logic to give away prizes. It checks to see if the player's choice matches.

```
if choice=="red":
    write("You win a toy bear")
```

If the choice is *red*, then the player wins a toy. The program stops looking for a match.

NO ↓
The player's choice does not match. The program looks elsewhere to see if there is a match in the next line of code.

```
elif choice=="blue":
    write("You win a pink balloon")
```

Else if the choice is *blue*, then the player wins a balloon. The program stops looking for a match.

NO ↓
Everything else fails – there is no match. The program does the action in the next line of code.

```
else:
    write("You did not win a prize.")
```

Else if the player's choice has no match then they win nothing.

Step Right Up to Win a Prize

1. Pick a game or come up with your own idea:

   ☐ Duck Pond ☐ Ring Toss ☐ Dart Throw

   Other: _____

2. Describe the game. How does the player win a prize?

3. What are the player's choices?

   choice 1: _____

   choice 2: _____

   The code will look something like this:

```
#play game
choice=textinput("Win a Prize", "Pick choice1 or choice2 to win a prize")
```

4. What happens if the player picks choice 1?

5. What happens if the player picks choice 2?

6. What happens if the player does not pick choice 1 or 2?

```
#prizes
if choice="    ":
     what is the prize?


elif choice="    ":
     what is the prize?


else:
     what happens?
```

Assignment 28 Become a Game Designer

Design a Carnival Game. The instructions provide an example only. You must adjust the code to match your plan from Assignment 27.

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 carnival as the file name. Click *Save*.

2. Import the Turtle library and hide the Turtle symbol:

```
from turtle import *
hideturtle()
```

Set the Game Window

3. ▷ Add the Game Name to the title bar and set the canvas color:

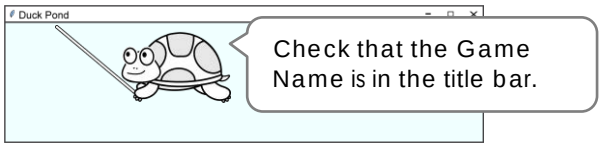
```
#game window
title("Duck Pond")
bgcolor("azure")
```

Color Names

cyan	fuchsia	gold

Find more color names:
https://www.w3schools.com/colors/colors_names.asp

   ▷ Save the program. From the Run menu, select *Run Module* to see the game window.



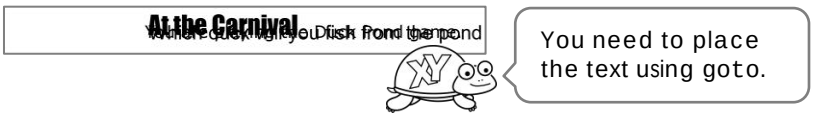
   ▷ Close the canvas and Python Shell.

Describe the Game to Players

4. ▷ Add a title and describe how to play. Format the text. For example:

```
#about game
write("At the Carnival", font=("Impact", 40))
write("You are playing the Duck Pond game.", font=("Verdana", 18))
write("Which duck will you fish from the pond to win?", font=("Arial", 20))
```

   ▷ Run the program to check the text.

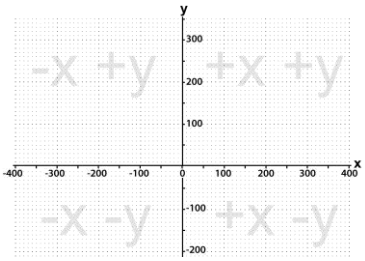


   ▷ Close the canvas and Python Shell.

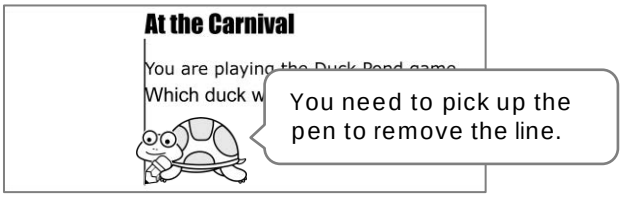
Place the Text on the Canvas

5. ▷ Where should the text go on the canvas? Use `goto(x, y)` to place it:

```
#about game
goto(0, 300)
write("At the Carnival", font=("Impact", 40))
goto(0, 200)
write("You are playing the Duck Pond game.", font=("Verdana", 18))
goto(0, 150)
write("Which duck will you fish from the pond to win?", font=("Arial", 20))
```



▷ Run the program to check the text.



▷ Close the canvas and Python Shell.

▷ Add `penup()` to the code to remove the line. Where should the code go?

Align the Text and Format the Color

6. ▷ Align the text. For example:

```
write("At the Carnival", align="center", font=("Impact", 40))
```

▷ Get creative! Set the pen color to format the words. For example:

```
#about game
goto(0, 300)
pencolor("darkblue")
write("At the Carnival", align="center", font=("Impact", 40))
```

Color Names

teal	slategray	navy

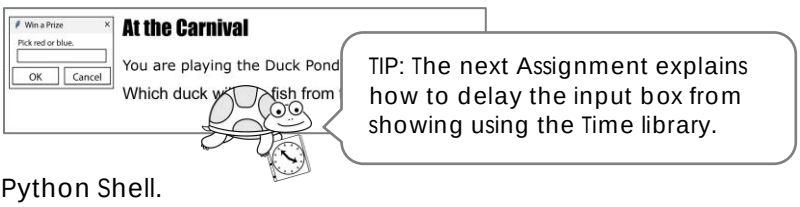
Find more color names.

Ask the Player to Make a Choice

7. Create an input box to store the player's choice. Refer to your plan for the two choices:

```
#play game
choice=textinput("Win a Prize", "Pick red or blue.")
```

▷ Run the program to look at the input box.



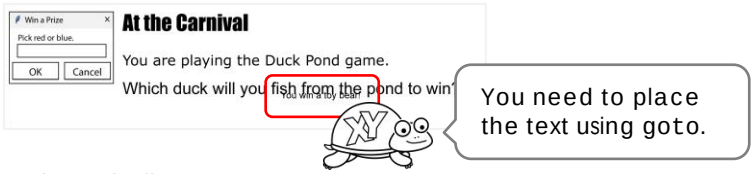
▷ Close the canvas and Python Shell.

Give the Player a Prize

8. ▷ Tell the player the prize they win based on their choice. For example:

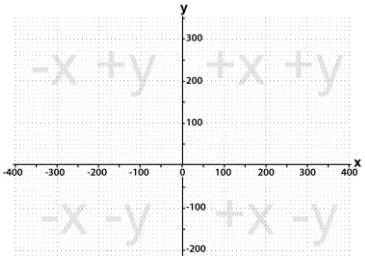
```
#prizes
if choice=="red":
     write("You win a toy bear!")
elif choice=="blue":
     write("You win a pink balloon!")
else:
     write("You do not win a prize!")
```

▷ Run the program. Type in the first choice.



▷ Close the canvas and Python Shell.

Place the Prize Text on the Canvas



9. ▷ Where should the prize text go? `goto(x, y)`

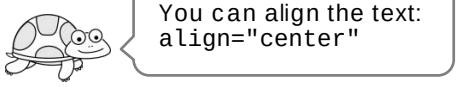
```
#prizes
goto(-300, -150)
if choice=="red":
    write("You win a toy bear!")
```

▷ Add code to format the font and font size. For example:

```
if choice=="red":
     write("You win a toy bear!", font=("Arial", 20))
elif choice=="blue":
     write("You win a pink balloon!", font=("Arial", 20))
else:
     write("You do not win a prize!", font=("Arial", 20))
```

Fonts

Arial	Impact	Verdana
Bookman	Times	Georgia



Complete the Game Design Checklist

10. Complete the checklist:

		✓
The window title bar has the Game Title.		
The canvas and text color look great.		
The text is easy to read.		
The correct prize shows for each choice.	if	
	elif	
	else	

Close the Program

This is a preview of the teacher guide.
Pages have been omitted.

Assignment 30 Step Right Up to Win a Prize

Invite a friend to play your Carnival Game.

1. Have the player control when the game closes.
   Add the code `exitonclick()` to the last line of your program.

2. Decide how to share the game with a player. The Carnival Game can be open on your device. You can also send the player the carnival.py file by email or a file sharing service.

3. Get feedback! Ask the player to answer the questions below after they are done playing.

Carnival Game Feedback (Completed by Player)

Player Name:

Programmer Name:

Carnival Game:

What did you like the most about the Carnival Game?

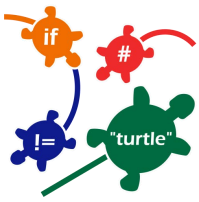
- ☐ The game description made it fun to play.
- ☐ Getting to pick a choice to win a prize.
- ☐ Being surprised by what I won.
- ☐ Other:

What did you win? Were you happy with the prize?

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

- ☐ Change the words on the canvas to make them stand out more.
- ☐ Space out the words more on the canvas.
- ☐ Change one of the prizes to
- ☐ Add another choice such as
- ☐ Other:

This is a preview of the teacher guide.
Pages have been omitted.



Appendices

Refer to the appendices for additional resources:

This is a preview of the teacher guide.
Pages have been omitted.

Carnival Game Marking Sheet

Game Window	
The window has *Game Title* in the title bar. The canvas has a colored background. The turtle symbol is hidden from view.	/3
Game Play	
The game has a title that is centered near the top of the canvas. The text is easy to read. The game description is easy to understand. An input box lets the player pick from two choices. The correct prize shows for each option. (if, elif, else)	/7
Coding	
Comments are used to organize the code.	/1
Creativity	
The prizes make playing the game fun. Extra coding has been added to enrich the game design (timing of events, flashing text, picture of prizes)	/4
Comments: TOTAL	/15