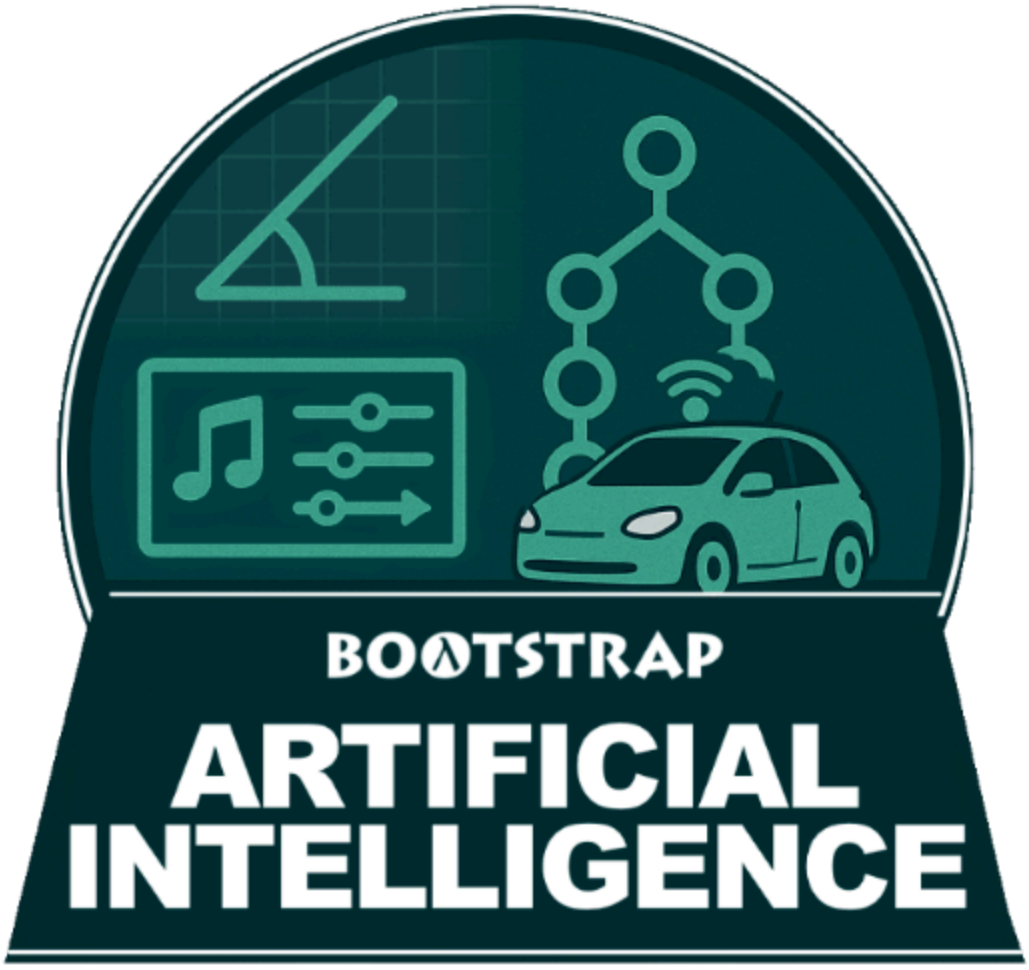
Name: _____



Student Workbook

Fall, 2025 — Pyret Edition

Workbook v3.1

Brought to you by the Bootstrap team:

- Emmanuel Schanzer
- Kathi Fisler
- Shriram Krishnamurthi
- Dorai Sitaram
- Joe Politz
- Ben Lerner
- Nancy Pfenning
- Flannery Denny
- Rachel Tabak

Measuring Similarity

Previewing the Photos

1) What about the nine photos makes them all *similar* to one another?

2) What about the nine photos makes them all *different* from one another?

Sorting the Photos

3) "Alt text," short for alternative text, is a short description added to images on a webpage to provide context for users who cannot see the image. Write a brief description of Photo 1 that could serve as alt text, describing the image for someone who cannot see it.

4) Which is *more similar* to Photo 1: Photo 2 or Photo 3? Explain how you decided which photo is *more similar*.

5) Write a brief description of Photo 4 that could serve as alt text, providing as much detail as you can.

6) Which is *more similar* to Photo 4: Photo 5 or Photo 6? Explain how you decided which photo is *more similar*.

Using the Coordinate Plane

7) We just thought about where these images would land on a coordinate grid with its axes labeled *"body of water pixels"* and *"mountain pixels"*. Think of other possible labels for the x and y axes. How many can you come up with?

Table of Contents

Table of Contents

Thinking about AI in Pop Culture

1) The media we chose: _____

2) The message about AI that we're going to discuss:

- □ AI must be safely controlled at all times.
- □ AI could lead to social isolation.
- □ AI can help us understand what it means to be human.
- □ AI can help advance humanity and scientific discovery.
- □ AI may pose a threat to humanity.
- □ AI can perpetuate biases and lead to unfair outcomes.
- □ Humans are helpless in the face of AI.

3) Our thoughts about how the media we chose supports the message we selected:

Introduction to Programming in a Nutshell

The **Editor** is a software program we use to write Code. Our Editor allows us to experiment with Code on the right-hand side, in the **Interactions Area**. For Code that we want to *keep*, we can put it on the left-hand side in the **Definitions Area**. Clicking the "Run" button causes the computer to re-read everything in the Definitions Area and erase anything that was typed into the Interactions Area.

Data Types

Programming languages involve different ***data types***, such as Numbers, Strings, Booleans, and even Images.

- Numbers are values like `1`, `0.4`, `1/3`, and `-8261.003`.
  - Numbers are *usually* used for quantitative data and other values are *usually* used as categorical data.
  - In Pyret, decimals *must* start with a zero. For example, `0.22` is valid, but `.22` is not.
- Strings are values like `"Emma"`, `"Rosanna"`, `"Jen and Ed"`, or even `"08/28/1980"`.
  - All strings *must* be surrounded by quotation marks.
- Booleans are either `true` or `false`.

All values evaluate to themselves. The program `42` will evaluate to `42`, the String `"Hello"` will evaluate to `"Hello"`, and the Boolean `false` will evaluate to `false`.

Operators

Operators (like `+`, `-`, `*`, `<`, etc.) work the same way in Pyret that they do in math.

- Operators are written between values, for example: `4 + 2`.
- In Pyret, operators must always have spaces around them. `4 + 2` is valid, but `4+2` is not.
- If an expression has different operators, parentheses must be used to show order of operations. `4 + 2 + 6` and `4 + (2 * 6)` are valid, but `4 + 2 * 6` is not.

Applying Functions

Functions work much the way they do in math. Every function has a name, takes some inputs, and produces some output. The function name is written first, followed by a list of ***arguments*** in parentheses.

- In math this could look like $f(5)$ or $g(10, 4)$.
- In Pyret, these examples would be written as `f(5)` and `g(10, 4)`.
- Applying a function to make images would look like `star(50, "solid", "red")`.
- There are many other functions in Pyret, for example `sqr`, `sqrt`, `triangle`, `square`, `string-repeat`, etc.

Functions have ***contracts***, which help explain how a function should be used. Every Contract has three parts:

- The *Name* of the function — literally, what it's called.
- The *Domain* of the function — what *type(s) of value(s)* the function consumes, and in what order.
- The *Range* of the function — what *type of value* the function produces.

Strings and Numbers

*Make sure you've loaded code.pyret.org (CPO), clicked "Run", and are working in the **Interactions Area** on the right. Hit Enter/return to evaluate expressions you test out.*

Strings

String values are always in quotes.

- Try typing your name *(in quotes!)*.
- Try typing a sentence like "I'm excited to learn to code!" *(in quotes!)*.
- Try typing your name with the opening quote, but *without the closing quote.* Read the error message!
- Now try typing your name *without any quotes.* Read the error message!

1) Explain what you understand about how strings work in this programming language. _____

_____

Numbers

2) Try typing `42` into the Interactions Area and hitting "Enter". Is `42` the same as `"42"`? Why or why not?

_____

3) What is the largest number the editor can handle?

_____

4) Try typing `0.5`. Then try typing `.5`. Then try clicking on the answer. Experiment with other decimals.

Explain what you understand about how decimals work in this programming language. _____

_____

5) What happens if you try a fraction like `1/3`? _____

_____

6) Try writing **negative** integers, fractions and decimals. What do you learn? _____

_____

_____

Operators

7) Just like math, Pyret has ***operators*** like `+`, `-`, `*` and `/`.
Try typing in `4 + 2` and then `4+2` (without the spaces). What can you conclude from this?

_____

8) Type in the following expressions, **one at a time**: `4 + 2 * 6` `(4 + 2) * 6` `4 + (2 * 6)` What do you notice?

_____

9) Try typing in `4 + "cat"`, and then `"dog" + "cat"`. What can you conclude from this?

_____

_____

Booleans

Boolean-producing expressions are yes-or-no questions, and will always evaluate to either `true` ("yes") or `false` ("no").
What will the expressions below evaluate to? *Write down your prediction, then type the code into the Interactions Area to see what it returns.*

	Prediction	Result		Prediction	Result
1) `3 <= 4`			2) `"a" > "b"`		
3) `3 == 2`			4) `"a" < "b"`		
5) `2 < 4`			6) `"a" == "b"`		
7) `5 >= 5`			8) `"a" <> "a"`		
9) `4 >= 6`			10) `"a" >= "a"`		
11) `3 <> 3`			12) `"a" <> "b"`		
13) `4 <> 3`			14) `"a" >= "b"`		

15) In your own words, describe what `<` does. _____

_____

16) In your own words, describe what `>=` does. _____

_____

17) In your own words, describe what `<>` does. _____

_____

	Prediction:	Result:
18) `string-contains("catnap", "cat")`		
19) `string-contains("cat", "catnap")`		

20) In your own words, describe what `string-contains` does. Can you generate another expression using `string-contains` that returns true?

_____

★ There are infinite string values ("a", "aa", "aaa"…) and infinite number values out there (…-2,-1,0,-1,2…). But how many different *Boolean* values are there? _____

Applying Functions

Open code.pyret.org (CPO) and click "Run". We will be working in the Interactions Area on the right.

Test out these two expressions and record what you learn below:

- `regular-polygon(40, 6, "solid", "green")`
- `regular-polygon(80, 5, "outline", "dark-green")`

1) You've seen data types like Numbers, Strings, and Booleans. What data type did the `regular-polygon` function produce? _____

2) How would you describe what a regular polygon is? _____

_____

3) The `regular-polygon` function takes in four pieces of information (called arguments). Record what you know about them below.

	Data Type	Information it Contains
Argument 1	Number	Length of each side
Argument 2	Number	
Argument 3		
Argument 4		

There are many other functions available to us in Pyret. We can describe them using ***contracts***. The Contract for `regular-polygon` is:

```
# regular-polygon :: Number, Number, String, String -> Image
```

- Each Contract begins with the function name: *in this case* `regular-polygon`
- Lists the data types required to satisfy its Domain: *in this case* `Number, Number, String, String`
- And then declares the data type of the Range it will return: *in this case* Image

Contracts can also be written with more detail, by annotating the Domain with *variable names*:

```
# regular-polygon :: ( Number , Number , String , String ) -> Image
                     size     number-of-sides  fill-style  color
```

4) We know that a square is a regular polygon because _____

_____

5) What code would you write to make a big, blue square using the `regular-polygon` function?

_____ (_____ , _____ , _____ , _____)
function-name — size :: Number — number-of-sides :: Number — fill-style :: String — color :: String

6) Pyret also has a `square` function whose contract is:

```
# square :: ( Number , String , String ) -> Image
             size     fill-style  color
```

What code would you write to make a big blue square using the `square` function?

_____ (_____ , _____ , _____)
function-name — size :: Number — fill-style :: String — color :: String

7) Why does `square` need fewer arguments to make a square than `regular-polygon`? _____

_____

★ Where else have you heard the word ***contract*** used before?

Practicing Contracts: Domain & Range

Note: The contracts on this page are not defined in Pyret and cannot be tested in the editor.

is-beach-weather

Consider the following Contract:

```
# is-beach-weather :: Number, String -> Boolean
```

1) What is the **Name** of this function? _____

2) How many arguments are in this function's **Domain**? _____

3) What is the **Type** of this function's **first argument**? _____

4) What is the **Type** of this function's **second argument**? _____

5) What is the **Range** of this function? _____

6) Circle the expression below that shows the correct application of this function, based on its Contract.

A. `is-beach-weather(70, 90)`
B. `is-beach-weather(80, 100, "cloudy")`
C. `is-beach-weather("sunny", 90)`
D. `is-beach-weather(90, "stormy weather")`

cylinder

Consider the following Contract:

```
# cylinder :: Number, Number, String -> Image
```

7) What is the **Name** of this function? _____

8) How many arguments are in this function's **Domain**? _____

9) What is the **Type** of this function's **first argument**? _____

10) What is the **Type** of this function's **second argument**? _____

11) What is the **Type** of this function's **third argument**? _____

12) What is the **Range** of this function? _____

13) Circle the expression below that shows the correct application of this function, based on its Contract.

A. `cylinder("red", 10, 60)`
B. `cylinder(30, "green")`
C. `cylinder(10, 25, "blue")`
D. `cylinder(14, "orange", 25)`

Matching Expressions and Contracts

Match the Contract (left) with the expression that uses it correctly (right).
Note: The contracts on this page are not defined in Pyret and cannot be tested in the editor.

Contract		Expression
`# make-id :: String, Number -> Image`	1	A `make-id("Savannah", "Lopez", 32)`
`# make-id :: String, Number, String -> Image`	2	B `make-id("Pilar", 17)`
`# make-id :: String -> Image`	3	C `make-id("Akemi", 39, "red")`
`# make-id :: String, String -> Image`	4	D `make-id("Raïssa", "McCracken")`
`# make-id :: String, String, Number -> Image`	5	E `make-id("von Einsiedel")`

Contract		Expression
`# is-capital :: String, String -> Boolean`	6	A `show-pop("Juneau", "AK", 31848)`
`# is-capital :: String, String, String -> Boolean`	7	B `show-pop("San Juan", 395426)`
`# show-pop :: String, Number -> Image`	8	C `is-capital("Accra", "Ghana")`
`# show-pop :: String, String, Number -> Image`	9	D `show-pop(3751351, "Oklahoma")`
`# show-pop :: Number, String -> Number`	10	E `is-capital("Albany", "NY", "USA")`

Contracts for Image-Producing Functions

Log into code.pyret.org (CPO) and click "Run". Experiment with each of the functions listed below in the interactions area. Try to find an expression that produces an image. Record the contract and example code for each function you are able to use!

Name		Domain		Range
`# triangle`	`::`	`Number, String, String`	`->`	`Image`
`triangle(80, "solid", "darkgreen")`				
`# star`	`::`		`->`	
`# circle`	`::`		`->`	
`# rectangle`	`::`		`->`	
`# text`	`::`		`->`	
`# square`	`::`		`->`	
`# rhombus`	`::`		`->`	
`# ellipse`	`::`		`->`	
`# regular-polygon`	`::`		`->`	
`# right-triangle`	`::`		`->`	
`# isosceles-triangle`	`::`		`->`	
`# radial-star`	`::`		`->`	
`# star-polygon`	`::`		`->`	
`# triangle-sas`	`::`		`->`	
`# triangle-asa`	`::`		`->`	

Catching Bugs when Making Triangles

Learning about a Function through Error Messages

1) Type `triangle` into the Interactions Area of code.pyret.org (CPO) and hit "Enter". What do you learn? _____

2) We know that all functions will need an open parenthesis and at least one input! Type `triangle(80)` in the Interactions Area and hit Enter/return. Read the error message. What hint does it give us about how to use this function?

_____

3) Using the hint from the error message, experiment until you can make a triangle. What is the contract for `triangle`?

_____

4) Read the explanation below. Then explain the difference in your own words.

syntax errors — when the computer cannot make sense of the code because of unclosed strings, missing commas or parentheses, etc.
contract errors — when the function isn't given what it needs (the wrong type or number of arguments are used)

The difference between **syntax errors** and **contract errors** is: _____

_____

Finding Mistakes with Error Messages

The following lines of code are all BUGGY! Read the code and the error messages below. See if you can find the mistake WITHOUT typing it into Pyret.

5) `triangle(20, "solid" "red")`

```
Pyret didn't understand your program around
triangle(20, "solid" "red")
```

This is a _____ (contract / syntax) error. The problem is that _____

6) `triangle(20, "solid")`

```
This application expression errored:
triangle( 20 , "solid")
2 arguments were passed to the operator. The operator evaluated to a function accepting 3
parameters. An application expression expects the number of parameters and arguments to be the
same.
```

This is a _____ (contract / syntax) error. The problem is that _____

7) `triangle(20, 10, "solid", "red")`

```
This application expression errored:
triangle(20, 10, "solid", "red")
4 arguments were passed to the operator. The operator evaluated to a function accepting 3
parameters. An application expression expects the number of parameters and arguments to be the
same.
```

This is a _____ (contract / syntax) error. The problem is that _____

8) `triangle (20, "solid", "red")`

```
Pyret thinks this code is probably a function call:
triangle (20, "solid", "red")
Function calls must not have space between the function expression and the arguments.
```

This is a _____ (contract / syntax) error. The problem is that _____

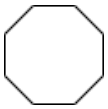
Using Contracts

For questions 1,2,4,5,8 & 9, use the contracts provided to find expressions that will generate images similar to the ones pictured.
Test your code in [code.pyret.org (CPO)] before recording it.

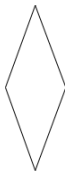
```
# ellipse :: ( Number , Number , String , String ) -> Image
             width    height   fill-style  color
```

1)		
2)		
3)	Write an expression using `ellipse` to produce a circle.	

```
# regular-polygon :: ( Number , Number , String , String ) -> Image
                      side-length  number-of-sides  fill-style  color
```

4)		
5)		
6)	Use `regular-polygon` to write an expression for a square!	
7)	How would you describe a **regular polygon** to a friend?	

```
# rhombus :: ( Number , Number , String , String ) -> Image
              size     top-angle  fill-style  color
```

8)		
9)		
10)	Write an expression to generate a `rhombus` that is a square!	

Triangle Contracts

Respond to the questions. Go to code.pyret.org (CPO) to test your code.

1) What kind of triangle does the `triangle` function produce? _____
There are lots of other kinds of triangles! And Pyret has lots of other functions that make triangles!

```
# triangle :: (Number, String, String) -> Image
#             (size, fill-style, color)
# right-triangle :: (Number, Number, String, String) -> Image
#                   (base, height, fill-style, color)
# isosceles-triangle :: (Number, Number, String, String) -> Image
#                       (leg, angle, fill-style, color)
```

2) Why do you think `triangle` only needs one number, while `right-triangle` and `isosceles-triangle` need two numbers?

_____

_____

3) Write `right-triangle` expressions for the images below using `100` as one argument for each.

_____

_____

4) Write `isosceles-triangle` expressions for the images below using `100` as one argument for each.

_____

_____

5) Write 2 expressions that would build **right-isosceles** triangles. Use `right-triangle` for one expression and `isosceles-triangle` for the other expression.

_____

_____

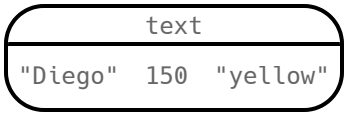
6) Which do you like better? Why? _____

Composing with Circles of Evaluation

Notice and Wonder

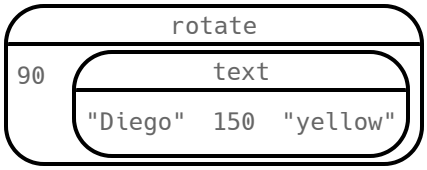
Suppose we want to see the `text` *"Diego" written vertically in yellow letters of size 150. Let's use Circles of Evaluation to look at the structure:*

We can start by generating the Diego image.



→

And then use the `rotate` function to rotate it 90 degrees.




```
text("Diego", 150, "yellow")
```

```
rotate(90, text("Diego", 150, "yellow"))
```

1) What do you Notice? _____

_____

2) What do you Wonder? _____

_____

Let's Rotate an Image of Your Name!

Suppose you wanted the computer to show your name in your favorite color and rotate it so that it's diagonal…

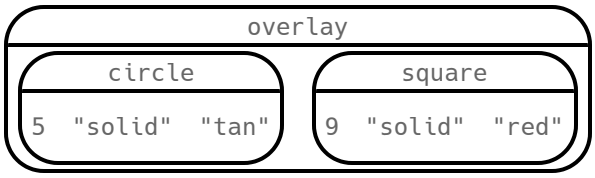
Write your name (any size), in your favorite color	**`rotate` the image so that it's diagonal**
3) Draw the circle of evaluation:	4) Draw the circle of evaluation:
5) Convert the Circle of Evaluation to code:	6) Convert the Circle of Evaluation to code:

Circle of Evaluation to Code (Scaffolded)

Complete the Code by Filling in the Blanks!

Finish the Code by filling in the blanks.

1)

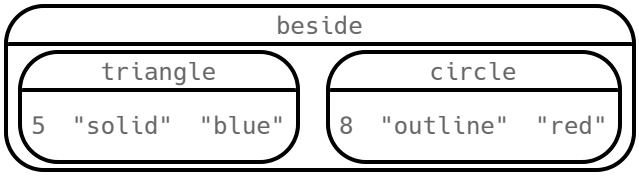



```
overlay(circle(____, "solid", _____________________), _____________________(9, _____________________, "red"))
```

Complete the Code by adding Parentheses

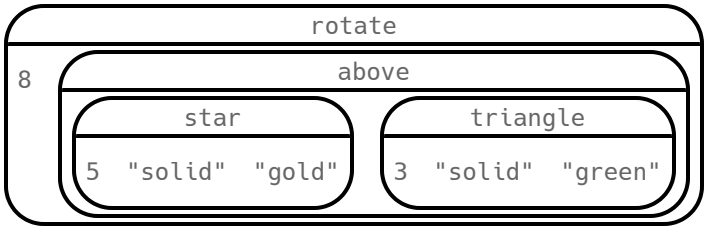
For each Circle of Evaluation, finish the Code by adding parentheses and commas.

2)



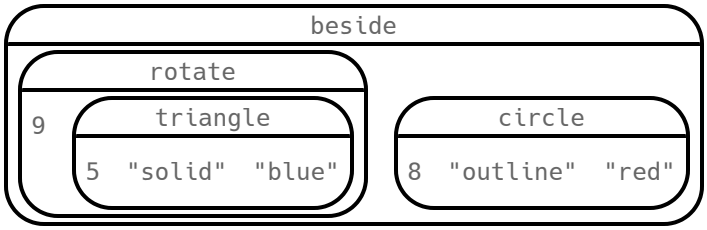

```
beside   triangle   5   "solid"   "blue"   circle   8   "outline" "red"
```

3)




```
rotate   8   above   star   5   "solid"   "gold"   triangle   3   "solid"   "green"
```

4)




```
beside   rotate   9   triangle   5   "solid"   "blue"   circle   8   "outline"   "red"
```

Human Spell Checking

"Before beginning the lesson, the teacher said wello to her students and asked them how thay were doing."

First Misspelled Word

1) What is the **first word that is spelled incorrectly** in the quote at the top? _____ What was the *intended* word? _____

How did you know that that word was misspelled? Explain in as much detail as you can. _____

_____

_____

_____

2) Do you think a 6-year old would be able to identify and correct the first misspelled word in this sentence? Explain. _____

_____

_____

_____

Second misspelled word

3) What is the **second word that is spelled incorrectly** in the quote at the top? _____ What was the *intended* word? _____

How did you know that that word was misspelled? Explain in as much detail as you can. _____

_____

_____

_____

4) Do you think a 6-year old would be able to identify and correct the second misspelled word in this sentence? Explain. _____

_____

_____

_____

Reflect

5) You probably didn't need a dictionary to correct the two errors above. When might consulting a dictionary be a part of your spell-checking process? Explain. _____

_____

_____

The First Spell Checker

The very first spell checker would use misspelled words as inputs.

- **First**, it would develop alternative candidates for the misspelled word (input) by making one of the following adjustments: (1) replace a single letter with a different one, or (2) swap the positions of two adjacent letters.
- **Next**, it would search the provided dictionary to see which alternatives were valid options.
- **Finally**, it would produce a list of valid correctly spelled words for the user to choose from.

Part 1: Follow the Algorithm

Misspelled Word: wello

1) Generate alternative candidates for the misspelled word. *(These candidates don't need to be real words!)*

- First, replace **one** letter from "wello" with a different one. *We've provided a first option to get you started.*

  dello _____ _____ _____ _____

- Next, swap the positions of two **adjacent** letters. *Again, we've provided a first option to get you started.*

  ewllo _____ _____ _____ _____

2) Circle the *actual* words you listed above. How many real-word suggestions did you come up with? _____

3) You already found 5 "words" by replacing a single letter of **wello**. Using that strategy, about how many more could you list?

about 10 more about 50 more about 100 more about 1000 more

Part 2: Reflect

4) What do you think are some limitations of this algorithm? _____

5) Compared to your own strategy for spell-checking, how similar / different is the first spell checker's algorithm?

A Pyret Spell Checker: The Algorithm

Open the Spell Checker Starter File and click "Run". Follow each of the instructions below to discover how our very own Pyret spell checker works.

1) In the Spell Checker Starter File, the function `alt-words` consumes a string and a dictionary… but what does it produce?

- Test `alt-words("straw", WORDS)` in the Interactions Area and record the results in the left column of the table below.
- In the second column, describe how each option differs from the input string ("straw").
- In the third column, record how many edits to the string "straw" this algorithm makes. *(A single edit could entail replacing one letter with a different one, for instance.)*

Suggestion	How it's different from the string "straw"	Number of Edits

2) Test `alt-words("planet", WORDS)` in the Interactions Area.

- Notice that "planet" has six letters, but all of the suggested words have five letters. What else can you conclude about the spell checker's algorithm? _____

_____

- To get from "planet" to "plait", delete _____ and change _____ to _____.
- To get from "planet" to "plank", delete _____ and change _____ to _____.
- The spell checker makes just *one* edit when it proposes _____ as a match for "planet".

Experiment with the `alt-words` function to answer the questions below, testing what it produces using Strings from your imagination.

3) We have seen multiple two-edit changes. Does the program propose words that include three edits? _____

4) What is the maximum length string that this spell checker will accept? _____

- Why does that maximum length make sense with what we know about the algorithm? _____

_____

5) What else do you notice? _____

_____

6) The Pyret spell check program does not use the same algorithm as the The First Spell Checker. How are the programs different?

_____

_____

_____

7) **Edit distance** is a word programmers use to describe the number of operations needed to transform one string into another string. How can **edit distance** help us measure similarity? _____

_____

A Pyret Spell Checker: Exploring Different Dataset Sizes

The player of the partially-completed Wordle game wants some Pyret "assistance". *Open the Spell Checker Starter File and click "Run".*

1) In the Interactions Area, test each of the three lines of code (below).

Pyret Expression	Number of suggestions returned
`alt-words("games", WORDS-100)`	
`alt-words("games", WORDS-1000)`	
`alt-words("games", WORDS-ALL)`	

2) Which line of code would you recommend for the player? Why? _____

_____

3) The solution to the game above is "camel". Look back at the suggestions in your Interactions Area. Did any of the lists include it?

_____

4) We used the very same function (`alt-words`) for each entry. Why didn't we receive the same results from all 3 expressions? _____

_____

_____

5) To see the words in the `WORDS-100` dictionary, type `WORDS-100` into the Interactions Area and then click on `tree-set`. How many words are in this list? _____

6) What do you notice about the words in the list? _____

_____

7) What do you wonder? _____

_____

8) As you might have guessed, the `WORDS-1000` dictionary contains 1000 words and the `WORDS-ALL` dictionary contains even more (2314). How does that help explain the fact that `alt-words("games", WORDS-ALL)` returned the most suggestions? _____

_____

★ Let's imagine that

- `this-word` and `that-word` represent two different five-letter words
- `alt-words(this-word, WORDS-ALL)` produces a list that includes `that-word`

Would you expect `alt-words(that-word, WORDS-ALL)` to produce a list that includes `this-word`? Why or why not?

_____

_____

★ Can you produce an input that returns more than one word from the `WORDS-100` dictionary? If so, what is it? _____

_____

Supervised Machine Learning: Training a Self-Driving Car

Thinking about Training

1) *Refer to the weather forecast (right) in your response.* During a week of daytime test drives:

- On Wednesday and Friday, the self-driving car drives safely.
- On the other days, the self-driving car is unsafe.

What might explain why the car drove unsafely on some days?

2) Imagine that a self-driving car has done extensive training on a one-lane road, in all weather conditions, and at all times of day. Would you expect it to be able to safely drive on a busy two-lane road? Explain.

3) Imagine that a self-driving car trained on isolated country roads, as well as city streets, and highways in all weather conditions and times of day. It would be pretty safe on many roads! What might it still not be prepared to navigate safely?

Confidence Rating

4) In addition to producing a steering angle for each image of the road, ALVINN produces a *numeric* measure of "confidence" in its response.

- What do you think causes ALVINN's "confidence" to increase or decrease?

- Is 100% "confidence" possible?

Case Study: Michelle's Spotify Use

Michelle started using Spotify six months ago. The first time she checked out her AI-produced "Discover Weekly" playlist, she was underwhelmed. The songs did not match her tastes, and she ended up skipping over many of them within a few seconds. Michelle was so disappointed in that first experience that she gave up entirely on the "Discover Weekly" playlist, writing it off as a feature she did not care for.

After three months of using Spotify and listening to all of her favorite songs, artists, and playlists, Michelle happened to click on "Discover Weekly" again — but this time, she was delighted by what she heard. It felt as though her best friend had curated a playlist just for her. Michelle discovered new music that she had never heard before, that she absolutely loved.

She remarked, **"Wow, Spotify has really improved their algorithms since I last checked out Discover Weekly."**

1) Circle your assessment of Michelle's statement: *definitely true* *possibly true* *definitely not true*

2) What do you think might explain why her playlist is much better after three months of using Spotify? Provide as much detail as you can.
Hint: There may be more than one possible explanation!

3) Michelle loves folk music featuring female vocals, but she has decided that she wants to branch out and is starting to feel frustrated by how Spotify keeps recommending folk music to her over and over and over! Why isn't Spotify giving Michelle the music suggestions she's looking for?

4) What do you think Michelle should do to start getting different recommendations?

Designing a Song Recommendation System

1) Think of a song that you know very well. _____
Describe it in as much detail as you can.

- Begin with facts like the topic, musical genre, artist, year of release, language the lyrics are sung in, tempo, instruments used, etc.
- Then describe it more personally… for example, how it makes you feel, when you choose to listen / avoid listening to it, etc.
- Have fun!

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

2) A song recommendation system must collect a user's listening history and extract data that is useful for making predictions about the user's music preferences.

- If you were designing a song recommendation system, what specific data would it store? Make a list of ideas.
- Then rank each item based on how heavily you think it should be weighted in your song prediction function.

Music Characteristic (e.g. tempo, pitch, volume, artist)	Ranking

A Primitive Plagiarism Detector

Review the Documents

Wikipedia article on "Elephants in Thailand"	Student-submitted essay on "Elephants in Thailand"
The elephant has been a contributor to Thai society and its icon for many centuries. The elephant has had a considerable impact on Thai culture. The Thai elephant is the official national animal of Thailand. The elephant found in Thailand is the Indian elephant, a subspecies of the Asian elephant.	The elephant is a contributor to Thai society. It has been an icon of Thai life for many centuries. The elephant, which it is possible to see found in every part of Thailand, is the Indian elephant, which is a subspecies of the Asian elephant. The Thai elephant has a considerable impact on culture. The elephant is the official national animal of Thailand.

1) Do **you** think the student who submitted this essay on Elephants in Thailand committed plagiarism? Circle one: Yes No

Detect Identicality

Open the Plagiarism Detection Starter File and click "Run".

- The two paragraphs above are defined in the starter file as `wiki-article` and `student-essay`, respectively.
- The file also defines a function `simple-equality`, which takes in two Strings and produces a Boolean: `true` indicates that there is a match; `false` indicates that there is not a match.

2) Record if the output is `true` or `false` for each line of code:

```
simple-equality("hello", "goodbye")  _____
simple-equality("hello", "HELLO")  _____
simple-equality("hello", "helo")  _____
simple-equality("hello,", " Hello.")  _____
```

3) How does `simple-equality` deal with punctuation, capitalization, and spaces? _____

4) Compare the essays using `simple-equality(student-essay, wiki-article)`. What does the program return? _____

Building a Better Plagiarism Detector

Yara and Xola agree that there has to be a way to measure the *similarity* of the essays. They each propose a different solution.

> Yara: I made a list of the four most unusual and unique words in each essay. Then I compared the lists. Because the two lists of unusual words are identical (elephant, Thai, Indian, and Asian), I decided that the student plagiarized!
>
> Xola: I figured out the four most common words in each essay. In the wiki-article, they are: "the" (7), "elephant" (6), "Thai" (3), and "a" (3). In the student essay, they are: "the" (7), "elephant" (6), "is" (5), and "of" (4). Because the most common words are not the same, I don't think the student plagiarized.

5) What do you like about Yara's method? What are its shortcomings?

6) What do you like about Xola's method? What are its shortcomings?

7) Come up with your own method! What is a **different** way to measure the similarity between the two essays?

A Slightly Less Primitive Plagiarism Detector

Open Plagiarism Detection Starter File. Click "Run".

Reviewing `simple-similarity`

1) Type `doc-a` into the Interactions Area, then hit "Enter". What appears? _____

2) Type `doc-b` into the Interactions Area, then hit "Enter". What appears? _____

3) If we compare these two documents using our `simple-equality` function, what do you predict the outcome will be… and why?

_____

_____

4) Type `simple-equality(doc-a, doc-b)`. What is the output? _____

Testing `string-to-bag` and `bag-equality`

5) This starter file uses the function `string-to-bag`, which converts a string into a bag of words. Type `string-to-bag(doc-a)` in the Interactions Area and hit "Enter". What is the output? _____

_____

6) Type `string-to-bag(doc-b)` in the Interactions Area and hit "Enter". What do you observe? _____

_____

7) The function `bag-equality` checks if two different bags of words are identical. What do you predict the output will be when you use `doc-a` and `doc-b` as the arguments for `bag-equality`? _____ Were you correct? _____

Assessing the model on our elephants texts

8) Use `string-to-bag` to build one bag of words for the `wiki-article`, and another for the `student-essay`. Do the bags of words appear to be identical to you, or not? _____

_____

9) Our `simple-equality` function returned `false` when we compared `wiki-article` with `student-essay`, because the two documents are not identical. What do you predict that `bag-equality` will produce, when we compare `wiki-article` with `student-essay`? Why? _____

_____

10) Use `bag-equality` to compare `wiki-article` and `student-essay`. Was your prediction correct? _____

Reflect

11) How is the `bag-equality` plagiarism detector different from our primitive `simple-equality` plagiarism detector? How is it similar?

_____

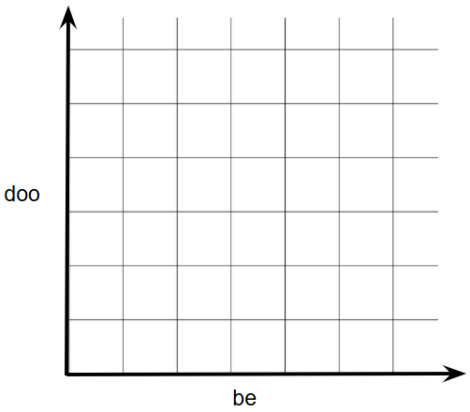
_____

Plotting Bags of Words

Complete the table below by filling in any blank cells. We've filled in the first row for you. Some cells have more than one possible solution.

Document	Text	Bag-of-Words Summary	Ordered Pair
A	be be be be be be	`be: 6, doo: 0`	(6,0)
B	doo doo doo doo doo doo		
C	doo doo doo be be be		
D	doo be doo be doo be		
E	doo doo be be be be		
F	be be doo doo doo doo		
G	doo be doo doo doo doo		
H			(5,1)
I		`be: 1, doo: 2`	
J	doo doo be be		
K		`be: 6, doo: 6`	
L			(4,6)

1) Plot and label the points with the appropriate letter on the coordinate plane (below), then record what you Notice and Wonder.



What do you Notice?

What do you Wonder?

2) Jane loves song A and wants to listen to a song that is similar.

- Sierra suggests song H, claiming that it is similar to song A because five of its six words are the same.
- Jaden recommends song B. He says those two songs are similar because they are the only songs that have just one word!

Who do you agree with?

Angle Difference (with Protractor)

Fill in the blank table cells, then use them to determine the ordered pairs. Plot and label point A and point B on the coordinate plane. Draw a ray from the origin to each of the points. Measure and record the size of the angle (in degrees) formed by the rays.

1 `StringA: doo doo doo doo`

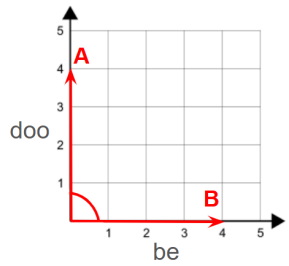
Word	Frequency
be	0
doo	4

Ordered pair: (0,4)

`StringB: be be be be`

Word	Frequency
be	4
doo	0

Ordered pair: (4,0)



The angle formed is _____.

2 `StringA: doo be`

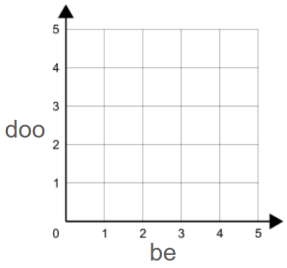
Word	Frequency
be	
doo	

Ordered pair: _____

`StringB: doo doo doo doo`

Word	Frequency
be	
doo	

Ordered pair: _____



The angle formed is _____.

3 `StringA: doo be`

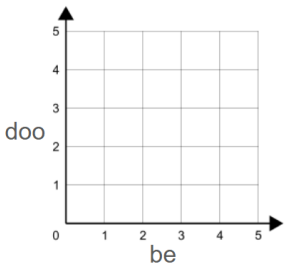
Word	Frequency
be	
doo	

Ordered pair: _____

`StringB: be doo doo doo`

Word	Frequency
be	
doo	

Ordered pair: _____



The angle formed is _____.

4 `StringA: doo doo be doo be`

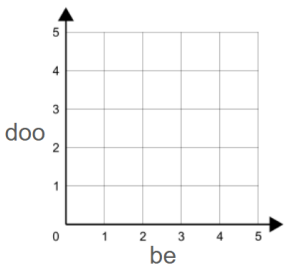
Word	Frequency
be	
doo	

Ordered pair: _____

`StringB: doo be doo be doo`

Word	Frequency
be	
doo	

Ordered pair: _____



The angle formed is _____.

The Importance of Human Judgment

How Many Dimensions?

A teacher asks students to write an essay on one of three topics: modern art, soccer, or zebras. The teacher feeds the essays to a plagiarism detector **to compare the essays to one another.** It plots 32 points in a 1023-dimensional space.

1) What does the number 32 tell us? _____

2) What does the number 1023 tell us? _____

3) The teacher knows, of course, that students don't merely copy one another's writing; instead, they copy/paste text from the internet! The teacher opens her plagiarism detection software and tells it to consume a large ***training corpus*** of internet text about modern art, soccer, and zebras. How many dimensions will the space contain, now?

Circle one: more dimensions the same number of dimensions fewer dimensions

4) Explain your choice.

_____

_____

_____

_____

Assessing the Output

On the left is a description of the output of a plagiarism detector. On the right is an interpretation of what each output means. Match the description of each output with the correct interpretation.

The 32 points are evenly scatter all over the 1023-dimensional space, with no apparent clustering.	**1**	**A**	Essays about one topic are likely to use similar words, but essays about different topics likely use different words. The three clusters probably represent the three different topics (modern art, zebras, and soccer).
The points are clustered into three groups, but two of the points are almost on top of one another.	**2**	**B**	Two of the essays use almost the same words, in almost the same frequencies. This look suspicious, and the teacher should take a closer look at those essays for potential plagiarism.
The points are clustered into three groups.	**3**	**C**	Only one student opted to write about a particular topic, while the rest of the class wrote essays on the other two. There is no evidence of plagiarism.
Almost all the points are clustered into two groups, but one point is positioned very far away from both clusters.	**4**	**D**	Every student appears to have written essays about entirely different topics! The teacher has no reason to suspect plagiarism, but they should *probably* worry about their students not following the assignment!

Exploring the Model

Use Plagiarism Detection Starter File to respond to the questions, below.

distance-to

1) In the Interactions Area, type `distance-to( student-essay )`. The table that appears displays the angle difference between `student-essay` and each of the other animal articles in the corpus. Use the information to fill in the blanks, below.

	article	angle difference
most similar to `student-essay`		
most different from `student-essay`		

2) In the Interactions Area, type `distance-to( chimpanzee-article )`. One of the angle differences listed in the table that appears is zero. What does that tell you? _____

_____

3) Use the output of `distance-to( chimpanzee-article )` to fill in the blanks, below.

	article	angle difference
most similar to `chimpanzee-article`		
most different from `chimpanzee-article`		

4) Is `distance-to` sophisticated enough to be able to determine with certainty whether or not plagiarism occurred? Explain.

_____

_____

5) Can you think of any ways to improve this model? _____

_____

_____

string-to-bag-cleaned

6) In the Interactions Area, type `string-to-bag( snail-article )`.

a) How many unique words are used in the snail article? _____

b) List each word that is used 2 or more times in the article. We've provided the first three: a (2), and (3), be (3), _____

_____

c) What do you Notice about the commonly used words? What do you Wonder? _____

_____

_____

7) In the Interactions Area, type `string-to-bag-cleaned( snail-article )`.

a) Notice that there are fewer unique words listed. How many unique words are on the table now? _____

b) List each word that appears 2 or more times, now: _____

distance-to-cleaned

Use Plagiarism Detection Starter File to respond to the questions, below.

Part 1: chimpanzee-article

Let's compare the output of `distance-to(chimpanzee-article)` with the output of `distance-to-cleaned(chimpanzee-article)`.

1) On both output tables, the angle produced in the "chimpanzee" row is zero. Why? _____

_____

2) When we used `distance-to-cleaned`, the angles were generally larger. Why? _____

_____

3) When we used `distance-to-cleaned`, four articles produced an angle measure of 90°. What does this tell us? _____

_____

_____

Part 2: mystery-article

Follow the steps below to make some discoveries about the `mystery-article`.

4) Type `distance-to(mystery-article)` in the Interactions Area. Look at the different angle measurements produced. Do you notice any outliers? _____ Explain. _____

_____

5) Think back to when you compared `student-essay` with `elephant-article`. Those two paragraphs were virtually identical, which resulted in an angle difference of about 23.706°. The difference between these two rays (`giraffe-article` and `mystery-article`) is much greater, but still seems unusual. Can you make any hypotheses about `mystery-article`? _____

_____

6) Let's investigate further. Type `distance-to-cleaned(mystery-essay)`. What do you notice? _____

_____

7) Time for the reveal! Type `mystery-article` into the Interactions Area. Why was the angle difference between this article and `giraffe-article` small? _____

_____

Part 3: Revisiting student-essay

8) Type `distance-to-cleaned(student-essay)` in the Interactions Area. What do you Notice? What do you Wonder?

_____

_____

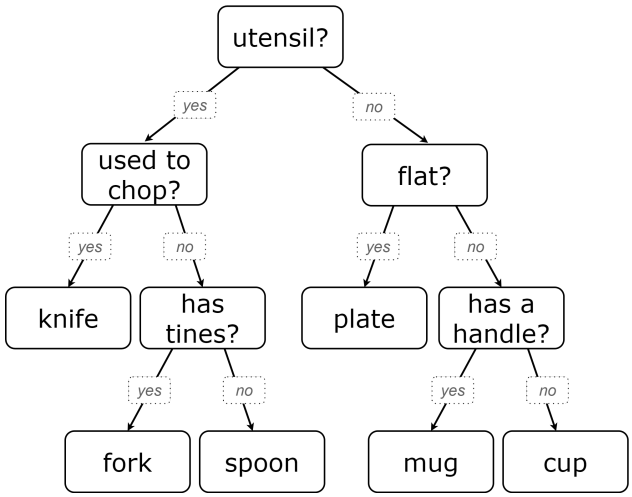
_____

Decision Tree: Spoon, Fork, Knife, Plate, Mug, Cup

Connecting Table and Decision Trees

Item	flat?	has-handle?	has-tines?	utensil?	used-to-chop?	category
A	no	yes	no	no	no	cup
B	no	yes	yes	yes	no	fork
C	yes	yes	no	yes	yes	knife
D	no	no	no	no	no	mug
E	yes	yes	no	no	no	plate
F	no	no	no	yes	no	spoon
G	yes	yes	no	yes	yes	knife

The decision tree below is one possible model for working with the data contained in the table above.



1) What do you Notice? What do you Wonder?

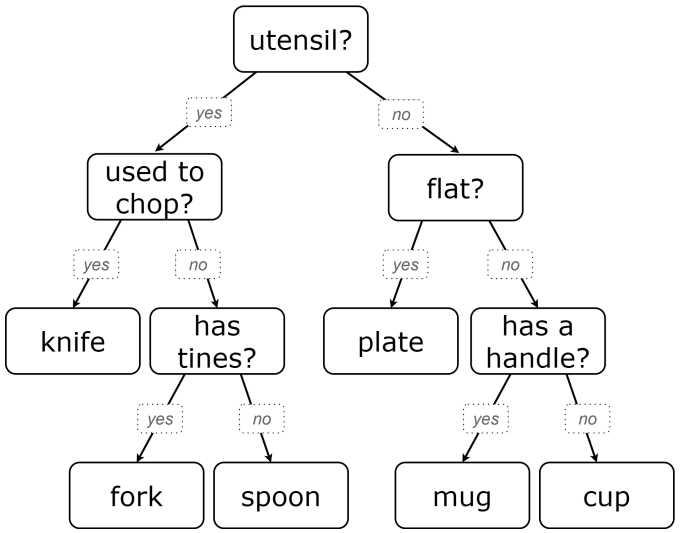
Another possible decision tree from this data

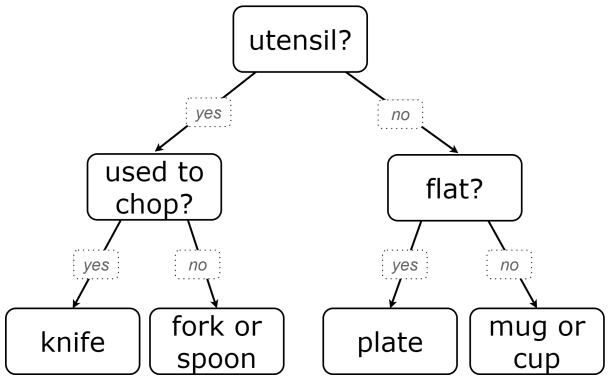
2) Make a second decision tree for the same 6 items listed above using "flat?"" as the root node with "used-to-chop" as the first decision node for **yes** and "utensil?"" as the first decision node for **no**

- For each question, add the missing "yes" or "no" arrow.
- Determine whether your arrow will lead to
  - a leaf node (a single item from the list)
  - a decision node (a new question)
- Keep adding to the decision tree until you've created pathways that lead to all 6 categories: Spoon, Fork, Knife, Plate, Mug, Cup

Testing and Comparing Decision Trees

Determine how each of these trees would categorize the items below. Explain your responses.





1) butter knife _____

2) chopstick _____

3) spork _____

4) butter knife _____

5) chopstick _____

6) spork _____

7) What do you Notice about the two decision trees above?

8) What do you Wonder about the two decision trees?

9) Which decision tree (above) would make predictions more efficiently? Explain.
*Hint: A program is **efficient** when it performs a task with minimal time, memory, energy or data.*

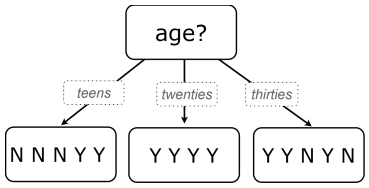
10) Which decision tree (above) would make predictions more accurately? Explain.

Decision Tree: Predicting Shopping Behavior — Part 1

name	age	shopping history	interest in game	buys game
Jan	teens	previous customer	no	no
Jose	teens	previous customer	no	no
Maribel	twenties	previous customer	no	yes
Noah	thirties	previous customer	no	yes
Sydney	thirties	previous customer	yes	yes
Mariana	thirties	new customer	yes	no
Rasula	twenties	new customer	yes	yes
Jillian	teens	previous customer	no	no
Ariella	teens	new customer	yes	yes
Isabela	thirties	previous customer	yes	yes
Danial	teens	previous customer	yes	yes
Kate	twenties	previous customer	no	yes
Taikhoom	twenties	previous customer	yes	yes
Peter	thirties	new customers	no	no

"Age" as the Root Node

The ***decision stump*** below splits the above training data by age and indicates whether the individuals in each group buy the game.



1) Where do the Y/N lists beneath each of the three branches come from? _____

_____

2) What prediction will our current model (decision stump) make for each group?

- people in their teens _____ buy the game
  will / will not
- people in their twenties _____ buy the game
  will / will not
- people in their thirties _____ buy the game
  will / will not

3) Place checkmarks below each of the values in the stump leaves for which our prediction is correct.

4) Find the likelihood of a correct prediction for each age group. teens: _____% twenties: _____% thirties: _____%

5) How accurate is the current prediction across our entire dataset? _____ *correct predictions out of 14 attempts.* (_____ % accuracy).

Improving Our Prediction

We made our prediction without considering all of the columns in our training data. If we add another level to our tree, we might be able to improve our accuracy!

6) Before moving on to the second level of his decision tree, Ernie removed all of the rows for people in their twenties. Bert said, *"I don't think that's a good idea! Why would we alter our dataset just because we're starting the second level of the tree?"* Explain Ernie's (correct) decision to Bert.

_____

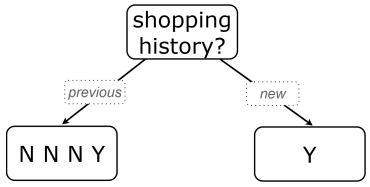
_____

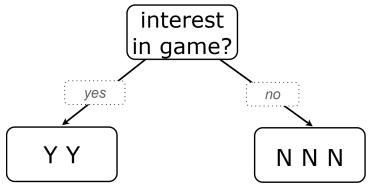
7) We used "age" as our root node. What questions could we ask at our second-level decision node? _____ [column] or _____ [column]

Decision Tree: Predicting Shopping Behavior — Part 2

For Teens — Is Shopping History or Interest in Game a better Indicator?

1) Use the decision stumps below to make rules that predict whether or not teens would buy the game based on each category.





Predict that

- previous teenage customers _____ (will buy / will not buy / no prediction)
- new teenage customers _____ (will buy / will not buy / no prediction)

Predict that

- interested teenage shoppers _____ (will buy / will not buy / no prediction)
- uninterested teenage shoppers _____ (will buy / will not buy / no prediction)

Place a checkmark below each value that the computer would predict correctly.

2) Use the decision stumps to determine how well each rule predicts whether or not teens would buy the game.

For our training data, this rule

- would make _____ correct predictions out of 5 attempts
- is _____ % accurate

For our training data, this rule

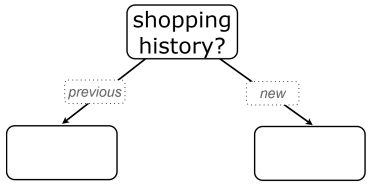
- would make _____ correct predictions out of 5 attempts
- is _____ % accurate

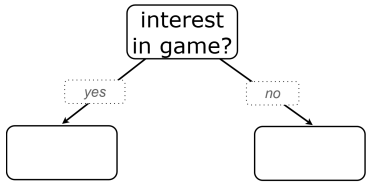
3) Will you use shopping history or interest for your second decision node for teens? _____

For People in their Thirties — Is Shopping History or Interest in Game a better Indicator?

Refer to the training data on Decision Tree: Predicting Shopping Behavior — Part 1 to complete the stumps below.

4) Complete the decision stumps below to make rules that predict whether or not people in their thirties would buy the game.





Predict that

- new customers in their 30s _____ (will buy / will not buy / no prediction)
- previous customers in their 30s _____ (will buy / will not buy / no prediction)

Predict that

- interested customers in their 30s _____ (will buy / will not buy / no prediction)
- uninterested customers in their 30s _____ (will buy / will not buy / no prediction)

5) Add checkmarks to the correct predictions. Then determine how well each rule predicts which customers in their thirties buy the game.

For our training data, this rule

- would make _____ correct predictions out of 5 attempts
- is _____ % accurate

For our training data, this rule

- would make _____ correct predictions out of 5 attempts
- is _____ % accurate

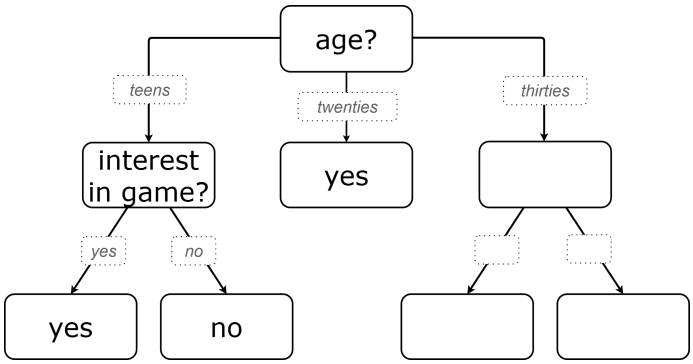
6) Will you use shopping history or interest for your second decision node for customer in their thirties? _____

Building and Testing a Decision Tree

For this page you will need to refer to your decisions from Decision Tree: Predicting Shopping Behavior — Part 2.

Build and Understand the Tree

Complete the tree (left), then fill in the blanks (right).



1) The root node of this tree is _____.

2) The first set of branches includes:

_____, _____, and _____.

3) Write the rules that this decision tree follows. Predict that:

_____ (interested / uninterested) teens will buy the game.

_____ (who?) in their 20s will buy the game.

_____ (who?) in their 30s will buy the game.

Test the Tree

4) Below is a new set of potential customers. Use the decision tree and rules above to predict whether or not they will buy the game.

name	age	shopping history	interest in game	buys game	model predicts
Kat	teen	new customer	yes	yes	
Billy	twenties	new customer	no	no	
Chen	twenties	previous customer	no	no	

5) Compare the "buys game" column with the predictions. For which customers was the computer correct? Kat Billy Chen

6) Should we change any of our rules based on the addition of this new data? Why? _____

Reflect

7) Our rules made 100% accurate predictions with our training dataset, but were only 33% accurate with our test dataset. Why?

8) What could we do to improve the quality of this decision tree? _____

Constructing a Language Model

Use There Was an Old Lady Who Swallowed a Fly to answer the questions, below. **Note that the title of the song is a part of the corpus!**

Did she swallow a fly?

Let's say we want to know the likelihood that the old lady "swallowed a fly". We need a ratio!

1) $p(fly \mid swallowed\ a) = \frac{count(swallowed\ a\ fly)}{count(swallowed\ a\ldots)} = \frac{\quad}{10}$

2) The likelihood that the old lady "swallowed a fly" in the excerpt is _____.

What about the other animals?

Find the ratio representing the likelihood that she swallowed each of the other animals.

3) $p(spider \mid swallowed\ a) = \frac{count(swallowed\ a\ spider)}{count(swallowed\ a\ldots)} = \frac{\quad}{10}$

4) $p(bird \mid swallowed\ a) = \frac{count(swallowed\ a\ bird)}{count(swallowed\ a\ldots)} = \frac{\quad}{10}$

5) $p(cat \mid swallowed\ a) = \frac{count(swallowed\ a\ cat)}{count(swallowed\ a\ldots)} = \frac{\quad}{10}$

6) Across the entire poem, the lady *most likely* swallowed a _____.

7) The second most likely option is that she swallowed a _____.

8) There is an equal probability that she swallowed a _____ or a _____.

To catch the…?

Find the ratio representing the likelihood that she caught each of the animals.

9) $p(fly \mid to\ catch\ the) = \frac{count(to\ catch\ the\ fly)}{count(to\ catch\ the\ldots)} = \frac{\quad}{6}$

10) $p(spider \mid to\ catch\ the) = \frac{count(to\ catch\ the\ spider)}{count(to\ catch\ the\ldots)} = \frac{\quad}{6}$

11) $p(bird \mid to\ catch\ the) = \frac{count(to\ catch\ the\ bird)}{count(to\ catch\ the\ldots)} = \frac{\quad}{6}$

The lady was most likely trying to catch the _____. It is the least likely that she was trying to catch the _____.

The…?

12) $p(fly \mid the) =$ ——

13) $p(spider \mid the) =$ ——

14) $p(bird \mid the) =$ ——

15) $p(cat \mid the) =$ ——

Sampling from the Language Model

Use There Was an Old Lady Who Swallowed a Fly to answer the questions, below. ***Note that the title of the song is a part of the corpus!***

Text Generation Exercise #1

1) Let's start our text generation with the phrase "She ".

Text Generation 1: She (first word) _____ (second word) _____ (third word) _____ (fourth word)

2) Determine which word is most likely to follow "She " and record it on the line for second word above.

3) Determine which word is most likely to follow the word you just wrote down and record it on the line for third word above.

4) Use statistical language modeling to determine the fourth word.

5) Everyone in your class should have generated the **same** text. Why do you think that was the outcome?

_____

_____

_____

Text Generation Exercise #2

Here is a list of the most common unigrams in the corpus: "the": 12 times "she": 11 times "a": 11 times

6) Let's start by choosing the **most** common word: _____

7) Determine which word is most likely to follow that word: _____

8) There are two words that have an equal probability of appearing in the third spot! What are they? _____ and _____

9) Flip a coin to determine which one you will use to complete **Text Generation 2a** and use statistical language modeling to determine the fourth word. Then repeat the process for **Text Generation 2b** using the other word that tied for the third spot.

Text Generation 2a: the (first word) _____ (second word) _____ (third word) _____ (fourth word)

Text Generation 2b: the (first word) _____ (second word) _____ (third word) _____ (fourth word)

10) Why was there only one result for **Text Generation Exercise 1**, while **Text Generation Exercise 2** had two possible results?

_____

_____

★ What is another starting word that would result in only one possible four-word phrase? _____

★★ What is another starting word that would result in at least two possible outcomes? _____

Meet Soekia!

Respond to the prompts below by interacting with https://bootstrapworld.org/Soekia/.

Generate text

The blue panel occupying most of your screen is where text generation takes place. This is the level typically visible to the user when using a chatbot.

1) To tell Soekia to start writing, click on the ➢ on the right. Read the text until Soekia finishes writing. Take a minute to scroll and click on the other icons in this panel: ↻ ⧉ 🤖 ☺ •○. What do you Notice? What do you Wonder?

2) Hover your mouse over one or two of the highlighted words in the fairy tale. What appears? (If the words in your fairy tale aren't already highlighted with different colors, click the 👁.)

Documents

The green *Documents* panel is where the training corpus lives. You can access it by clicking the "LOOK INSIDE → " button in the upper right-hand corner of Soekia and scrolling all the way to the right.

3) How many documents are there in this particular corpus? _____

4) Give the title of one of the documents: _____

5) Click on the 🔥 (the middle icon in the upper right). Hover over each of the 7 icons that drop down. Which one are you most interested in exploring? _____

6) Take a minute to scroll and click on any remaining icons in this panel. What else do you Notice about the *Documents* panel? What do you Wonder?

Meet Soekia! (n-grams)

Respond to the prompts below by interacting with https://bootstrapworld.org/Soekia/ using the ***Intelligent Monkeys?*** *collection.*

Exploring the N-Grams Panel

The orange *N-grams* panel is where Soekia lists possible ***n-grams*** and how frequently they occur in the training corpus. The default setting (**3**) displays a list of every ***trigram***. Clicking on the other numbers at the top will display lists of ***n-grams*** of other lengths.

1) Hover your mouse over the **3**. How many different ***trigrams*** are there in this collection? _____ How does that compare to the number of ***n-grams*** of other lengths? _____

2) The most common ***trigram*** appears at the top of the list. Click on it. What do you learn? _____

_____

3) Click on the **5** tab. Notice that all of the 5-word ***n-grams*** occur equally often. Can you explain why this might be? _____

_____

4) Take a minute to explore the *N-grams* Panel. What do you Notice? What do you Wonder?

_____

_____

_____

Predicting the Next Word using N-Grams

For this section, make sure you are in the ***Intelligent Monkeys?*** *collection. Go to the Suggested words panel, click on "Customize temperature/ number of suggestions", and set the temperature to* ***low****.*

Without introducing any randomization into the algorithm, Soekia generates text by selecting words one at a time from the most-frequent valid ***n-gram*** of the highest order available.

5) For the first "word", Soekia looks in the **1** tab to find the most frequently occurring ***unigram***. What do you expect it to choose? _____

6) To choose the second "word", Soekia:

- Looks at the **2** tab to find the most frequently occurring ***bigram*** that begins with the first "word".
- If there isn't one, it will return to the **1** tab and select the next most popular ***unigram***.

What do you expect Soekia to choose? _____ Which list did you select it from? _____

7) Why do you think there weren't any bigrams that began with the most popular "word"? *Hint: Read the documents closely!* _____

_____

8) To choose the third "word", Soekia:

- Looks at the **3** tab to find the most frequently occurring ***trigram*** that begins with the first and second "words".
- If there isn't one, it will look in the **2** tab for the most frequently occurring ***bigram*** beginning with the second "word".
- If there isn't one, it will return to the **1** tab and select the next most popular ***unigram***.

What do you expect Soekia to choose? _____ Which list did you select it from? _____

9) Continuing this process, what do you expect Soekia to choose for the:

fourth "word"? _____ fifth "word"? _____ sixth "word"? _____

Testing our Prediction

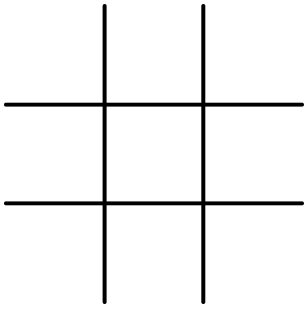
For this section, make sure you are still in the ***Intelligent Monkeys?*** *collection with the temperature set to* ***low****.*

10) How does Soekia answer the question *How intelligent are monkeys?…* when you click ➢ _____

11) How does that text compare to your prediction? _____

Tic Tac Toe

Warmup: Play A Game of Tic Tac Toe



In case you need a refresher on the game:

- The tic-tac-toe board is a 3x3 grid.
- One person will draw an **X** in one of the squares.
- The other person will draw and **O** in one of the squares.
- Keep taking turns — the goal is to get three in a row or block your neighbor from getting three in a row.
- The game ends when one of you gets three in a row or the grid is full.

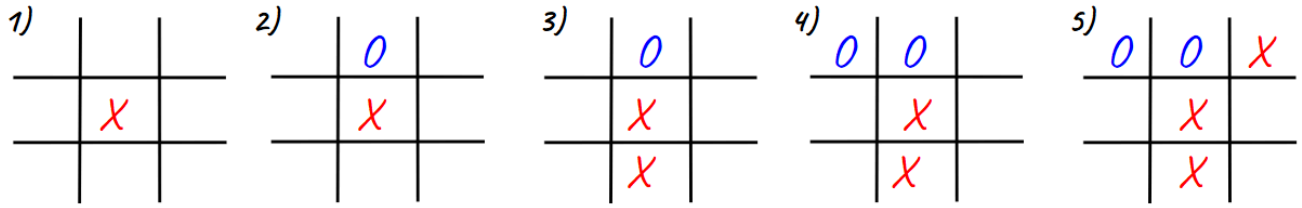
Design a Tic Tac Toe Notation

In order to communicate with Soekia about tic tac toe games, we'll need to record the moves using an annotation.

- Let's think of the 3x3 tic-tac-toe grid as a first quadrant coordinate plane with the origin (0,0) in the bottom left corner
- For each move, our notation must indicate:
  - the player whose turn it is (X or O)
  - the ordered pair (x, y) for the location of the player's move on that turn

1) How would you annotate the first turn (above)? _____ The second turn? _____

Annotating a Game of Tic Tac Toe



2) Complete the table by translating the 5 turn sequence above into our tic tac toe annotation.

For Reference:

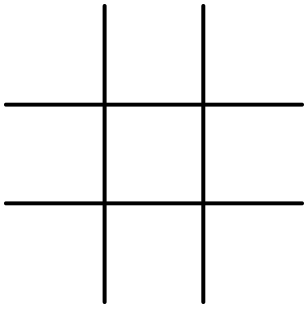
- *If player X makes a move in the bottom right corner, we would describe that turn as: X31*
- *If player O makes a move in the middle of the left column, we would describe that turn as: O12*

1st move	2nd move	3rd move	4th move	5th move

Translating Notation to the Game Board

3) Translate the list of moves below into a game played on the tic tac toe board.

- X32
- O22
- X12
- O11
- X33
- O13
- X31



4) Is there a winner? _____

What Makes a Language?

Respond to the prompts below by interacting with https://bootstrapworld.org/Soekia/.

Soekia & Tic-Tac-Toe

Access the *Documents* panel by clicking the "LOOK INSIDE →" button in the upper right-hand corner of Soekia.

- Click and select from the drop-down menu.

1) Describe what you see. _____

2) How are the documents of the tic-tac-toe collection **similar** to the documents of the fairy tale collection? How are they **different**?

In the *Generate Text* panel,

- Click to tell Soekia to generate a set of game moves.
- Click **TicTacToe-player** (beneath the list of moves) to see a visualization of the game you just generated. *Note that the player only appears after you have generated and paused a tic-tac-toe game.*

3) In what ways does the game seem **similar** to a normal game of Tic Tac Toe? In what ways does it seem **different**?

Thinking About Natural Language Processing

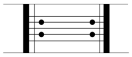
In the *Documents* panel,

- Click and select from the dropdown menu.

In the *Generate Text* panel,

- Click to tell Soekia to write you some music. Click **Music-Player** to try playing your composition.

4) What do you Notice? What do you Wonder? _____



5) In musical notation, a "repeat sign" (pictured on right) indicates that a section of the music should be repeated. Does Soekia's music player repeat when it arrives at a repeat sign? Why or why not? (*Stuck? Test it out in Soekia.*)

In the *Documents* panel,

- Click and select from the bottom of the dropdown menu.

In the *Generate Text* panel,

- Click to tell Soekia to generate a set of chess moves. Click **Chess-Player** to visualize the game.

6) In actual chess, a player starts with two knights and there is no way to gain additional knights. How come additional knights sometimes appear in this chess game?

Contracts for Ai

Contracts tell us how to use a function, by telling us three important things:

1. The **Name**
2. The **Domain** of the function - what kinds of inputs do we need to give the function, and how many?
3. The **Range** of the function - what kind of output will the function give us back?

For example: The contract `triangle :: (Number, String, String) -> Image` tells us that the name of the function is `triangle`, it needs three inputs (a Number and two Strings), and it produces an Image.

With these three pieces of information, we know that typing `triangle(20, "solid", "green")` will evaluate to an Image.

Name	Domain	Range
`# circle`	`::  ( Number , String, , String )` `radius  fill-style  color`	`->  Image`
`circle(50, "solid", "purple")`		
`# ellipse`	`::  ( Number , Number , String , String )` `width  height  fill-style  color`	`->  Image`
`ellipse(100, 50, "outline", "orange")`		
`# isosceles-triangle`	`::  ( Number , Number , String , String )` `size  vertex-angle  fill-style  color`	`->  Image`
`isosceles-triangle(50, 20, "solid", "grey")`		
`# overlay`	`::  ( Image , Image )` `top  bottom`	`->  Image`
`overlay(circle(10, "solid", "black"), square(50, "solid", "red"))`		
`# radial-star`	`::  ( Num , Num , Num , Str , Str )` `points  outer  inner  fill-style  color`	`->  Image`
`radial-star(6, 20, 50, "solid", "red")`		
`# rectangle`	`::  ( Number , Number , String , String )` `width  height  fill-style  color`	`->  Image`
`rectangle(100, 50, "outline", "green")`		
`# regular-polygon`	`::  ( Number , Number , String , String )` `size  vertices  fill-style  color`	`->  Image`
`regular-polygon(25,5, "solid", "purple")`		
`# rhombus`	`::  ( Number , Number , String , String )` `size  top-angle  fill-style  color`	`->  Image`
`rhombus(100, 45, "outline", "pink")`		
`# right-triangle`	`::  ( Number , Number , String , String )` `leg1  leg2  fill-style  color`	`->  Image`
`right-triangle(50, 60, "outline", "blue")`		
`# rotate`	`::  ( Number , Image )` `degrees  img`	`->  Image`
`rotate(45, star(50, "solid", "dark-blue"))`		

Name	Domain	Range
# sqr	:: (Number)	-> Number
sqr(4)		
# sqrt	:: (Number)	-> Number
sqrt(4)		
# square	:: (Number size , String fill-style , String color)	-> Image
square(50, "solid", "red")		
# star	:: (Number radius , String fill-style , String color)	-> Image
star(50, "solid", "red")		
# star-polygon	:: (Number size , Number point-count , Number step-count , String fill-style , String color)	-> Image
star-polygon(100, 10, 3 ,"outline", "red")		
# string-contains	:: (String haystack , String needle)	-> Boolean
string-contains("hotdog", "dog")		
# string-length	:: (String)	-> Number
string-length("rainbow")		
# text	:: (String message , Number size , String color)	-> Image
text("Zari", 85, "orange")		
# triangle	:: (Number size , String fill-style , String color)	-> Image
triangle(50, "solid", "fuchsia")		
# triangle-asa	:: (Number top-left-angle , Number left-side , Number bottom-angle , String fill-style , String color)	-> Image
triangle-asa(90, 200, 10, "solid", "purple")		
# triangle-sas	:: (Number bottom-R-side , Number top-R-angle , Number top-side , String fill-style , String color)	-> Image
triangle-sas(50, 20, 70, "outline", "dark-green")		
	::	->
	::	->
	::	->
	::	->

These materials were developed partly through support of the National Science Foundation (awards 1042210, 1535276, 1648684, and 1738598) and are licensed under a Creative Commons 4.0 Unported License. Based on a work at www.BootstrapWorld.org. Permissions beyond the scope of this license may be available by contacting contact@BootstrapWorld.org.

These materials were developed partly through support of the National Science Foundation (awards 1042210, 1535276, 1648684, and 1738598) and are licensed under a Creative Commons 4.0 Unported License. Based on a work at www.BootstrapWorld.org. Permissions beyond the scope of this license may be available by contacting contact@BootstrapWorld.org.