# Programming practice problems

Please save each solution in its own file, in a practice problems folder.

# Wordplay

A file containing all valid Scrabble words, one word per line:

<https://www.dropbox.com/s/qkg62nkh483g635/sowpods.txt?dl=0>

**For loops and if conditions**

1. What are all of the words containing UU?
2. What are all of the words containing an X and a Y and a Z?
3. What are all of the words containing a Q but not a U?
4. What are all of the words that contain the word CAT and are exactly 5 letters long?
5. What are all of the words that have no E or A and are at least 15 letters long?
6. What are all of the words that have a B and an X and are less than 5 letters long?
7. What are all of the words that start and end with a Y?
8. What are all of the words with no vowel and not even a Y?
9. What are all of the words that have all 5 vowels, in any order?
10. What are all of the words that have all 5 vowels, in alphabetical order?

**Setting up storage to use during a for loop, including counters and arrays**

1. How many words contain the substring "TYPE”?
2. Create and print an array containing all of the words that end in "GHTLY"
3. What is the shortest word that contains all 5 vowels?
4. What is the longest word that contains no vowels?
5. Which of the letters Q, X, and Z is the least common?
6. What is the longest palindrome?
7. What are all of the letters that never appear consecutively in an English word? For example, we know that “U” isn’t an answer, because of the word VACUUM, and we know that “A” isn’t an answer, because of “AARDVARK”, but which letters never appear consecutively?

# Countries

A file containing all of the countries in the United Nations, one country per line:

<https://www.dropbox.com/s/k6xcq55tqanbdmz/countries.txt?dl=0>

<https://www.dropbox.com/s/k6xcq55tqanbdmz/countries.txt?dl=0>

**For loops and if conditions**

1. What are all of the countries that have “United” in the name?
2. What countries both begin and end with a vowel?
3. What country names are > 50% vowels?

**Setting up storage to use during a for loop, including counters and arrays**

1. What is the shortest country name? Make sure your solution can handle ties.
2. What countries use only one vowel in their name (the vowel can be used multiple times)
   * For example, if the word “BEEKEEPER” were a country, it would be an answer, because it only uses “E”.
3. There is at least one country name that contains another country name. Find all of these cases.

# Baby names

A file containing the top 40 baby names for boys in 2020:

<https://www.dropbox.com/s/drpcy1yrsq4uadk/baby_names_2020_short.txt?dl=0>

A file containing the top 40 baby names for boys in 1880:

<https://www.dropbox.com/s/fvdi7jl280lqp8c/baby_names_1880_short.txt?dl=0>

**More for loops, if conditions, and storage**

1. What is the shortest baby name in the top 40 baby names for 2020?
2. What are the longest baby names in the top 40 baby names for 2020? Make sure you can handle if there’s a tie.

**Nested for loops**

1. There is at least one baby name from the top 40 baby names for 2020 that, when spelled backwards, is a valid Scrabble word. Find and print all such names.
   1. Solve this two ways: first with an array to hold the Scrabble words, and then with a dictionary (or set) to hold the Scrabble words. Use timer functions to measure how long it takes to complete this work under each implementation. Why is the time different?
2. What are all of the names that were top 40 baby names in both 1880 and 2020?

# NBA Finals

A CSV containing NBA finals data for every year:

<https://www.dropbox.com/s/8dx99bjfwh9eh38/nba_finals.csv?dl=0>

<https://www.dropbox.com/s/8dx99bjfwh9eh38/nba_finals.csv?dl=0>

**Dictionaries**

Read the NBA finals CSV data into one more more dictionaries as needed to complete the following:

1. Write a function that takes as an argument a year and returns the winner of the NBA finals that year.
2. Write a function that takes as an argument a team name and returns an array of all of the years the team has won the NBA finals.
3. Which teams have made it to the NBA finals but have never won?
4. Print out a ranking of who has won the MVP more than once, by times one, e.g. this output:
   * 6 times: Michael Jordan
   * 3 times: Shaquille O'Neal, LeBron James
   * 2 times: <etc>

# Billboard Hot 100 of 2000

A CSV containing the Billboard Hot 100 data for every week of 2000:

<https://www.dropbox.com/s/v56vasmua65qjgy/billboard100_2000.csv?dl=0>

<https://www.dropbox.com/s/v56vasmua65qjgy/billboard100_2000.csv?dl=0>

1. Print out all of the #1 songs and the artists who made them. If a song was #1 for more than one week, only print it once. Example output:
2. These were the number one songs of 2000:
3. "Try Again" - Aaliyah
4. "Say My Name" - Destiny's Child
5. "What A Girl Wants" - Christina Aguilera
6. "Maria Maria" - Santana Featuring The Product G&B
7. "Smooth" - Santana Featuring Rob Thomas
8. "Independent Women Part I" - Destiny's Child

…

1. What song was the #1 song for the most weeks of 2000, who was the artist, and how many weeks was it at #1?
2. What artist had the most songs chart in 2000, and what were those songs?
3. What song(s) were on the charts (anywhere on the charts) for the most weeks of 2000?

# Interview question: Scrabble solver

This is a real 60-minute interview question at a tech company.

## Part 1

Write code that:

* Accepts a string (either as a command-line argument or as an argument to a function)
* Prints out the words that can be made from those characters, along with their Scrabble scores, one word per line, in descending score order

Example input and output:

$ python scrabble\_cheater.py SPCQEIU  # Use any language you like.

17 piques

17 equips

16 quips

16 pique

16 equip

15 quip

…

Resources:

* [Word list](https://www.dropbox.com/s/qkg62nkh483g635/sowpods.txt?dl=0)
* [Letter scores](https://www.dropbox.com/s/talrnaxaftbb1rz/letter_scores.txt?dl=0)

## Part 2

Extend the script to handle blank tiles. When reading the input, the character \_ can be used as a wildcard — it can represent any letter.

Wildcards do not count towards a word's score.

# Interview question: rot solver

## Part I

(If you’ve heard of a rot13 letter substitution cipher, this question is a generalization of that cipher)

Write a function rot that:

* takes as arguments: an input string and an amount by which to shift the letters in the string
* returns: the input string, shifted by the shift amount

The function should preserve case — it should be able to handle both upper and lowercase letters — and it should not alter punctuation. The function should support negative numbers. The function should support large shift numbers.

Sample inputs and outputs:

rot("HELLO", 1) -> "IFMMP" # shift right by 1

rot("HELLO", 2) -> "JGNNQ" # shift right by 2

rot("HELLO", -1) -> "GDKKN" # shift left by 1

rot("HELLO", 27) -> "IFMMP" # shift right by 27, wrapping back to the beginning

rot("Hello, Rick", 1) -> "Ifmmp, Sjdl" # Preserve case and punctuation

rot(rot("Hello, Rick", 1), -1) -> "Hello, Rick"

Writing this function will require familiarity with converting between character and ordinals. For example, Python has the ord and chr functions, and JavaScript has the charCodeAt and fromCharCode String methods.

You may also find reviewing modular arithmetic (using %) to be helpful.

## Part II

Using your rot function, write a function decrypt that takes a text encrypted using a shift substitution cipher of an unknown shift amount, and returns a tuple containing (the shift used to encrypt the original string, the original string).

You will need a dictionary or word list. An input string needs to be long enough to unambiguously determine the the shift used, or there could be multiple valid shifts.

Sample inputs and outputs:

decrypt("Ju xbt uif cftu pg ujnft, ju xbt uif xpstu pg ujnft") -> ("It was the best of times, it was the worst of times", 1)

# Interview question: Boggle solver

This is a retired coding question from Dropbox. I don’t know how often candidates would actually solve this completely during a 60-minute interview, so instead I’m going to break this into a couple of pieces with expectations for each.

## Part 1

Implement a recursive depth-first search that you fully understand and could reproduce in front of someone from scratch if you needed to.

Then, run your DFS implementation on some example graphs. This means being comfortable with how to represent a graph with nodes and edges in your preferred interview language.

This will become the foundation for the Boggle solver.

## Part 2

Review how to play Boggle. [Here](https://www.puzzle-words.com/boggle-4x4/) is an example online version.

The goal of the game when played with humans is to find as many words as you can in a grid of 16 letters, in a limited amount of time. Words can only be made from connected letters — i.e. from a given letter you can only use the letters directly adjacent, including diagonally. You can’t reuse letters, and words must be at least 3 letters long.

## Part 3

Implement a Boggle solver. You will likely want to break this into the follow steps:

1. Stub out a function that takes as input a grid (e.g. a 2 x 2 or 4 x 4 matrix of letters) and returns all of the words that can be made from that grid.
2. Write a function to create a graph from the grid. *[Goal: complete this sub-part in 30 minutes]*
   1. Refer back to your graph representations from Part I.
   2. Test that you get the correct graph back from a grid.
3. Set up a data structure containing all words in the dictionary, with efficient lookup. *[Goal: complete this sub-part in < 10 minutes (you probably already have this implemented if you’ve completed the prior problems in this problem set)]*
4. Use depth-first search to find all of the words in the grid. *[Goal: complete this sub-part in 60 minutes]*
   1. You’ll need to run DFS starting from every letter.
   2. You don’t need any fancy data structures to solve a 4x4 grid quickly.
   3. You’ll need a way of confirming if a path through the grid makes a valid word.
   4. You might find it helpful (both as a light efficiency optimization and for easier debugging) to stop pursuing paths in the grid that cannot possibly lead to a valid word, i.e. that path is not a valid prefix for any word in the dictionary.

## Example inputs and outputs

**Example 1**

Grid:

BE

TQ

Valid words: {'BET'}

**Example 2**

Grid:

ZQQZ

ZAEZ

ZUDZ

ZQQZ

Valid words: {'ZUZ', 'ZED', 'ADZ', 'QUAD', 'EAU', 'DAE', 'DUE', 'ZEA', 'QUA', 'ADZE', 'AUE', 'ZZZ'}

**Example 3**

Grid:

MSEF

RATD

LONE

KAFB

Valid words: {'FONDEST', 'TOR', 'RANTED', 'OAF', 'FETES’ … 486 words in all}

# Interview question: Word Pattern

This is a retired 60-minute coding question from Dropbox. A good goal would be to be able to finish Part 1 within the 60 minutes, and then to come up with a plan for Part 2 even if you don’t have time to implement it fully.

## Part 1

Write a function that takes as arguments two strings: pattern and input. Return whether or not the words in input match the pattern of the characters in pattern.

Example 1:

|  |
| --- |
| pattern: 'abba' |
| input: 'red blue blue red' |
| result: True |

Example 2:

|  |
| --- |
| pattern: 'abcabc' |
| Input: 'red blue green red blue green' |
| result: True |

Example 3:

|  |
| --- |
| pattern: 'abba' |
| Input: 'red blue green red' |
| result: False |

## Part 2

Write a function that takes as arguments two strings: pattern and input. Return whether or not input can be broken into words to match the pattern of the characters in pattern.

(In other words, this is the same problem as part 1, but input doesn’t contain spaces, so you’ll need to determine if it is possible to split up the input into words in a way that matches pattern. You will likely want to use recursion.)

Example 1:

|  |
| --- |
| pattern: 'abcba' |
| input: 'redbluegreenbluered' |
| result: True |

Example 2:

|  |
| --- |
| pattern: 'aba' |
| Input: 'xxyyyxx' |
| result: True, with multiple solutions:   * x, xyyyx, x * xx, yyy, xx |

Example 3:

|  |
| --- |
| pattern: 'abba' |
| Input: 'redbluegreenred' |
| result: False |