# Python Coding Challenges

## 1. Saying Hello

Write a program that can say hello to everyone named in a list. The list should be defined like this:

```
names = [ "Helen", "Bob", "Sarah" ]
```

Write code that loops through each name in the list, and displays a greeting. The output should look like this:

```
Hello, Helen
Hello, Bob
Hello, Sarah
```

Next, change the array initialisation statement so the names array contains four names:

```
names = [ "Helen", "Bob", "Sarah", "Linda" ]
```

Make sure your code still works and now prints four greetings, without you having to make any other changes.

## 2. Saying Hello, in Reverse

Write a new program that can say hello to everyone named in a list, but in reverse order. The list should be defined like this:

```
names = [ "Helen", "Bob", "Sarah" ]
```

The expected output is:

```
Hello, Sarah
Hello, Bob
Hello, Helen
```

## 3. Multiplication table

Write a program that can display a multiplication table (up to x 10) for a number input by the user. For example, if the user input the number 7, then the program would display the 7 times table as follows:

```
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
```

## 4. Blast off

Write a program that uses a loop to display these lines:

```
10
9
8
7
6
5
4
3
2
1
Blast Off!
```

However, it should leave a 1 second delay between displaying each line. You can achieve a 1 second delay by *importing* the **time** module, then calling **time.sleep(1)**

## 5. Summing numbers 1 - 100

Write a program that adds up all the numbers from 1 to 100 (e.g. 1 + 2 + 3 + … + 99 + 100), and displays the answer.

When you test your code, check you get the correct answer: 5,050.

## 6. Summing numbers 1 - 100, but skipping every 4th number

Repeat the above exercise, but this time miss out all numbers which are multiples of 4 (e.g. add up 1 + 2 + 3 + 5 + 6 + 7 + 9 + 10 + 11 + 13 + 14 + 15 + 17 + … + 99 + 100).

When you test your code, check you get the correct answer: 3,750.

## 7. Biggest number

Ask the user to input 5 numbers, using a loop (one number input each time). Store each number in an ever-growing list after it is typed in. The numbers can be entered in any order, e.g. 54, 216, 12, 99, 11.

After the numbers have been typed in, find and display the biggest number in the list.

Extra challenge: can you do this a different way, which doesn't require a list to be stored, but which can still tell you the biggest number that was entered?

## 8. Fizz Buzz

Generate and display a list of numbers [1, 2, 3, 4, 5, …] BUT - if a number is a multiple of 3 it is replaced with the string "FIZZ", if it is a multiple of 5 it is replaced by the string "BUZZ", and if it is a multiple of both 3 and 5, it is replaced with the string "FIZZ BUZZ". Allow the user to say how long you want the list to be.

## 9. Palindrome

Input a string, and display a message to say whether or not the string is a palindrome (the same forwards as backwards). For example, "ABBA" is a palindrome, but "ACDC" is not.

## 10. Fibonacci generator

Calculate and display the first 50 numbers in the Fibonacci series (see below for what that means).

In the Fibonacci series, the next number is the sum of the previous two numbers (after 0 and 1).

So, the sequence starts like this: [ 0, 1, 1, 2, 3, 5, 8, 13, 21 ... ]

## 11. Who's in the building?

Write a program that can keep track of who's in a building, as people check in and check out.

Keep going round a loop, asking the user to enter an event, which is either an "IN" event or an "OUT" event. The line entered by the user will be the word IN or OUT, followed by the name of the person who has just arrived or left.

After each event is input, the program should display a list of the names of the people who are currently in the building. You can assume that no two people in the building have the same name.

Extension: can you make the program print an error message if the user claims that someone has left who wasn't in the building, or someone arrives who was already in the building?

## 12. Estimated time of arrival

Input distance to go (miles) and current speed (miles per hour). Calculate and display the current estimated time of arrival, based on the time now.

## 13. Anagram detector

Input two string parameters, and say whether or not they are anagrams of each other (all the same letters, different order).

## 14. Sleeps until next birthday

Input date of birth, including year. Calculate and display how many sleeps until your next birthday.

## 15. Text to Morse Code converter

Input some plain text, and print the Morse code equivalent (a sequence of dots and dashes that can be transmitted as beeps or flashes of light, as a primitive way of communicating a message).

For example, "SOS" → "... --- ...".

See here for a list of all the morse code letter patterns: https://en.wikipedia.org/wiki/Morse_code

Note that there should be a gap (space) between individual morse code letters, and a longer space between individual words.

## 16. Alphabet timer

Time how long it takes the user to input all 26 letters of the alphabet (in the correct order). Check the user entered the correct sequence. Show their time.

## 17.Train message display

Write a function that takes a list of train stations that a train will be calling at, and returns a message suitable for the scrolling display in the carriage. For example, the list [ "Rochdale", "Halifax", "Bradford", "Pudsey", "Leeds" ] should generate the message "This train is for Leeds, and will be calling at Rochdale, Halifax, Bradford, Pudsey and Leeds".

Make it work for any list, including ones with a single destination. Take care with the commas and "and" in the message!

## 18.Word count

Input a string from the user. Analyse the text to count the words in the text, and display the result.

Bonus: how well does your code cope with hyphenated words, commas, full stops, numbers, etc?

## 19.Random password generator

Generate a random 10-character password. The password must include at least one uppercase letter, at least one lowercase letter, at least one digit and at least one punctuation / special character (such as # ! @ $ %).

## 20.Higher / lower

Shuffle the numbers 1 - 10 into a random order. Show the user the first number, and invite them to guess whether the next number in the sequence is higher or lower. Reveal the actual next number, and whether the user was right. If they were wrong, the game is over. If they were right, continue until they have got to the end of the sequence - then they have won. Show them how many they got right when the game ends.

## 21.Simple encryption and decryption

Input a string from the user. Shift every character by 5, e.g. A → F, B → G, Y → D (wrap around). Output the "encrypted" text.

Bonus: how do you deal with other characters in the message, such as spaces, punctuation or numbers?

Extension: allow the user to enter another string, and do the reverse (decrypt it), showing the plain text result.

## 22.Morse Code to text converter

Input a line of of morse code (expressed as dots, dashes, spaces and multi-spaces), and turn it back into plain text. See above link for an explanation of the reverse process.

## 23.Guess the number

Write a program that thinks of a secret random number between 1 – 100. Keep repeating the following sequence until the user has correctly guessed the secret number…

- Ask the user to make a guess (they type in an integer)
- If they didn't type in an integer, keep asking them until they do so
- If the user guessed correct, tell them so, and tell them how many guesses it took them
- Otherwise, tell the user whether their guess was "too high" or "too low"
- Repeat until game is over

Enhancements:

- Have a limit of, say, 5 guesses. If the player hasn't won by then, tell them it's game over and they've lost.

## 24.How many guests?

Write a program to work out how many guests (including yourself) you can afford to attend your birthday meal. Here are the constraints:

- You have a budget of £100
- You will go to a restaurant where the cost of each meal is 5% cheaper than the one before
- The first meal costs £12
- So, the next one costs £11.40 (12.00 * 0.95), then the next one costs £10.83 (11.40 * 0.95)…
- You can't invite someone if you can't pay for their meal in full

Make the program display the following information:

- How many guests can come (including you)?
- How much does the last guest's meal cost?

Extension challenge:

- Also calculate and display the amount of money left over from your £100 budget
- Also calculate and display the average price of all the meals

*To check it's working…*

- *You should find you can invite 10 guests, with £3.70 left over, with the cheapest meal being £7.56 and the average meal price being £9.63*

## 25.Christmas Tree

Ask user for the size of tree, then display a tree of that size using asterisks, as shown below.

Example tree size 3:

```
  *
 ***
*****
  *
  *
```

Example tree size 4:

```
   *
  ***
 *****
*******
   *
   *
```

## 26.Rock, Paper, Scissors

Write a program to play a game of Rock, Paper, Scissors. Input player 1's choice then input player 2's choice, and display either "Player 1 wins", "Player 2 wins" or "Draw" depending on which player won the round.

For example, if player 1 chose "rock" and player 2 chose "scissors", it would say "Player 1 won" because rock blunts scissors.

The full rules are:

- Rock blunts scissors (the player who chose rock wins)
- Scissors cut paper (the player who chose scissors win)
- Paper wraps rock (the player who chose paper wins)
- If both players chose the same thing, it's a draw
- If one of the players picks something not on the list, the other player automatically wins
- If both players choose something not on the list, it's a draw

## 27.Snakes and Ladders

Write a program to simulate a very simple snakes and ladders game.

Player starts at position 1.

Keep repeating the following sequence until the player reaches position 100…

- Player rolls dice (program picks a random number between 1 and 6)
- Player moves that many squares forwards (e.g. was on square 5, rolls 3, moves to square 8)
- Any squares which are multiples of 17 (e.g. 17, 34, 51, 68, 85) but not 97 are ladders: if player lands on a ladder they advance 12 spaces
- Any squares which are multiples of 13 (e.g. 13, 26, 39, 52, 65, 78, 91) are snakes: if player lands in a snake, they go back 12 spaces
- Players cannot go forwards beyond square 100 - if a roll would take them beyond 100 then they go *back* that many spaces instead!
- Players cannot go backwards beyond square 1

At each step of the game, display what's happening (where the player is, what they rolled, where they moved to, up a ladder, down a snake, etc.)

When the game finishes, tell the player the total number of goes they took.

Enhancements:

- Make this a two-player game. Your program tracks each player's progress, alternating who's go it is. The first player to get to 100 wins.
- Can you find a way to do this without repeating the code for most of the logic for each player?

## 28.Hangman

Implement a simple game of hangman.

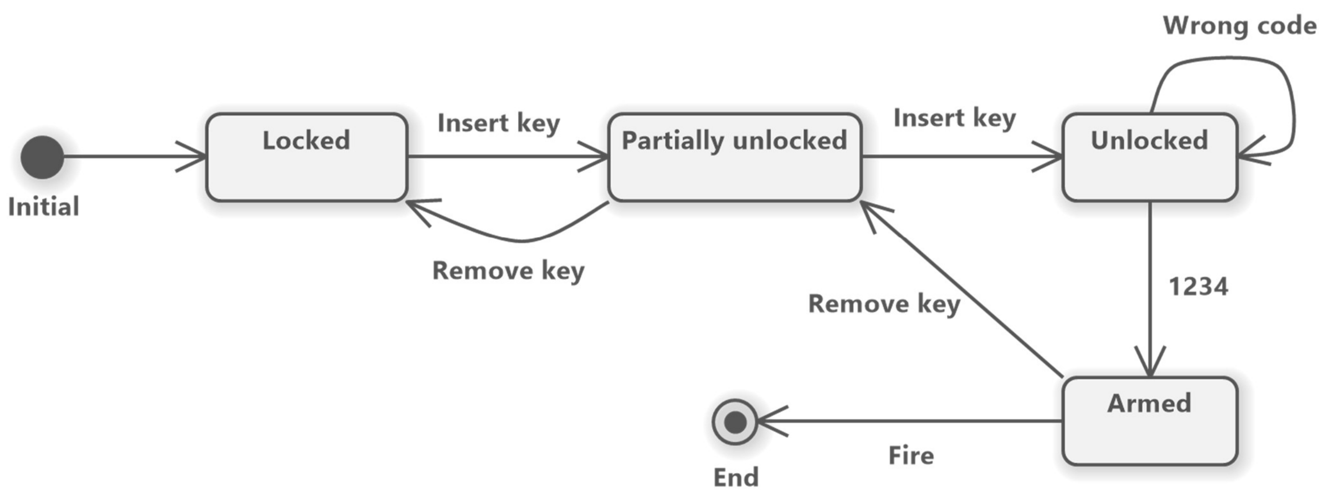
- Player 1 enters a word to be guessed
- Player 2 is shown the --------- pattern for the word (changes depending on length of word)
- Player two enters one letter at a time
- If letter is in word, player is told so, and ------- pattern changes to show position of all letters correctly guessed so far
- If letter is not in word, player told so, and count of incorrect guesses is incremented
- Game is over either when word has been guessed, or player runs out of moves (limit = 10?)

## 29.Missile launcher

Write a program to control a missile launcher. The system is secured by having two keys that must both be inserted, as well as a secret launch code. The program waits for the operator to type in commands (which are simulating real physical things happening, just so we can write this as a console application). The system responds by saying what state it is now in. The rules are as follows:

- The system is initially "locked".
- When the system is "locked" and the input is "insert key", the system becomes "partially unlocked".
- When the system is "partially unlocked" and the input is "remove key", the system becomes "locked".
- When the system is "partially unlocked" and the input is "insert key", the system becomes "unlocked".
- When the system is "unlocked" and the input is "remove key", the system becomes "partially unlocked".
- When the system is "unlocked" and the input is "1234", the system becomes "armed".
- When the system is "unlocked" and the input is any other number, the system state does not change.
- When the system is "armed" and the input is "key removed", the system becomes "partially unlocked".
- When the system is "armed" and the input is "fire", the system displays "Missile launched!" and exits.

The above rules are also shown in this "state machine" diagram.



## 30.Sorted shopping list

Write a program that can display a sorted version of a shopping list input by the user, where the items are sorted into the order they will be found in the store (to make it quicker to collect everything). You may hard-code into your program all the things sold by the shop, and the order in which they are positioned.

Let the user key in a list of items, one line at a time. When the user has finished entering their list, they type "END".

Then, display the list back tpo the user, but sorted in other in the order that the items are located in the store. Only include things in the output list that were actually on the user's shopping list!

For example, suppose that quickest route round all the isles in the shop takes you past veg, fruit, salad, meat, milk, bread, soup, water, chips, cat food and toilet roll. If your original shopping list was [ "cat food", "salad", "fruit", "milk" ], then the sorted list would be [ "fruit", "salad", "milk", "cat food" ].

## 31. Run length encoding compression

Input a string of letters, containing many repeats. Replace the repeated characters with a shorter version using a number, and display the "compressed" version of the origianl input. For example, the input "AAAAABBBBBBBAAAA" would be converted to "A5B7A4".

Extension: write another program to work the other way around, expanding compressed strings back the original version.

Bonus: can you make this work for sequences where the same letter is repeated more than 10 or more times?

## 32. Credit card number validator

Input a 16-digit credit card number as a string (e.g. "1234567890123456") and display whether or not the card number is potentially valid or definitely invalid.

NB: this isn't to do with whether the card number is for an actual real card; it's whether the first 15 digits of the card match with the 16th "check" digit. Research the "Luhn" algorithm to see how this is done. (The "Luhn" algorithm is designed specifically to detect transposed digits and "off by one" typing mistakes.)

Examples of (made up) valid and invalid credit card numbers are:

- 2720994469296833 = valid
- 5133991942799163 = valid
- 5195408152966387 = valid
- 7220994469296833 = invalid
- 5233991942799163 = invalid
- 5195408152966386 = invalid

## 33. Roman Numerals

Input an integer, and show the equivalent roman numerals string. For example, 24 → "XXIV". As above, do this without hard-coding each individual outcome ahead of time.

Bonus: can you write another program that works the other way, i.e. given a Roman Numeral number, decode it back into the actual number.

For details of how Roman Numerals work, see https://en.wikipedia.org/wiki/Roman_numerals

## 34. Convert number to text

Input an integer (e.g. 21), and display the text version of the number. For example, 21 → "twenty one". What limit will you put on the largest number you can convert?

Hint: try this without writing a giant if num == 1… if num == 2 piece of code; try to write it the shortest way possible, so that the answer is "worked out" rather than "looked up", as far as possible.