Computer Code for Beginners

Week 2

Logo Shapes 2

Last week we used a `while` loop to repeat parts of our program. This exercise will introduce using `for` loops, using the Logo Shapes programs we built. This will reduce the repeated code.

- Open the `triangle.py` file again and look at the lines that draw the triangle (16-20)
  - You'll notice that it repeats the same commands three times
- Make a new variable `sides` and give it the value `3` (since we're drawing a triangle)
- Identify the two commands that we repeat on lines 16-20 and put them into a `for` loop that repeats them for each side
  - The simplest way to do this is to say `for i in range(sides):`
- Run the program and check that it works as you expect
- Can you do the same thing with the `square.py` program?
- Now can you do the same thing with the `rectangle.py` program?
  - Hint: either change the number of times you loop or use an `if...else` statement inside the loop…

Day of the Week

Write a program that asks the user to input a word and checks if it is a valid day of the week. If the user input is a valid day of the week, then the program should also report if it is a week day or a weekend day. For example, if the user enters `Sunday` then the program should accept that as a valid day of the week and print that it is a weekend day.

Design:

- What variable type should each day of the week be?
- How will you store all the days of the week?
- How will you check if the user's input is a day of the week?
- You might want to try checking if `Sunday` (for example) if a valid day of the week first, then add in user input

Code:

- Ask the user for input
  - Use the `input()` function to ask the user for a (potential) day of the week
- Check the input
- Print the response to the user
  - Invalid (not a day)
  - Valid Weekday
  - Valid weekend day
- Now add a loop to the program, so that it asks the user for a word and checks it, until the user indicates they want to exit
  - The simplest way to do this is to loop until the user enters a particular string (`e` for example)
  - Make sure you tell the user how to exit!
- Test your program, try to produce each of the three different outputs

Challenge:

- Make your program work if the user types in a string in upper, lower, or mixed case

  - We can use the `upper()` and `lower()` operations of a string to change the case. E.g. `"TEST".lower()` will give us the string `test`
- Write a `checkDay(day)` function, which prints if the `day` parameter is invalid (not a day), a weekday, or the weekend
  - Define the function with `def checkDay(day)`
  - You now need to move the code you've written to check the user's input into the body of this function

Random Number Guessing Game

Write a program that generates a random number and asks the user to guess it. The user should be given a limited number of guesses (try 10) and asked to guess until they have used up their guesses or they get it right.

Design:

- What needs to happen only once and what needs to be repeated for each guess at the random number?
- Other than the user's guess, what data do you need to store?

Code:

- Generate a random number
  - Use `import random` to import the `random` module
  - Call `random.randrange(1, 101)` to generate a random number between 1 and 100 (it works like `range()`
- Ask the user for their guess
- Define a function `checkGuess(tries, guess)` that should:
  - Check if the `guess` parameter matches the random number we've generated
  - If the guess is correct, print a message telling the user that they guessed correctly and the number of tries it took them
  - If the guess is wrong, print a message telling the user if the random number higher or lower than their guess
  - The function should return the number of tries that the user has left
- Now repeatedly ask the user for a guess and check it, until they user guesses correctly or there are no more guesses left

Challenge:

- Make sure that the program can handle the user inputting something that isn't a number
  - We can check if a string `s` contains only numbers with `s.isdigit()`

Caesar Cipher

Write a program that can encrypt and decrypt strings using the Caesar Cipher, which simply shifts each letter in the string by a given offset (i.e. up x places in the alphabet). For example, encrypting the letter "a" and shifting it 1 place produces "b". Decryption is the reverse, shifting each letter down by the given offset. Remember that each character is represented by a unique number, in a character set called UniCode. We can find the UniCode number of the letter `a` by calling `ord("a")`, and we can find the letter for a UniCode number using `char(number)`. For simplicity, we don't need to return the string in the same case as it was originally.

Design:

- Remember, start off small and then add detail
- The UniCode number for:
  - Lowercase "a" is 97
  - Lowercase "z" is 122
  - The full list for the Basic Latin alphabet can be found online at www.unicode-table.com/en/#basic-latin
- Shifting a letter just requires us to add/subtract the offset to/from the letter's UniCode number
  - But, if we encrypt a "z" and shift it by 1, it needs to become an "a". You need to make sure that the your program checks if the new character is higher than "z" and 'wraps around'.
  - If we had three character codes - 1, 2, and 3 - how would you make it so that shifting 3 up by one 'wrapped around' to 1?

Code:

- Implement an `encrypt(string, offset)` function, which returns the `string` parameter after it has been shifted up by the given `offset`
  - Remember that a string is a sequence (like a list)
  - For now, change the case of the `string` parameter to lower case
- Non-alphabetic characters (like numbers, punctuation, or spaces) should remain the same in the string we return, we can check if a character `char` is an alphabetic character using `char.isalpha()`
- Test the `encrypt(string, offset)` function with some different strings and offsets
- Then implement a `decrypt(string, offset)` function, which returns the `string` after it has been shifted down by the `offset`
  - This will have a similar structure to the `encrypt()` function, but will shift the letter down the alphabet
- Now test the `decrypt(string, offset)` function
- Try encrypting a string and then decrypt the result to see if you get the original string back.

Use your program to decrypt these strings, each has an offset of 12:

- hqzu, hupu, huou
- qjbqduqzoq ue ftq fqmotqd ar mxx ftuzse

Challenge :

- Make your `encrypt()` and `decrypt()` functions work for mixed case strings
  - Each function should shift the string up/down but respect the case of each letter in the original string
  - You'll need to find the range of UniCode numbers for the uppercase letters
  - Your functions will need to check if each letter is an uppercase or lowercase letter before shifting it