# **Silly Sentences**

Predict .

Take a look at the code below. Read it carefully and try to make a prediction about what might happen when this code is executed.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | noun = "Car"  adverb = "gently"  adjective = "loud"  print(f"The {noun} was {adjective} when it {adverb} went to school")  noun = "Zebra"  adverb = "aggressively"  adjective = "giant"  print(f"The {noun} was {adjective} when it {adverb} went to school") |

**It will correctly print the first line of code as “the car was loud when it gently went to school” and “the zebra was giant as it aggressively went to school”**

Run .

Open and **run** the file with this code. A copy can be found [here](https://repl.it/@NCCE/Silly-Sentences) (ncce.io/py-silly-sentences) if needed.

Was your prediction correct? Did anything unexpected happen? Write down your thoughts below:

|  |
| --- |
| n/a |

Investigate .

|  |  |
| --- | --- |
| **Questions/activities** | **Your answers** ▿ |
| In which line is the variable adjective **initialised**? | line 3 |
| In which line is the variable adverb **first reassigned**? | Line 8 |
| When is the variable noun **first used**? | Line 4 |
| Is there a **difference** between the code in line 4 and line 8? | No differences |
| On **line 4**, remove the f after print and before the speech mark. Run the code and write down what happens.  **! Remember to place the f back in when you have finished.** | The {noun} was {adjective} when it {adverb} went to school |
| What do you think the f is used for? This [article](https://realpython.com/python-f-strings/) (realpython.com/python-f-strings/) might help with your understanding. | Allows to embed expressions within curly braces |
| On **line 4**, remove the curly {} brackets that surround noun and run the code again. What happens?  **! Remember to place the {} back in when you have finished.** | It says the noun was loud when it gently went to school |
| Why does it not display The Car Zebra was loud giant when it gently aggressively went to school when the code is executed? | because the defined string “noun” allows the computer to differentiate the two lines of code to give a different result to deliver the second line of code. |

Modify .

|  |  |
| --- | --- |
| **Activities** | **Hints**▿ |
| Change all values in both occurrences of noun, adjective, and adverb to something different. | You can search for lists of nouns, adjectives, and adverbs on the internet if you need some ideas.  Remember to keep the speech marks around the values. These are important. |
| Create a new variable called proper\_noun and **initialise** it as London. | Think about the placing of the variable when it is initialised. Where are all of the other variables initialised? Place it with those. |
| Replace the word school with {proper\_noun} in **both** print statements. | Be careful not to delete any important syntax. Look at the code on line 8 to make sure that you still have the right brackets and speech marks in place. |
| Make a change to your code that will ensure that the second print statement displays a different proper noun. | Remember that variables can be assigned something different later on in the code. Think carefully about placement. |
| Add a completely new silly sentence to the bottom of the code. You can use the same variables, but think of a different sentence to write. | Look carefully at the syntax used for the other print statements, and use this to help you find errors in your code. |

noun = "cat"

adverb = "stupidly"

adjective = "great"

proper\_noun= "London"

print(f"The {noun} was {adjective} when it {adverb} went to {proper\_noun}")

noun = "carton"

adverb = "spectacularly"

adjective = "broken"

proper\_noun= "Tahiti"

print(f"The {noun} was {adjective} when it {adverb} went to {proper\_noun}")

noun = "dog"

adverb = "thinly"

adjective = "ultimate"

proper\_noun= "Denmark"

print(f"The {noun} was {adjective} when it {adverb} went to {proper\_noun}")

Make .

Create your own silly story using the code from this activity as a guide. There are lots of ‘fill in the blank’ style stories online that you could use for inspiration.

1. Write a silly story
2. Decide which words should be blank
3. Assign variables to the blanks
4. Write the program

You don’t have to use an adverb, adjective, or noun. You could use words like popular\_restaurant or public\_transport. Be as creative as possible, the sillier the better!

from time import sleep

def silly1():

person = "Rob"

home = "Sheffield"

public\_transport = "hovercraft"

proper\_noun= "The Shard"

place= "London"

print(f"{person} took the {public\_transport} from {home} to {place} and base jumped from the {proper\_noun}")

sleep(2)

def silly2():

noun = "mattress"

crisps = "bag of wotsits"

adjective = "fall"

verb = "crumbled"

print(f"The {noun} broke his {adjective} but his {crisps} had {verb}")

sleep(2)

def silly3():

person = "he"

object = "meal deal"

verb = "to buy"

proper\_noun= "Pret a Manger"

print(f"{person} had {verb} a {object} from {proper\_noun}")

silly1()

silly2()

silly3()