Working with For Loops and If Statements

Here is what we worked on last on Day 2.

```
sWords = ["the", "bunny", "hops", "over", "the", "fence"]
print(*sWords)

sWords[0] = sWords[0].capitalize()
print(*sWords)

sWords.append(".")
print(*sWords)
```

After running the module. The result was:

```
The bunny hops over the fence .
```

The goal is now to remove the space between the last word and the period that ends the sentence.

We must abandon the print(*sWords) function to make this work. Instead we'll use a for loop and make decisions using an if-then-else mechanism.

Use pen and paper to trace the flow chart in *Figure 1 – Flow Chart* into *Table 1 - Tracing Task*.

The table is partially filled, and you must complete all the tracing steps.

If you are not sure how to proceed, check out the solution given in *Table 2 - Solution*Table 2.

Table 1 - Tracing Task

STATION	sSentence	nListSize	the last position	nPos, the variable for the current position	The word at the current position is…	Is the current position the last position?
1	“”					
2		7				
3			6			
4				0		
5					“The”	0 == 6? No.
6	“ The”					
7				1 = 0 + 1		
5					“bunny”	1 == 6? No.
6	“ The bunny”					
7						
5						
6						
7						
5						
6						
7						
5						
6						
7						
5						
6						
7						
5						
9						
10						

The table below shows the solution completing the tracing from the beginning till end.

Table 2 - Solution

STATION	sSentence	nListSize	the last position	nPos, the variable for the current position	The word at the current position is…	Is the current position the last position?
1	“”					
2		7				
3			6			
4				0		
5					“The”	0 == 6? No.
6	“ The”					
7				1 = 0 + 1		
5					“bunny”	1 == 6? No.
6	“ The bunny”					
7				2 = 1 + 1		
5					“hops”	2 == 6? No.
6	“ The bunny hops”					
7				3 = 2 + 1		
5					“over”	1 == 6? No.
6	“ The bunny hops over”					
7				4 = 3 + 1		
5					“the”	1 == 6? No.
6	“ The bunny hops over the”					
7				5 = 4 + 1		
5					“fence”	1 == 6? No.
6	“ The bunny hops over the fence”					
7				6 = 5 + 1		
5					“.”	6 == 6? Yes.
9	“ The bunny hops over the fence.”					
10						

The flow chart that follows was enhanced with commands to show how the instructions are translated into code. The code is explained immediately after the chart on “STATION” at a time.

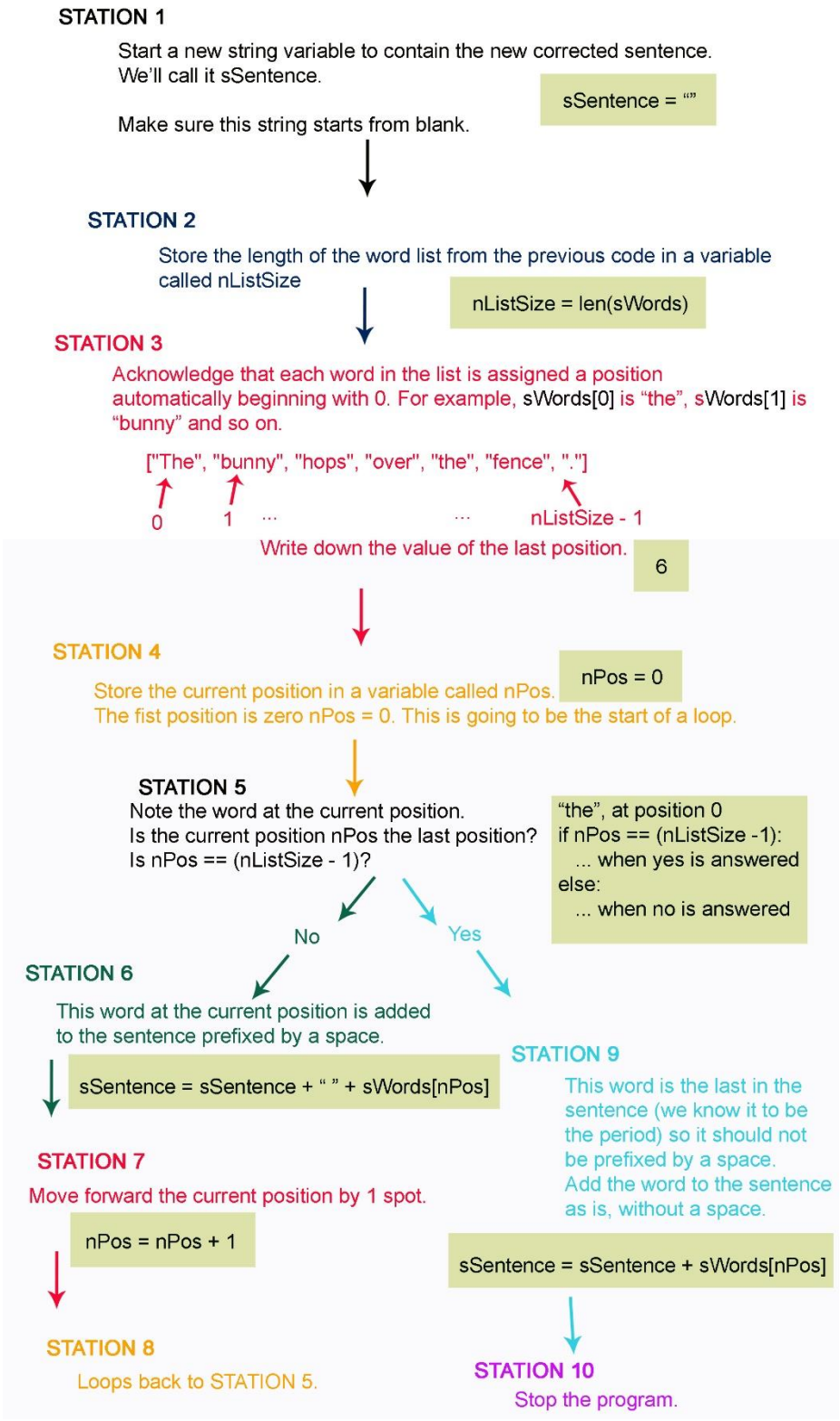


Figure 1 – Flow Chart

STATION 1

The following statements take the empty string "" and delivers it into the memory slot occupied by the variable sSentence.

```
sSentence = ""
```

As with all equalities in coding, the right side of the equal is reserved for the value we want to copy. In this case that is the empty string "", and on the left side we state the name of the variable that should store this value.

STATION 2

The len() function tells us the length of the list, i.e. how many elements are present. The function returns a count which is then stored in the variable nListSize based on how we organize around the equal sign as described in STATION 1.

```
nListSize = len(sWords)
```

The len() function tells us the length of the list, i.e. how many elements are present. The function returns a count which is then stored in the variable nListSize based on how we organize around the equal sign as described in STATION 1.

STATION 3

Count the number of elements in the list:

```
sWords = ["The", "bunny", "hops", "over", "the", "fence", "."]
```

It has 7 elements. Since the first element is considered to be position 0, we count 7 elements as follows: 0, 1, 2, 3, 4, 5, 6. Therefore the last element is at position 6.

STATION 4

The following statements take the number 0 and delivers it into the memory slot occupied by the variable nPos.

```
nPos = 0
```

Again, as with all equalities in coding, the right side of the equal is always the value we want to copy, and on the left side we state the name of the variable we want to store it in.

STATION 5

The following is a decision mechanism called if-then-else. It has a precise syntax, as shown below. This mechanism is often called an if-then-else-block. It can also be used as a simpler if-block when skipping

the else part.

The block has a question in the if section. This question can be answered with either a Yes or No answer, or a True or False answer. Dots show where to place code that relates to each part. The blue line shows where code should go if the questions was answered with a yes, and the red line shows where the code that related to the no answered should be written. You must **make sure that such code is indented** by one tab below the if, for blue, and below the else for the red. **Indentation** is how Python knows which commands belong to which block.

```
if nPos == (nListSize -1):

  ... when yes is answered

else:

 ... when no is answered
```

STATION 6

Just like in STATION 4, on the right side of the equal is the value we want to copy. On the left side we place the name of the variable to store it in.

```
sSentence = sSentence + " " + sWords[nPos]
```

First, we analyze the right side. If you traced the code into the tables above, you noticed that the current position used in the flow chart is **nPos**. Thus, the current word is **sWords[npos]**.

This word is to be prefixed with a space which means the space is set first. Therefore, we construct a string made of a space and the current word and glue them together with the + operator.

" " + sWords[npos]

This string is only the current word, but throughout the loop we have collected previous words and we don't want to lose them. These words would have been stored in the **sSentence** variable. By attaching the current word and its space to **sSentence** we are growing our string. What we built so far must appear first and again we glue things together using the + operator.

sSentence + " " + sWords[nPos]

Note the order is important.

We decide where to store this combination. The name of the variable to store in must be noted on the left side of the equal. In this case we use the variable: **sSentence = … .** We store our computation there, so we don't lose the result.

STATION 9

This station is nearly identical to STATION 6, except this part does not have the space.

In Conclusion…

Here is what the final code looks like. Add it to a new .py file and run the module in the IDLE console as shown in the previous lessons.

```
sWords = ["The", "bunny", "hops", "over", "the", "fence", "."]
sSentence = ""
nListSize = len(sWords)

for nPos in range(0, nListSize):
   sCurrentWord = sWords[nPos]
   if nPos == (nListSize -1):
       sSentence = sSentence + sCurrentWord
   else:
      sSentence = sSentence + " " + sCurrentWord

print(sSentence)
```

Run the module. The result is:

```
The bunny hops over the fence.
```

Notice the for loop syntax

for nPos in range(0, nListSize):

   ….

   ….

The for-loop does a bit of work for us. It picks out the first number in the brackets (in this case 0) into the variable nPos. Every time it runs through the loop (we call this an **iteration**) it increases nPos by one point. When nPos equals the second number in the brackets (here nListSize) it no longer dives into the loop and moves on to whatever statement appears after the for-loop.

In summary, 0 is the number to begin the loop with and gets assigned to nPos, and nListSize is the number that stops the loop. The loop increases nPos by one with each run of the loop. One cycle of the loop is called an **Iteration**.

```
   if nPos == (nListSize -1):
       sSentence = sSentence + sCurrentWord
   else:
      sSentence = sSentence + " " + sCurrentWord
```

Notice that Python uses indentation to show which lines of code belong to which part of the program. All statements above in the for-loop are indented by one tab (see dots …. above) to show they belong to the loop. This is similar to the indentation used by the for if-then-else block seen earlier.

Code indented below the **if** (see red line above) belongs to the part of the code that answers yes to the question "**is nPos == (nListSize -1)?**"

Code indented below the **else** (see blue line above) belongs to the part of the code that answers no to the question "**is nPos == (nListSize -1)?**"

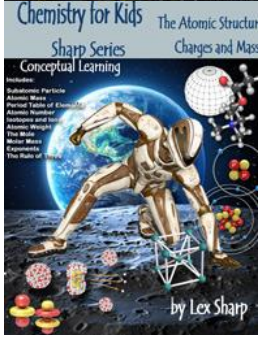
Make sure you related these questions and answers back to the flow diagram in *Figure 1 – Flow Chart*.

Books in Sharp Series

Find out more about **Workbooks 1 to 4** in this series at:
https://sharpseries.ca/scratch/w.html.

Find out about **Color Your Way to Math** Volumes 1 to 3 at:
https://sharpseries.ca/cyw/cyw2m.html.

Find out about **Early Math Concepts** at:
https://sharpseries.ca/em/v1.html.



Find out about **Chemistry for Kids** at:
https://sharpseries.ca/chem/v1.html.