

Mini-Lesson 4:

Pseudocode

Ralph Valery Valiere

What you will learn today:

- What a pseudocode is
- What a pseudocode is useful for
- Best Practices for Pseudocodes
- Test how to write a pseudocode (do-pair-share exercise after minilesson)

What is a pseudocode?

What a pseudocode is (1)

A pseudocode is :

- A **step-by-step *text*** outline of an algorithm/code that you can later translate into running codes.
- It combines **human language** (most of the time *English*) and **programming language**.
- A **reader** can quickly understand the main structure and ideas of your code.

In summary, it's a simplified combination of English and code outlines for an algorithm

What a pseudocode is (2)

- **Step-by-step:** It reflects the syntax of the actual algorithm/code
- **Text:** It's not a *Flowchart* (another way to outline an algorithm/code)
- **Human language:** Our concern is more about readability than technical specificities
- **Programming language:** It also includes some programming syntax and statements
- **Readers:** Even someone unfamiliar with your programming language can understand it.

Writing a pseudocoe in one picture



Source: A good father on [Reddit](#)

What a pseudocode is NOT

- It's *NOT* a full English description of your code
- It's *NOT* runnable codes
- It's *NOT* tied to a specific programming language
- It's *NOT* a graphical representation (diagram or chart)
- It *DOES NOT* replace your algorithm

There is *NO* universal rules/standard on how you should write a pseudocode. It is **subjective**.

While it should be clear, there is *NO* guarantee it will not be ambiguous for some people reading it.

Actual Code VS Pseudocode (1)

Using this dataset, I wrote a function to find the average lab attendance per week, only for 1st and last weeks.

```
1 import pandas as pd
2 import numpy as np
3
4 # This is fake data I created
5 lab_attendance_data = {
6     'lab_week': [1, 2, 3, 4, 5, 6, 7, 8],
7     'attendance': [15, 18, 16, 19, 18, 16, 19, 19]
8 }
9 lab_attendance = pd.DataFrame(lab_attendance_data)
10
11 # This is function we want to create for this example
12 def avg_firstlast(data):
13     fw_attendance = data['attendance'].iloc[0]
14     lw_attendance = data['attendance'].iloc[-1]
15     avg_attendance = np.mean([fw_attendance, lw_attendance])
16     return avg_attendance
```

Actual Code VS Pseudocode (2)

Pseudocode Example 1

Here is one way to write the pseudo code for the function we created earlier:

....

```
FUNCTION avg_firstlast(data):  
  BEGIN  
    fw_attendance <- first value in 'attendance' column in data  
    lw_attendance <- last value in 'attendance' column in data  
    avg_attendance <- average of fw_attendance & lw_attendance  
    RETURN avg_attendance  
  END  
ENDFUNCTION
```

....

This pseudocode is using more description.

Actual Code VS Pseudocode (3)

Pseudocode Example 2

Here is **ANOTHER** way to write the pseudo code for the function we created earlier:

....

```
FUNCTION avg_firstlast(data)
    first_val ← data.attendance[0]
    last_val ← data.attendance[-1]
    avg_attendance ← (first_val + last_val) / 2
    return avg_attendance
END FUNCTION
```

....

This pseudocode uses less English description, using more conventional expressions.

Why use a pseudocode?

Reasons why pseudocode is useful

If pseudocode is subjective, why even use it in the first place?

Here is why:

- Can help you clarify and structure your thoughts before writing runnable codes
- Describes how your algorithm/code works (explaining the process behind your code)
- Allows someone with no coding background to understand what your code is doing
- Makes reviewing codes easier
- Reduces the number of comments needed in your script

**What are the
pseudocode best
practices?**

Common conventions for pseudocode (1)

While there is no universal rules on how to write a pseudocode, there are some best practices:

Common conventions for pseudocode (2)

- **Capitalize the first keyword** to indicate the primary functions of each block (like FUNCTION)
- Some keywords like **IF, WHILE, FOR, FUNCTION** etc. are reserved by some conventions
- **Write exactly what you mean**, NOT exactly what you wrote as a code. You are not trying to reproduce your codes
- **Describe everything** that is happening in your code (DO NOT leave out anything)
- Use **standard code structure** as much as possible (For example, use indentation for this class)

**Do-pair-share
(OPTIONAL)**

Exercise to practice writing a pseudocode

Try to understand the code below and write its pseudocode.

```
1 import numpy as np
2
3 # Tip 1: First, try to understand what each line are doing
4 random_list = [12, 25, 32, 16, 33, 28, 18, 19, 21, 22]
5 updated_list = []
6
7 # Tip 2: Try to write pseudocode for each block
8 for num in random_list:
9     num_half = num / 2
10    updated_list.append(num_half)
11
12 # Tip 3: I decided to not add comments on purpose
13 final_list = updated_list[1:-1]
14
15 # Tip 4: This is optional. You can do it at home!
16 avg_list = np.mean(final_list)
```

Do-pair-share Solutions (1)

This is ONE pseudocode for the do-pair-share code

...

```
SET random_list TO [12, 25, 32, 16, 33, 28, 18, 19, 21, 22]
```

```
SET updated_list TO empty list
```

```
FOR EACH num in random_list DO:
```

```
    SET num_half TO num / 2
```

```
    APPEND num_half TO updated_list
```

```
END FOR
```

```
SET final_list TO all elements of updated_list  
except the first and last elements
```

```
SET avg_list TO the average value of final_list
```

...

Do-pair-share Solutions (2)

This is ANOTHER pseudocode for the do-pair-share code

...

```
random_list <- [12, 25, 32, 16, 33, 28, 18, 19, 21, 22]  
updated_list <- empty list
```

```
FOR EACH num in random_list:  
  num_half <- num / 2  
  APPEND num_half TO updated_list  
END FOR
```

```
final_list <- updated_list[1:-1]
```

```
avg_list <- average value of final_list
```

...