Bioinformatics Computing (BINF 5240)

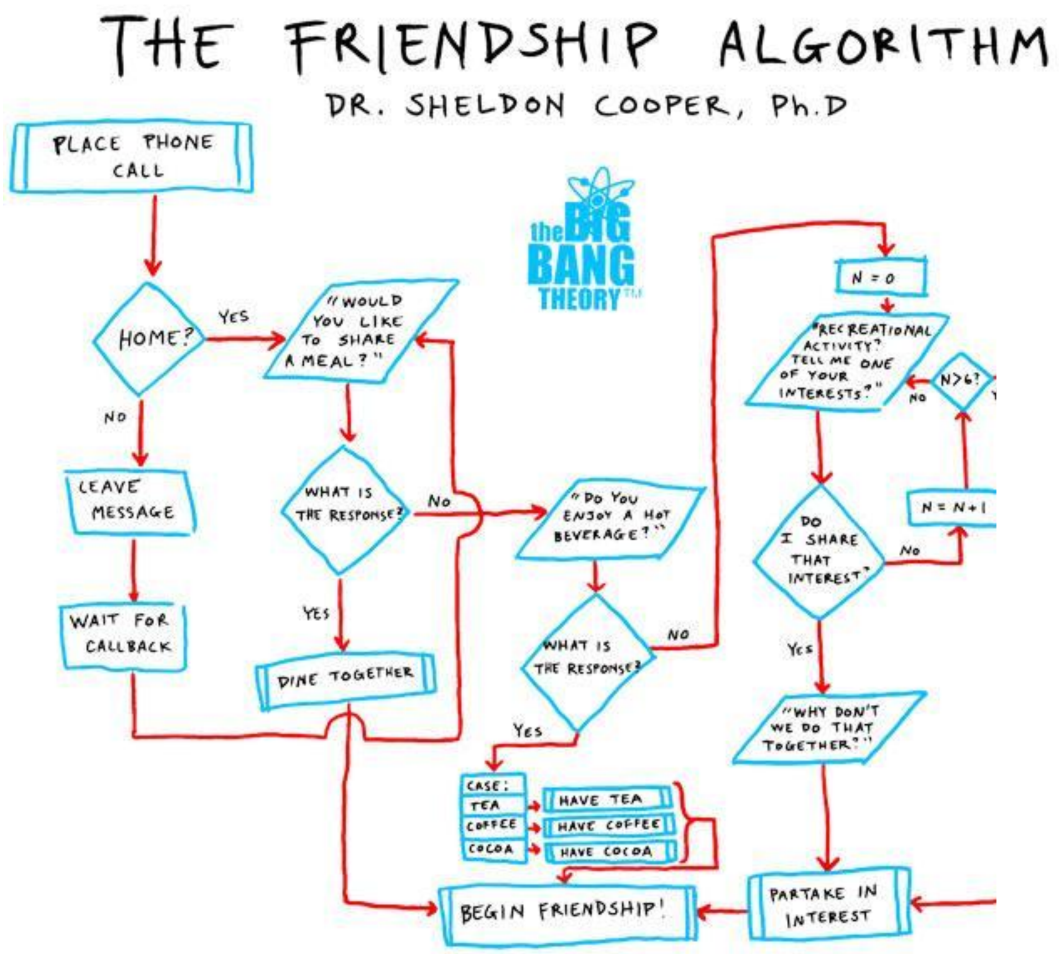
How to make a plan - Introduction to Pseudocode
(approx. 20-25 minutes)
Dr. Markus Hoffmann

The Friendship Algorithm

https://www.youtube.com/watch?v=OPHRgGc3A90

What is an algorithm?

- An algorithm is a sequence of instructions that one must perform in order to solve a problem.
- Define problems in terms of inputs and outputs.

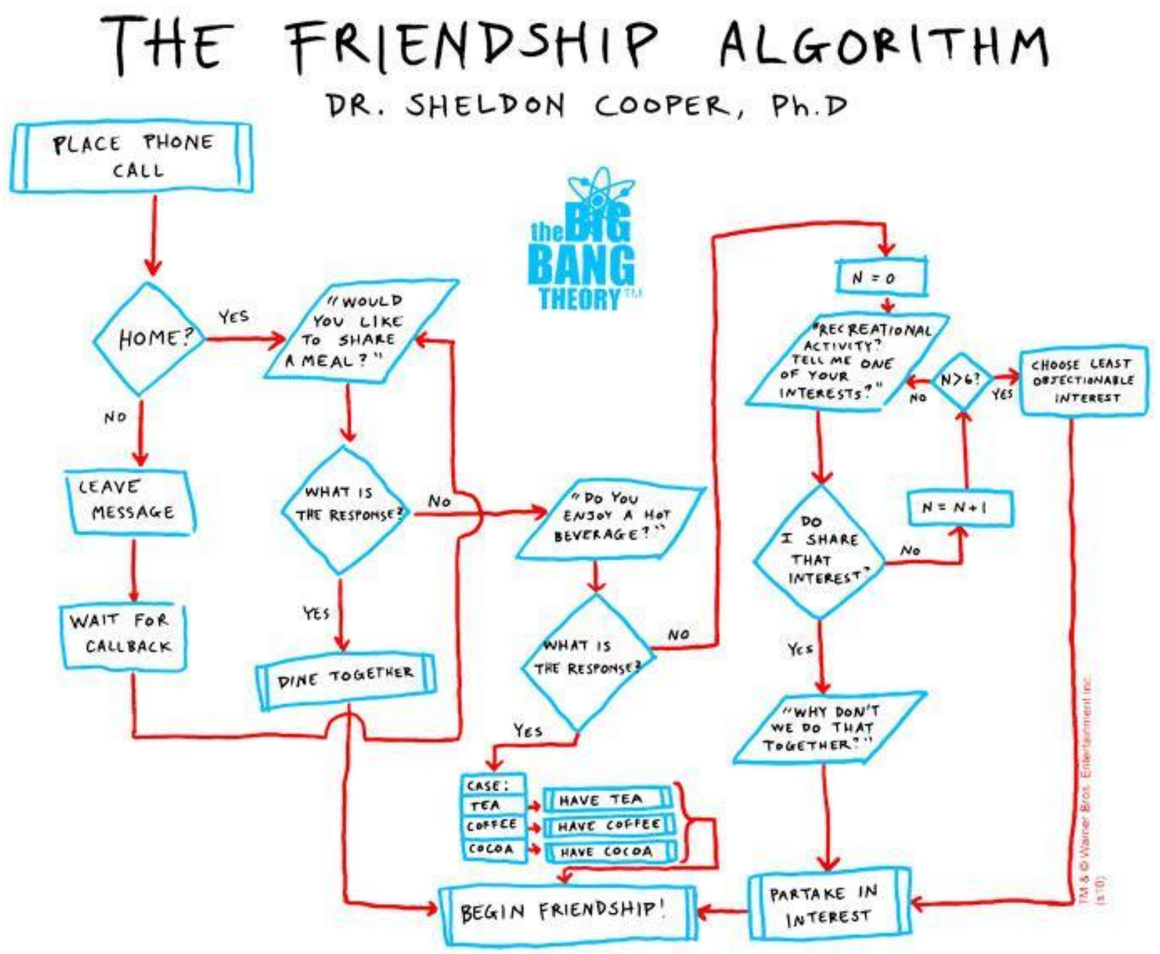


Source: http://bigbangtheory.wikia.com

What is an algorithm?

- An algorithm is a sequence of instructions that one must perform in order to solve a problem.

- Define problems in terms of inputs and outputs.



Source: http://bigbangtheory.wikia.com

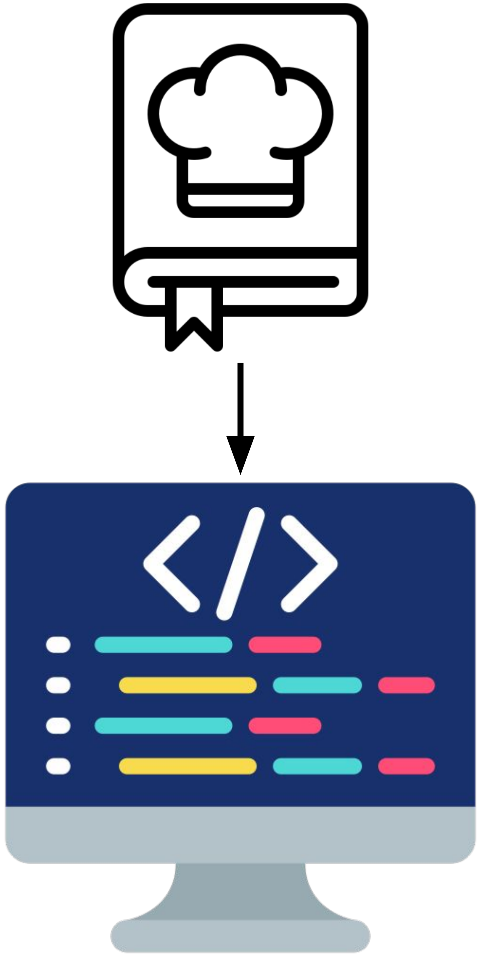
What is Pseudocode?

Real code is tied to syntax (which can be different between programming languages) → pseudocode strips that away.

Definition: human-readable individual computational steps that describe the logic of an algorithm, independent of programming language.

What is Pseudocode?

- Breaks big problems into computational manageable steps.
- Lets you organize your thoughts and plan before coding.
- Helps others understand (e.g., collaborators, publications, reproducibility).
- <u>Analogy: recipe before cooking.</u>



There are multiple pseudocode conventions….

- However, we merely want to break down our problem into **INDIVIDUAL** computational work steps to plan how to solve our problem.
- Systematically analyze how to solve the problem
- So you do not necessarily need to follow any convention.
- Basically think on paper if you are stuck with a task!

- **You will need to use one of the conventions if you publish an algorithm**

Think about known computational operations …

```
# Output Hello World to the terminal

print("Hello World!")

print("Hello Georgetown!")

print('Hello Everyone')
```

Think about known computational operations …

```
# Program input
cars = 100
people_per_car = 4
drivers = 30
passengers = 90

# Compute the dependent values
cars_not_driven = cars - drivers
cars_driven = drivers
carpool_capacity = cars_driven * people_per_car
average_people_per_car = ( drivers + passengers ) / cars_driven
people_in_last_car = ( drivers + passengers - 1 ) % people_per_car + 1
```

Think about known computational operations …

```
# DNA is cool!
dna_sequence = 'gcatgacgttattacgactctgtgtggcgtctgctggg'

# Compute dependent values
first_nucleotide = dna_sequence[0]
last_nucleotide = dna_sequence[-1]
first_four_nucs = dna_sequence[0:4]
last_ten_nucs = dna_sequence[-10:]
sequence_length = len(dna_sequence)

# Output results
print("First nucleotide",first_nucleotide)
print("Last nucleotide",last_nucleotide)
print("First four nucleotides",first_four_nucs)
print("Last ten nucleotides",last_ten_nucs)
print("Sequence length",sequence_length)
```

Think about known computational operations …

```
# DNA is cool!
dna_sequence = 'gcatgacgttattacgactctgtgtggcgtctgctggg'

# Compute dependent values
first_nucleotide = dna_sequence[0]
last_nucleotide = dna_sequence[-1]
first_four_nucs = dna_sequence[0:4]
last_ten_nucs = dna_sequence[-10:]
sequence_length = len(dna_sequence)

# Output results
print("First nucleotide",first_nucleotide)
print("Last nucleotide",last_nucleotide)
print("First four nucleotides",first_four_nucs)
print("Last ten nucleotides",last_ten_nucs)
print("Sequence length",sequence_length)
```

This is python syntax - do not use this in pseudocode but describe in a simple phrase what you want to do!

Think about known computational operations …

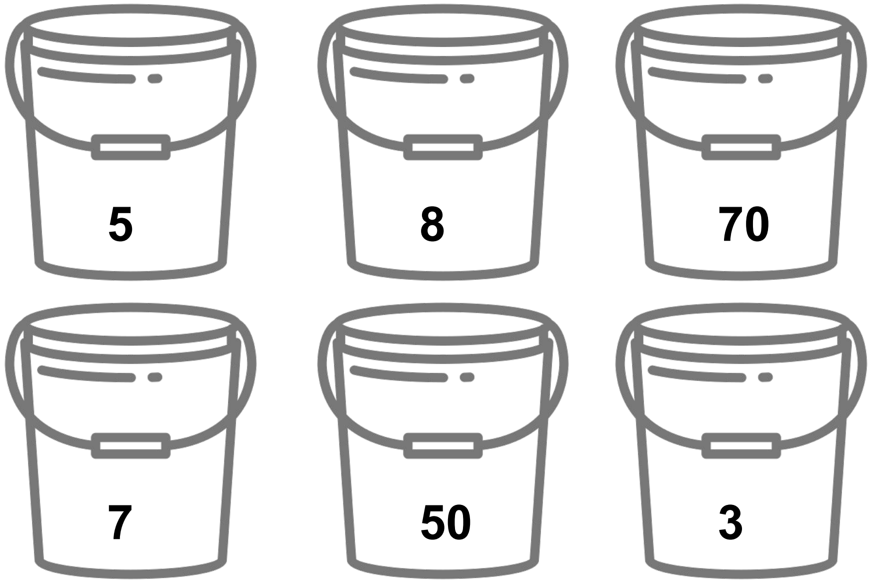
```
# DNA is cool!
dna_sequence = 'gcatgacgttattacgactctgtgtggcgtctgctggg'

# Compute dependent values
first_nucleotide = dna_sequence[0]
last_nucleotide = dna_sequence[-1]
first_four_nucs = dna_sequence[0:4]
last_ten_nucs = dna_sequence[-10:]
sequence_length = len(dna_sequence)

# Output results
print("First nucleotide",first_nucleotide)
print("Last nucleotide",last_nucleotide)
print("First four nucleotides",first_four_nucs)
print("Last ten nucleotides",last_ten_nucs)
print("Sequence length",sequence_length)
```

Strictly, this is more than one operation but it is a method that you know exists. ONLY use something like this if you know a method like this exists!

This is the level we have to think when planning our program!

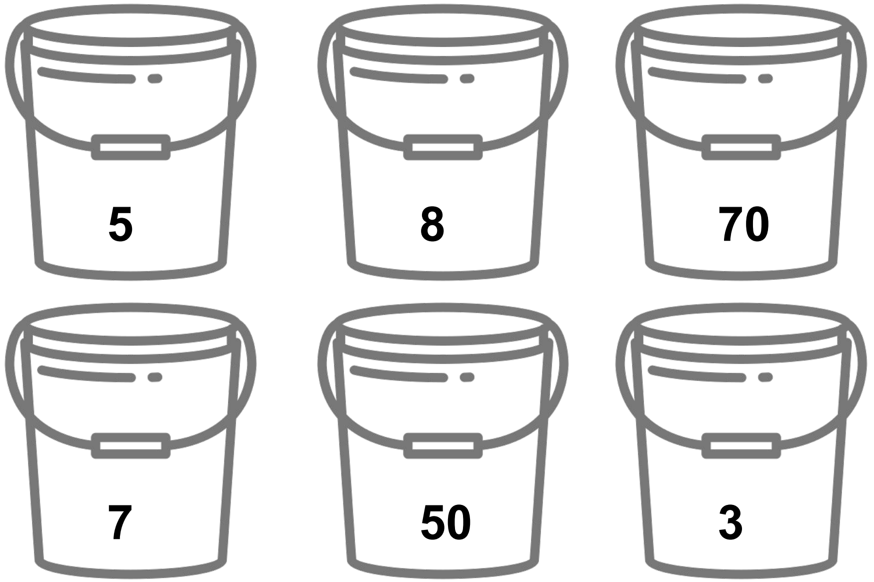
Example:
We want to find the maximum weight of those buckets

How would you code it?



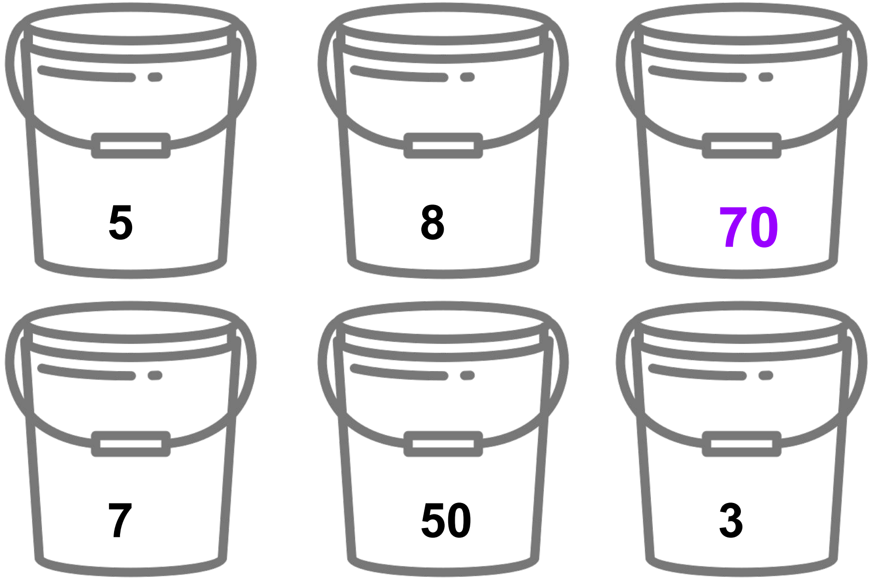
Example:
We want to find the maximum weight of those buckets

SET max to first number



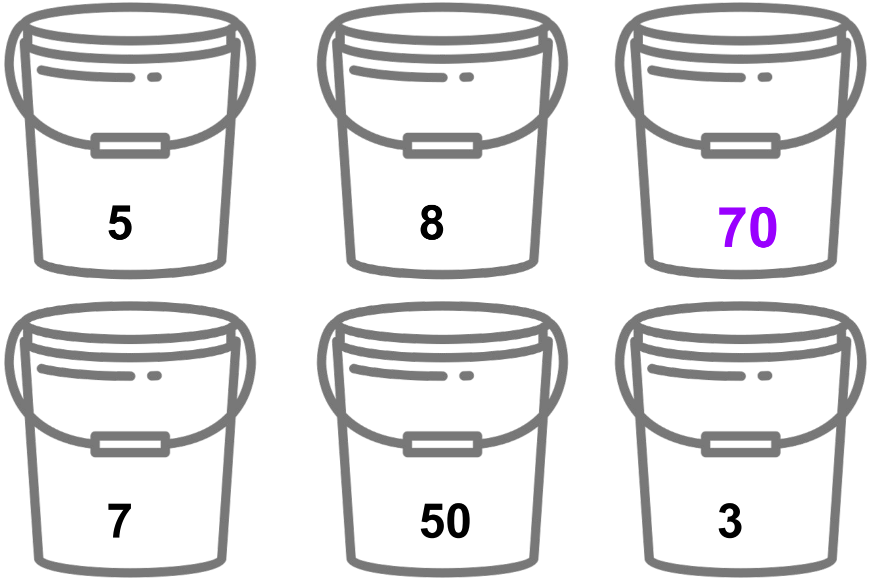
Example:
We want to find the maximum weight of those buckets

```
SET max to first number
FOR each number in the list of buckets
    IF number > max
        UPDATE max
RETURN max
```



Example:
We want to find the maximum weight of those buckets

```
SET max to first number
FOR each number in the list of buckets
    IF number > max
        UPDATE max
RETURN max
```



You can use your own words to describe something. **BUT: make sure it relates to ONE computational logical step!**

Now it is your turn to create such a Pseudocode

"Write a program that asks the user for a DNA sequence and finds the longest run of a single nucleotide (A, T, C, or G)."

Example: Input `AAATTTCCCAAAAA`, Output: `"A" with length 5`.

"Write a program that asks the user for a DNA sequence and finds the longest run of a single nucleotide (A, T, C, or G)."

Example: Input `AAATTTCCCAAAAA`, Output: `"A" with length 5`.

1. Write the example input and output on a piece of paper and draw how you would find it step by step. (2-3 minutes)
2. Try to formulate your steps in words (2-3 minutes)
3. Try to transform it into pseudocode (5 minutes)

UPLOAD A PHOTO OF YOUR DRAWINGS TO THESE SLIDES

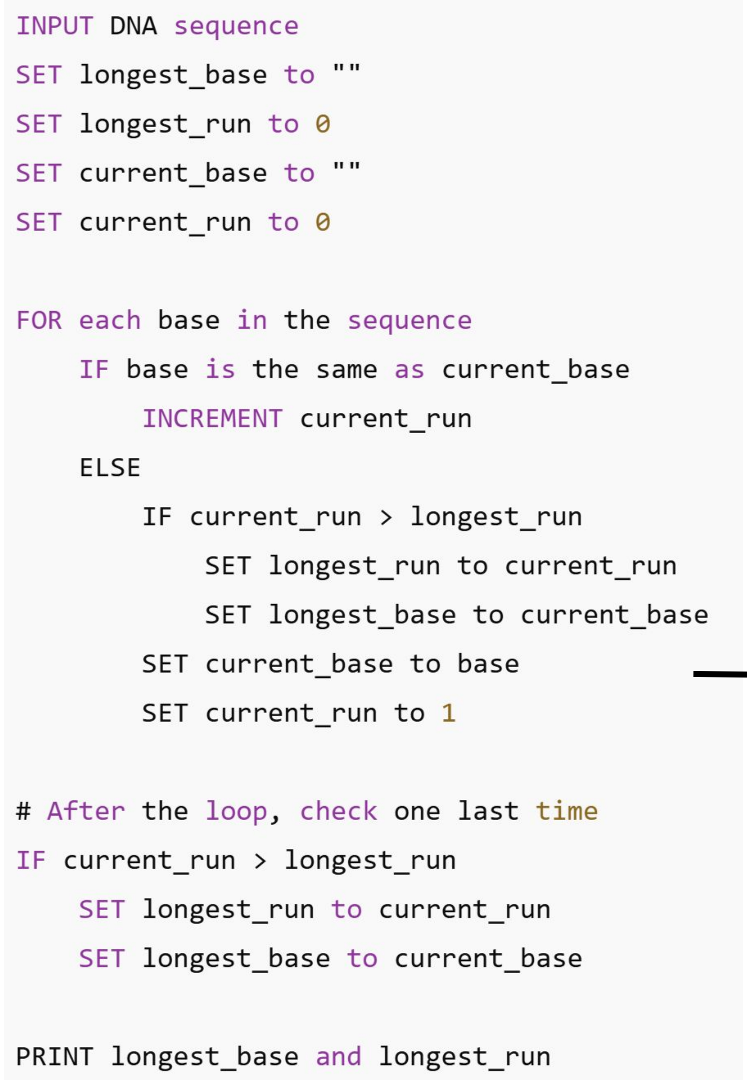
4. Could you now program it in python? – try it

https://tinyurl.com/366yv6yw

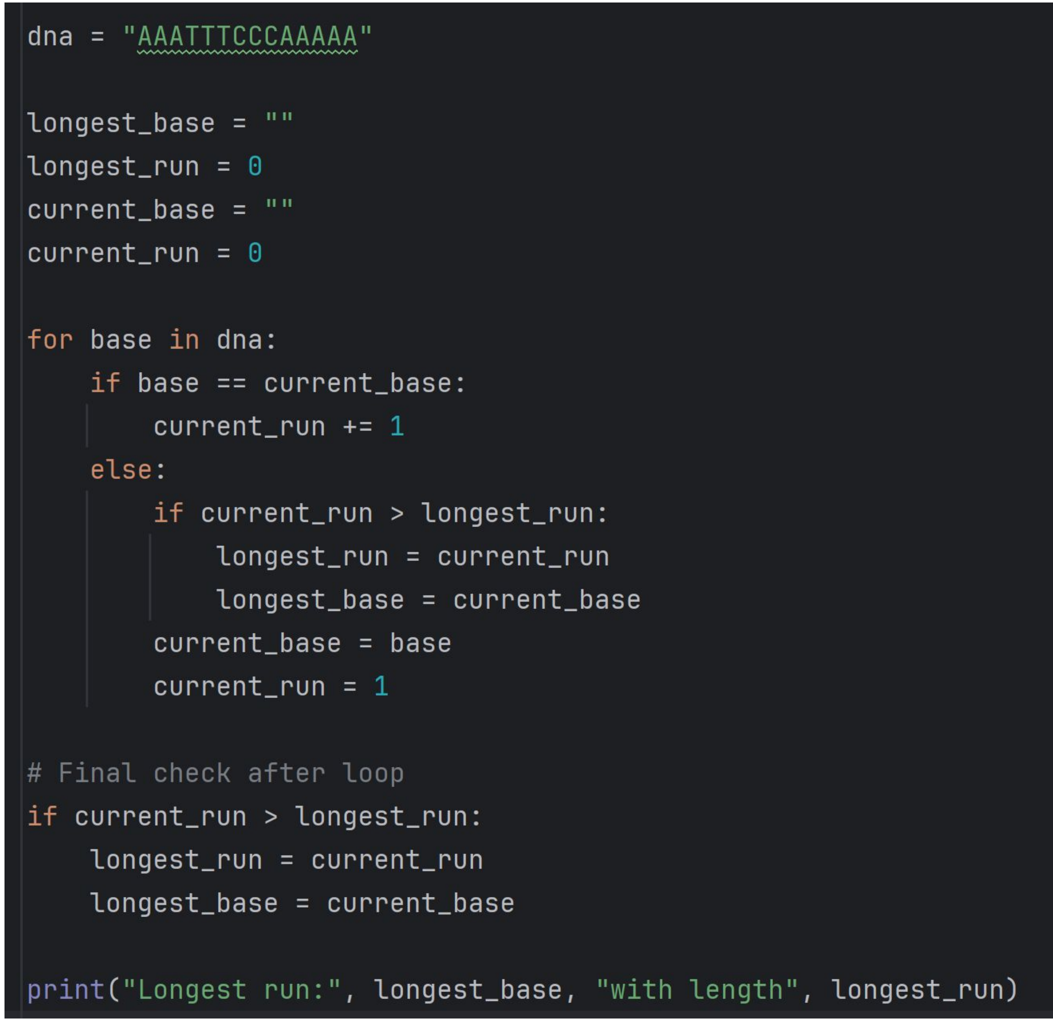
!!Do not go further than this slide until told so!!

One solution possibility

```
INPUT DNA sequence
SET longest_base to ""
SET longest_run to 0
SET current_base to ""
SET current_run to 0

FOR each base in the sequence
    IF base is the same as current_base
        INCREMENT current_run
    ELSE
        IF current_run > longest_run
            SET longest_run to current_run
            SET longest_base to current_base
        SET current_base to base
        SET current_run to 1

# After the loop, check one last time
IF current_run > longest_run
    SET longest_run to current_run
    SET longest_base to current_base

PRINT longest_base and longest_run
```

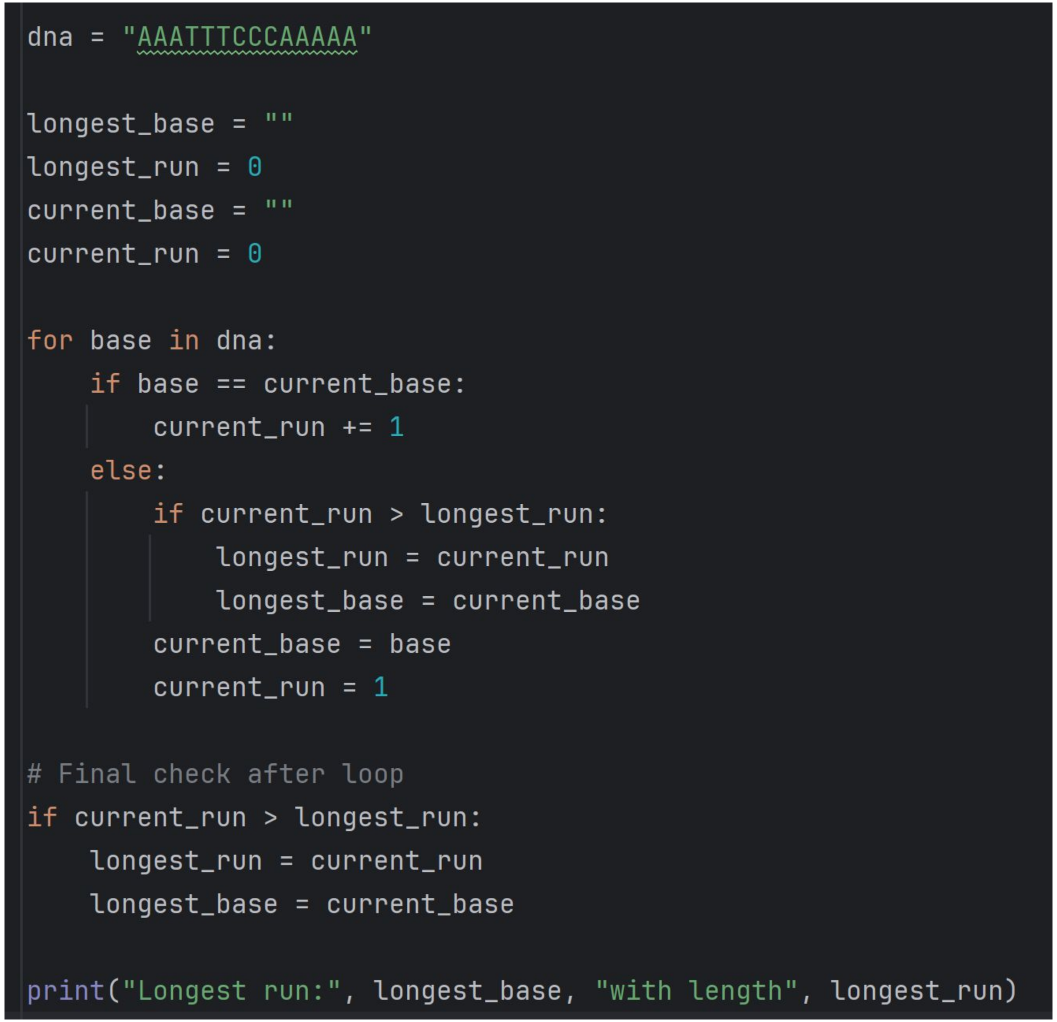


```
INPUT DNA sequence
SET longest_base to ""
SET longest_run to 0
SET current_base to ""
SET current_run to 0

FOR each base in the sequence
    IF base is the same as current_base
        INCREMENT current_run
    ELSE
        IF current_run > longest_run
            SET longest_run to current_run
            SET longest_base to current_base
        SET current_base to base
        SET current_run to 1

# After the loop, check one last time
IF current_run > longest_run
    SET longest_run to current_run
    SET longest_base to current_base

PRINT longest_base and longest_run
```



```python
dna = "AAATTTCCCAAAAA"

longest_base = ""
longest_run = 0
current_base = ""
current_run = 0

for base in dna:
    if base == current_base:
        current_run += 1
    else:
        if current_run > longest_run:
            longest_run = current_run
            longest_base = current_base
        current_base = base
        current_run = 1

# Final check after loop
if current_run > longest_run:
    longest_run = current_run
    longest_base = current_base

print("Longest run:", longest_base, "with length", longest_run)
```

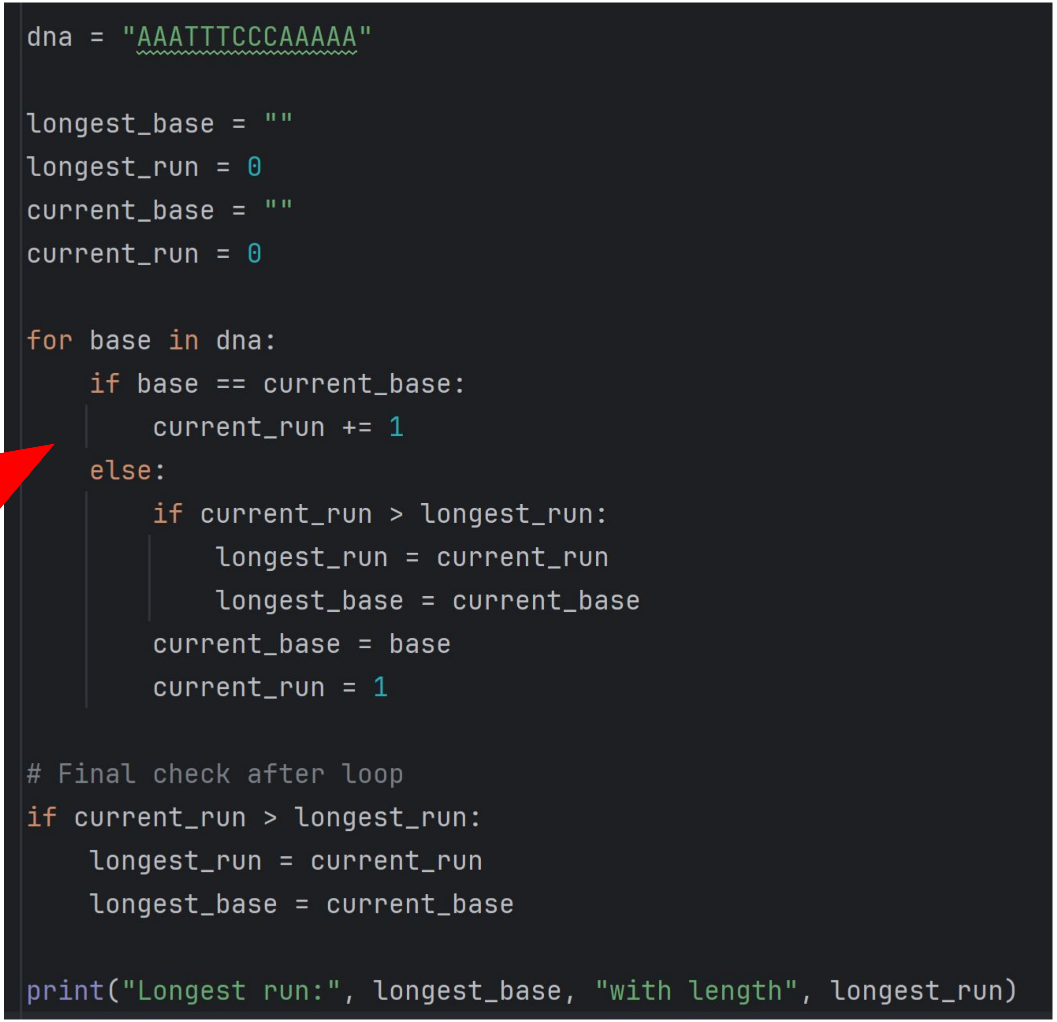
Great… but what if something does not work?



```python
dna = "AAATTTCCCAAAAA"

longest_base = ""
longest_run = 0
current_base = ""
current_run = 0

for base in dna:
    if base == current_base:
        current_run += 1
    else:
        if current_run > longest_run:
            longest_run = current_run
            longest_base = current_base
        current_base = base
        current_run = 1

# Final check after loop
if current_run > longest_run:
    longest_run = current_run
    longest_base = current_base

print("Longest run:", longest_base, "with length", longest_run)
```

Great… but what if something does not work?

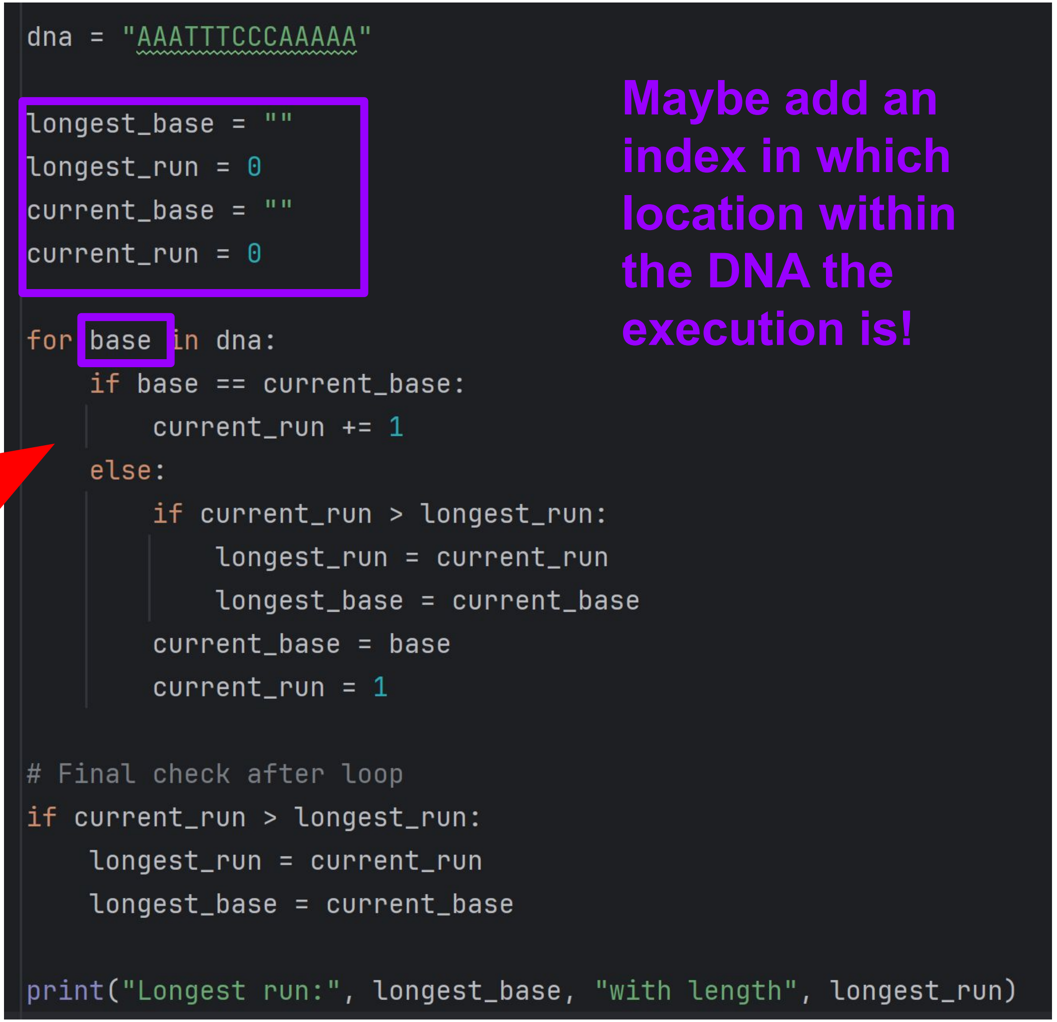
Add **print** statements to the loop **with all the variables** so you can see **how they change if they change like they should!**



```python
dna = "AAATTTCCCAAAAA"

longest_base = ""
longest_run = 0
current_base = ""
current_run = 0

for base in dna:
    if base == current_base:
        current_run += 1
    else:
        if current_run > longest_run:
            longest_run = current_run
            longest_base = current_base
        current_base = base
        current_run = 1

# Final check after loop
if current_run > longest_run:
    longest_run = current_run
    longest_base = current_base

print("Longest run:", longest_base, "with length", longest_run)
```

Great… but what if something does not work?

Add **print** statements to the loop **with all the variables** so you can see **how they change if they change like they should!**



```
dna = "AAATTTCCCAAAAA"

longest_base = ""
longest_run = 0
current_base = ""
current_run = 0

for base in dna:
    if base == current_base:
        current_run += 1
    else:
        if current_run > longest_run:
            longest_run = current_run
            longest_base = current_base
        current_base = base
        current_run = 1

# Final check after loop
if current_run > longest_run:
    longest_run = current_run
    longest_base = current_base

print("Longest run:", longest_base, "with length", longest_run)
```

Maybe add an index in which location within the DNA the execution is!

Use pseudocode/flow charts to make a plan for a problem that you do not understand how to teach the computer!