pseudocode / algorithm : step-by-step instructions to solve a problem

CONTROL FLOW :

General case: in-order execution ; exceptions → ① CONDITIONALS this or that → ② LOOPS repetition → ③ FUNCTIONS mini programs

execute each step once in provided order

Motivation: perhaps the two most important concepts for a programmer to understand are **control flow** and **state**. One challenge when learning programming is that details about the particular language (in this case, Python) can distract from these two core concepts. In this worksheet, we'll explore control flow and state using pseudocode. Pseudocode is fake code. It's similar to real code, but a computer wouldn't know how to run it. The advantage of writing pseudocode is that it's easier for humans to think about.

Directions: for each problem, we'll have some state, represented by one or more boxes. Each box will have a value inside, and a name to the left. Each problem will also have some code. The code is just a numbered list of instructions written in English. Some instructions might tell you to change the value in a box. When that happens, cross out the previous value in the box, then write the new value there.

1

State:

VARIABLE X | ~~10~~ ~~11~~ ~~13~~ 26

~~10~~ ~~11~~ ~~22~~ 24

Code:

```
→ 1. Add 1 to the "X" box (the box should then look like: ~~10~~ 11)
→ 2. Add 2 to the "X" box
→ 3. Double the value in the "X" box
```

Questions:

- What was the final value in the "X" box?
- Say your friend got a different answer, 24, because they don't like doing things in order. What's a good rule to make sure everybody computes the same? SWAP STEPS 2 & 3 (SEE ABOVE)

2

State:

in | 4   out | ~~0~~ ~~1~~ ~~4~~ ~~16~~ 64

Code:

```
1. Put 1 in the "out" box
2. Multiply the value in the "out" box by the value in the "in" box *
3. Multiply the value in the "out" box by the value in the "in" box
4. Multiply the value in the "out" box by the value in the "in" box
```

Questions:

- What was the final value in the "out" box?
- Mathematically speaking, what was this computation doing? "in power 3"

3

State:

first: | Ada   last: | Lovelace

msg: | Hello~~AdaLovelace~~ AdaLovelace

STRING DATA TYPE

Code:

```
1. Add the value in "first" to the value in "msg"
2. Add the value in "last" to the value in "msg"
```

Questions:

- How is "Add" here different than "Add" in example 1? → CONCATENATION (instead of add)
- What additional instructions would make msg more readable? ↳ Add space between Ada Lovelace

* How can you convert solution to "in power 10"? Repeat step #2 10 times

* OPPOSITE TERMINOLOGY : INCREMENT

ABSOLUTE OF A NUMBER (X)

4

State:

X ~~-4~~ 4   abs ~~0~~ 4

CONDITIONAL

Code:

1. If the value in “X” is negative, ✓ continue to step 2, ✓ otherwise ✗ skip to step 3 SOMETIMES WE SKIP STEPS
2. Multiply the value in “X” by -1, and put the result back in “X”
3. Copy the value in “X” to the “abs” box

Questions:

- What was “abs” at the end? SAME ANSWER
- What if 4 had been in “X” instead at the beginning?
- What is the code trying to do?

5

State:

age 150   msg ~~success~~ too old

CONDITIONAL

Code:

1. If the value in “age” is less than 0, ✗ continue to step 2, ✗ otherwise skip to step 3 ✓
2. Put “too young” in the “msg” box
3. If the value in “age” is more than 125, ✓ continue to step 4, ✓ otherwise skip step 4 and finish ✗
4. Put “too old” in the “msg” box

Questions:

- What was msg at the end?
- What is the code trying to do? BOUNDS CHECKING

DATA VALIDATION /

DATA CLEANUP

LOOP 6

State:

N ~~4~~ ~~3~~ ~~2~~ 1

total ~~0~~ ~~1~~ ~~4~~ ~~12~~ 24   answer ~~0~~ 24

Code:

1. Put 1 in the “total” box False False False ...ue
2. If “N” equals 1, skip to step 6, otherwise continue to step 3
3. Multiply the value in “total” by the value in “N”, and put the result back in “total”
4. Decrease the value in “N” by 1 DECREMENT*
5. Go to step 2
6. Copy the value in total to the answer box

Questions:

- What is the value in answer?
- What would have happened if N started at -1?
- What is the code meant to do?
- Finish listing the order in which you performed the steps in the code:

1, 2, 3, 4, 5, 2, 3, ... 4, 5, 2, 3, 4, 5, 2, 6

INFINITE LOOP (NEVER ENDS)

FACTORIAL OF N

$N! = N \times (N-1) \times (N-2) \times \ldots \times 1$

PARAMETER

ARGUMENT

Arguments go into parameters

State:

7

moves: ~~0~~ ~~2~~ ~~1~~ ~~0~~ ~~3~~ ~~2~~ ~~1~~ 0   result: ♡

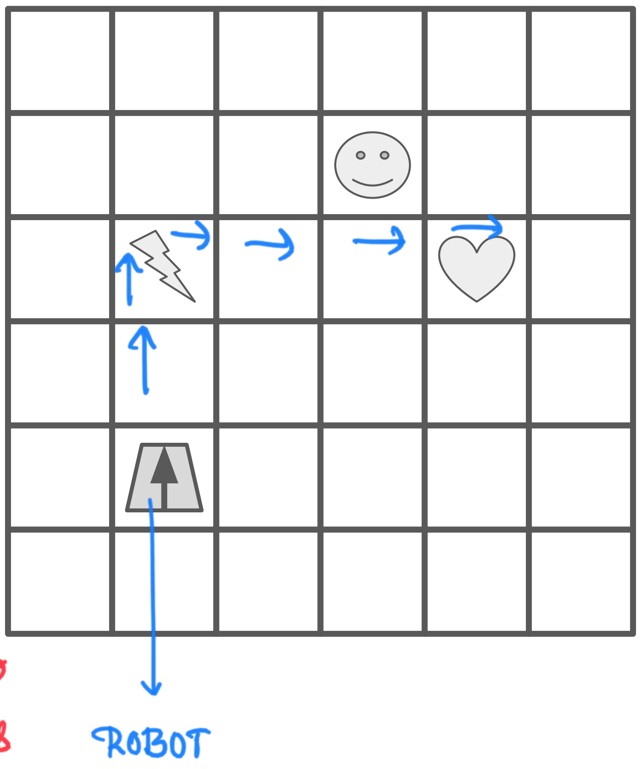
→ CALLING / INVOKING A FUNCTION (Move Code)

Main Code:
1. Put 2 in the "moves" box
2. Perform the steps under "Move Code", then continue to step 3 *
3. Rotate the robot 90 degrees to the right (so arrow points to right)
4. Put 3 in the "moves" box
5. Perform the steps under "Move Code", then continue to step 6
6. Whatever symbol the robot is sitting on, write that symbol in the "resut" box

Move Code:*

A. If "moves" is 0, stop performing these steps in "Move Code", and go back to where you last were in "Main Code" to complete more steps
B. Move the robot forward one square, in the direction the arrow is pointing
C. Decrease the value in "moves" by one
D. Go back to step A

LOOP



Questions:
- What symbol did you write in the "result" box?
- Could you have written a different set of steps under the "Main Code" section that did not require you to ever perform the steps in the "Move Code" section?
- How would you change "Main Code" if you didn't want to ever let the robot touch the lightning bolt?

→ Instead of calling Move Code in steps 2 & 5, we could have pasted the Move Code Steps
→ REDUNDANT CODE IS A RED FLAG (DON'T DO IT!)

→ ① Use conditionals (OR)
② Don't use 2 as argument in step 1.

8

State:

LISTS

items	9	1	3	5	7
sorted	1	3	5	7	9

middle

* OPTIMIZATION

output	5

LOOP

Code:

1. If there are no empty boxes next to "sorted", skip to step 4, otherwise continue to line 2
2. Find the smallest number in "items" that hasn't been crossed off (the first time you do this step, note that nothing will have been crossed off). Write that smallest number in the first empty box by "sorted" (first is the most left). Cross off that smallest number in "items" so that you don't use it again later
3. Go back to step 1
4. Find the number in the middle of sorted, and write it in the "output" box

Questions:

- What number is in "output"?
- What is the code trying to do? MEDIAN OF NUMBERS
- Is there any way to skip some of the steps and still get the same answer in output?

USES SORTING

* You could terminate the loop as soon as you find median (skip sorting 7 & 9)

9

State: DICTIONARY : mapping

player1	alice
player2	bob
score1	~~0~~ 175
score2	~~0~~ 199

scores

key $\longrightarrow$	value
alice	175
bob	199
cathy	213
david	180

winner	~~tie~~ bob

Code:

1. Find the number in "scores" next to the name that also appears in the "player1" box, then write that number in the "score1" box
2. Find the number in "scores" next to the name that also appears in the "player2" box, then write that number in the "score2" box
3. Put "tie" in "winner" box
4. If the value in "score1" is greater than the value in "score2", continue to step 5, otherwise skip to step 6 ✓
5. Copy the value in the "player1" box to the "winner" box ✓
6. If the value in "score2" is greater than the value in "score1", continue to step 7. Otherwise, skip step 7 and just finish
7. Copy the value in "player2" to the "winner" box

Questions:

- What does the winner box say?
- What is the code trying to do?
- What would happen if the "player1" and "player2" boxes both initially said "cathy"? results in a "tie"

Use a mapping (key → value) to determine winner