



Introduction to Information Systems

- Understanding the digital world

5 Algorithms, Programming and Programming
languages

Liang Zhao

ILA, Doshisha University

12001102, Fall, 2024





Today's schedule

- **Answers to the coding tasks (15')**
- **Mini test #5 (10')**
- **Algorithms & Programming (20')**
- **Programming languages (45')**

Last coding task 1

This program calculates $1 + 2 + \dots + N$ for a number **N given by the user**.

Toy Machine Simulator ->

<http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html>



1. **_GET**
2. **_STORE N**
3. **L_LOAD N**
4. **_IFZERO E**
5. **_ADD S**
6. **_STORE S**
7. **_LOAD N**
8. **_SUB 1**
9. **_STORE N**
10. **_GOTO L**
11. **E_LOAD S**
12. **_PRINT**
13. **_STOP**
14. **N**
15. **S 0**

Last coding task 2

Write a program that prints 1, 2, ..., 99 **in that order** with loop: a smart answer by students.

Toy Machine Simulator ->

<http://www.cs.princeton.edu/courses/archive/fall18/cos109/toysim.html>



1. L_load a
2. _add 1
3. _store a
4. _print
5. _sub 99
6. _ifzero b
7. _goto L
8. b stop
9. a 0

Algorithm is the abstract procedure to solve a task

The efficiency of a program depends on the algorithm.

Only 17 operations

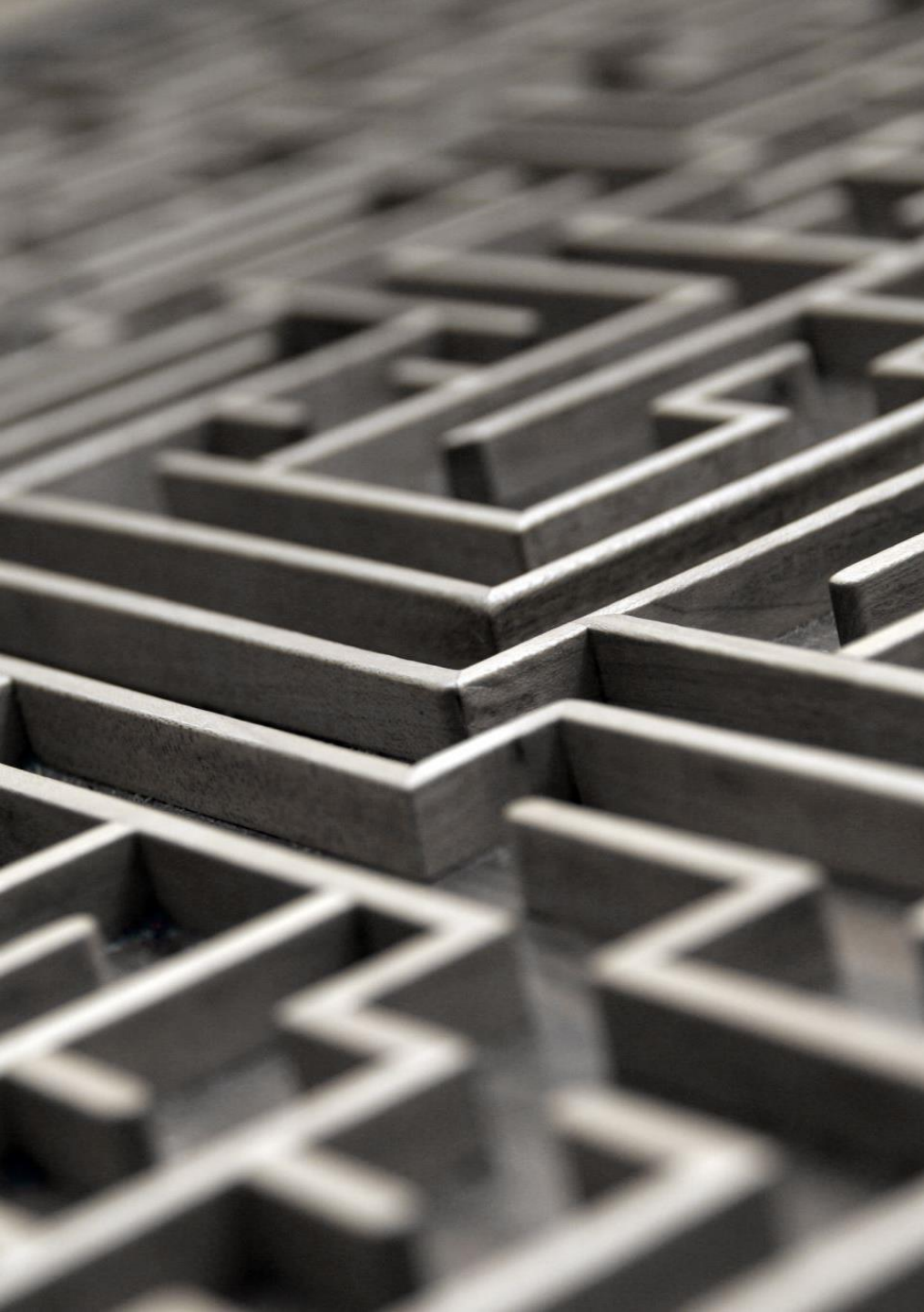
```
  _GET  
  _STORE A  
  _ADD A  
  {  
  _...  
  _ADD A  
  }  
  _PRINT  
  _STOP  
  A
```

99 times

> 100 operations

```
1.  _GET      9.  STORE S  
2.  _STORE A 10.  _GOTO L  
3.  L_LOAD N 11.  E_LOAD S  
4.  _IFZERO E 12.  _PRINT  
5.  _SUB 1   13.  _STOP  
6.  _STORE N 14.  A  
7.  _LOAD S  15.  S_0  
8.  _ADD A   16.  N_100
```

```
1.  _GET      10.  _ADD A  
2.  _STORE A  11.  _STORE A  
3.  _ADD A     12.  _ADD A  
4.  _STORE A  13.  _ADD A  
5.  _ADD A     14.  _ADD A  
6.  _STORE A  15.  _ADD A  
7.  _ADD A     16.  _PRINT  
8.  _ADD A     17.  _STOP  
9.  _ADD A     18.  A
```



Hard problems and complexity

- **Complexity** is how difficult an algorithm is. It is evaluated by the **#steps** and **#memory** used.
- <https://www.youtube.com/watch?v=Q4gTV4rDzRs>
(See HOW HARD a hard problem can be. It is in Japanese with English subtitles. About 8 minutes)

```
object to mirror_
mirror_mod.mirror_object
operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
mirror_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob
mirror_ob.select = 0
= bpy.context.selected_object
data.objects[one.name].select
print("please select exactly
```

Program

- A collection of instructions to **implement** an algorithm.
- There exist many dialects, called **programming languages**.
- The one we learned is **assembly**. There are > 200 languages.

Three fundamental structures of ALL programs (algorithms).

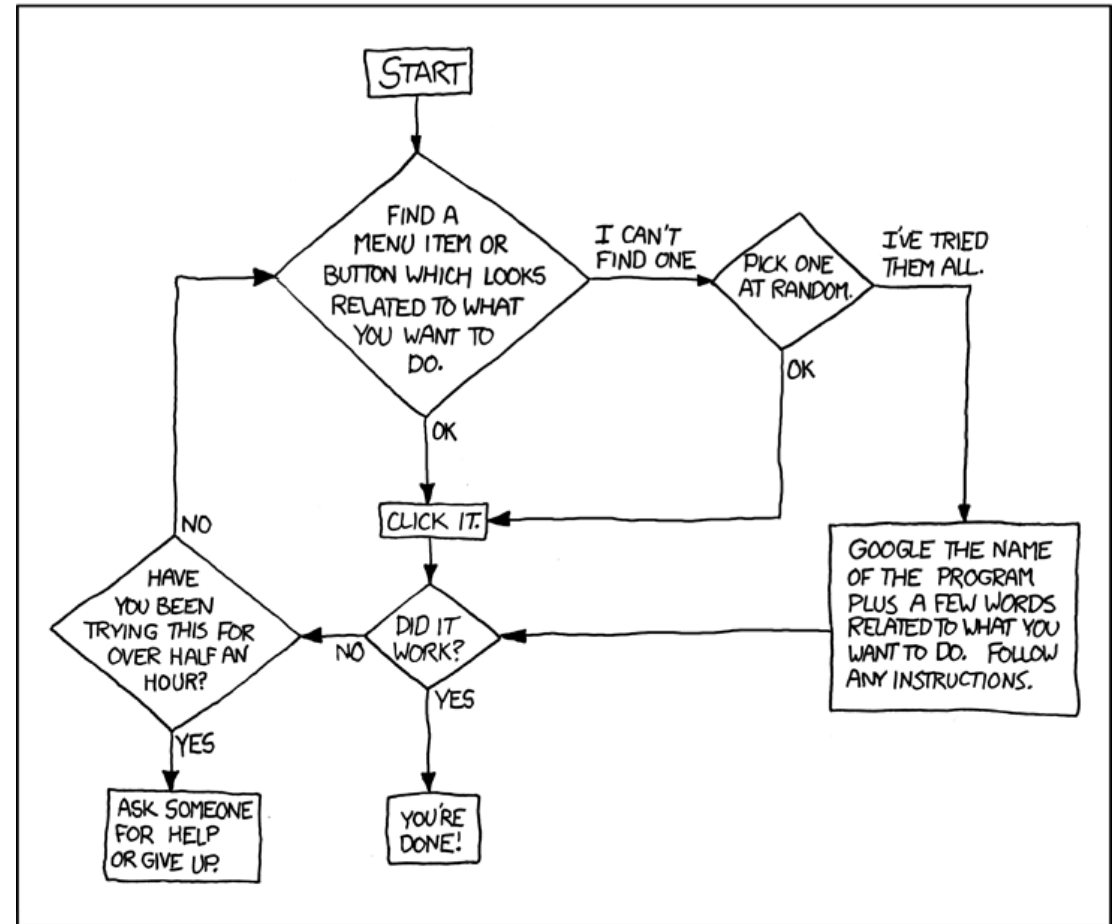
1. Sequential

2. Conditional

3. Iterative (-> loop)

DEAR VARIOUS PARENTS, GRANDPARENTS, CO-WORKERS,
AND OTHER "NOT COMPUTER PEOPLE."

WE DON'T MAGICALLY KNOW HOW TO DO EVERYTHING IN EVERY
PROGRAM. WHEN WE HELP YOU, WE'RE USUALLY JUST DOING THIS:



PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN.
CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

Python: a high-level programming language



1. A Python program to calculate $1 + 2 + \dots + N$ for a number N input by the user.
2. A Python program to print 1, 2, ..., 99 in that order with loop.

<https://paiza.io/en/projects/new?language=python3>

Note: use 4 spaces for the indent.

Program 1

1. `n = int(input())`
2. `s = 0`
3. `for i in range(1, n+1):`
4. `s = s + i`
5. `print(s)`

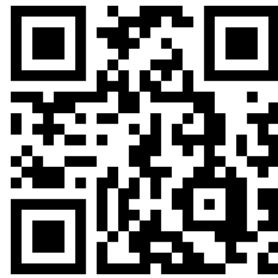
Program 2

1. `for i in range(100):`
2. `print(i)`

1. Put the code into the code window (skip # lines).
2. Put the input in the "Input" window.
3. Click the "Run (Ctrl+Enter)" button to execute.

Scratch (yet another programming language)

1. Go to <https://scratch.mit.edu>
2. "Start Creating" -> "Tutorials"
3. Open the next sample (click flag -> start, click cat -> stop)

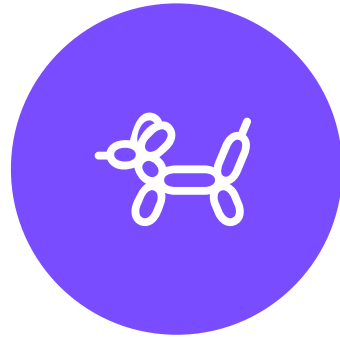


<https://scratch.mit.edu/projects/750318208/>



Click "See inside" to investigate the code. Modify the program so that the cat runs faster, or slower, or says something different, or does something different, etc. Notice how the three fundamental structures are implemented in Scratch.

Homework



WATCH VIDEOS

Watch the movies mentioned so far if you have not
(You are not expected to understand everything)



READ THE TEXTBOOK

IF YOU HAVE NOT