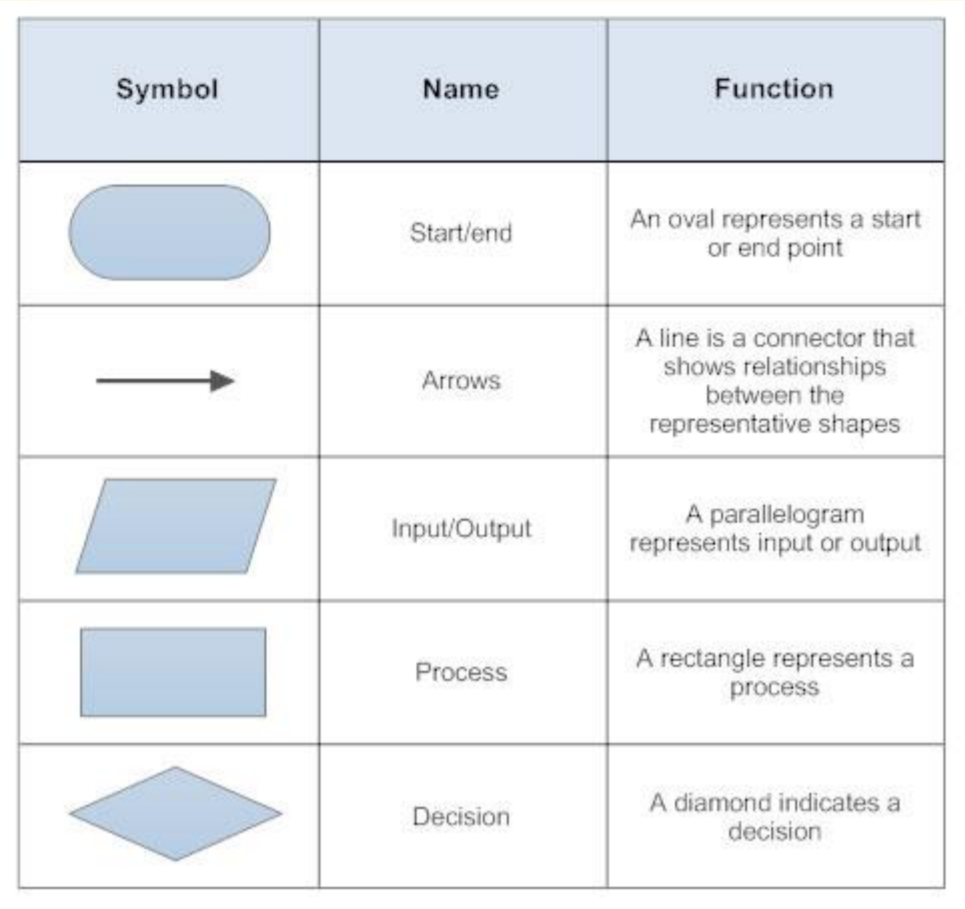
COMPUTATIONAL THINKING

IDENTIFYING INPUTS AND OUTPUTS

When given a problem it is best to decide what outputs are required from the system. You can then work backwards to decide the inputs needed by the program.

We can diagrammatically show the flow of data through a system, identifying inputs and outputs using a flow diagram.



Symbol	Name	Function
⬭	Start/end	An oval represents a start or end point
→	Arrows	A line is a connector that shows relationships between the representative shapes
▱	Input/Output	A parallelogram represents input or output
▭	Process	A rectangle represents a process
◇	Decision	A diamond indicates a decision

Using a simple ATM Machine:

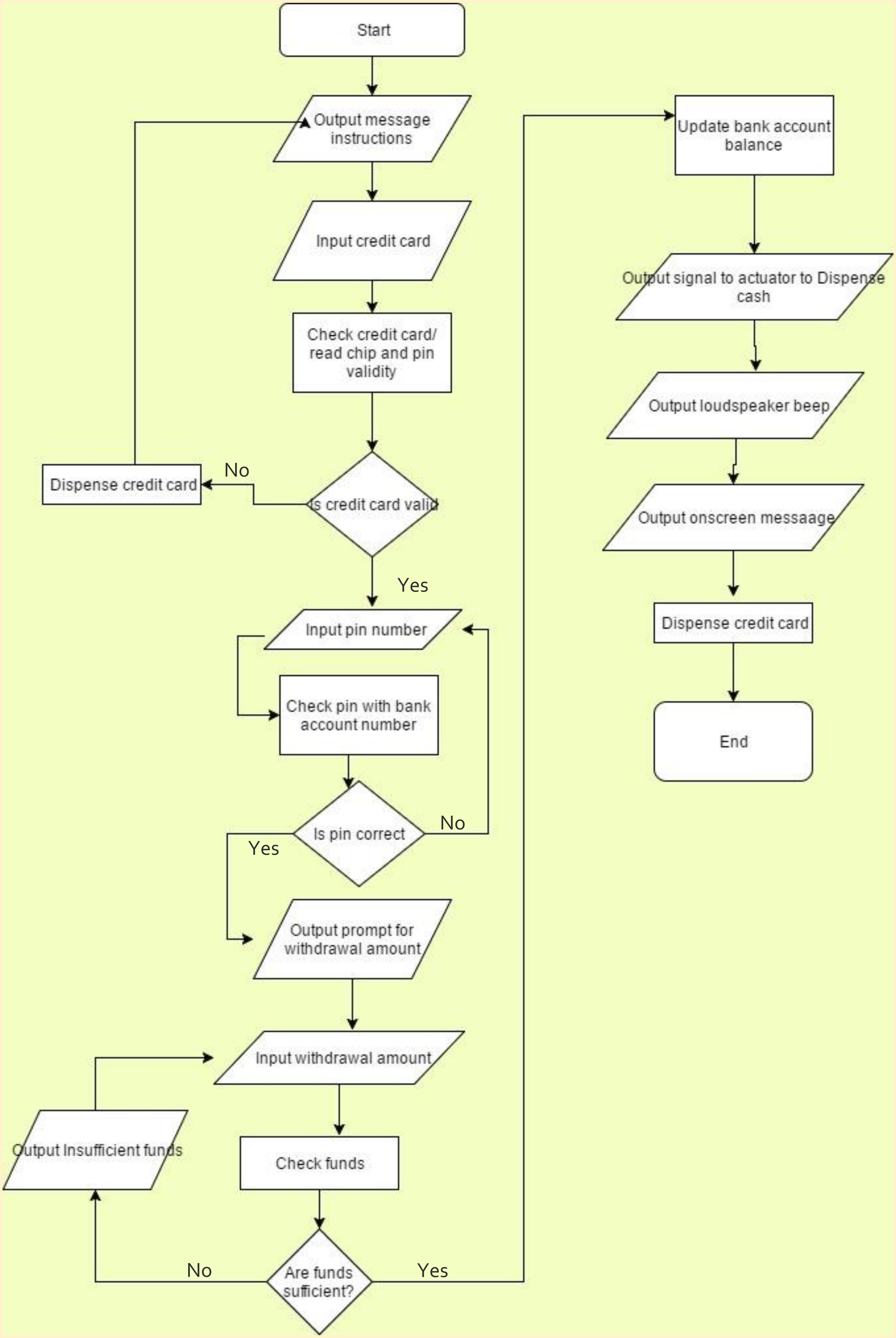
Outputs

1. Actuator/motor to release cash
2. Speaker to emit a confirmation noise on dispense
3. Screen to output data of balance and options (graphical menu interface)
4. Printer to print receipt

Inputs

1. Card reader (chip and pin) to read card
2. Keypad to input pin number
3. Touchscreen to select options on the GUI

Practicing flowcharts


Start
Output message instructions
Input credit card
Check credit card/ read chip and pin validity
Is credit card valid
No
Dispense credit card
Yes
Input pin number
Check pin with bank account number
Is pin correct
No
Yes
Output prompt for withdrawal amount
Input withdrawal amount
Output Insufficient funds
Check funds
Are funds sufficient?
No
Yes
Update bank account balance
Output signal to actuator to Dispense cash
Output loudspeaker beep
Output onscreen messaage
Dispense credit card
End

Preconditions:

Before devising a solution to a problem it is important to consider what resources are already in place.

For example, when planning an upgrade on an IT suite, rather than simply going ahead and looking and new computer specifications you can consider what is required: are new keyboards, mouses and other peripherals required?

If a new CPU is required, do we also have to factor in the cost of a new motherboard to support the new socket. Maybe the Pu only works with newer forms of RAM.

The need for reusable program components:

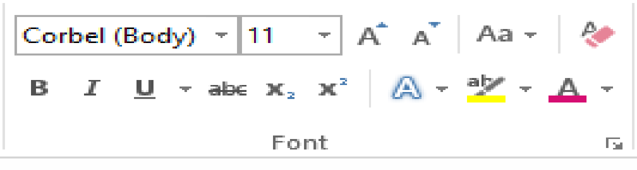
In the world of software development, code reuse has very beneficial. Code should be designed to have components that are largely reusable in future projects.

- It saves time when programming solutions
- It can save on resources and money
- It can reduce redundancy

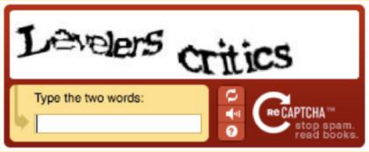
This is all achieved by taking advantage of sections of code that are already created and tested by a previous developer. It should be considered what part of a program may be reusable, specific code to the program is not intended for reuse.

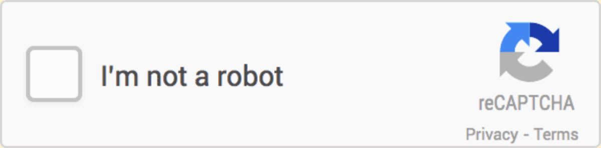
Modules such as the random function in python can be easily reused in future programs with the code "Import random" that will import the whole module in a single line so features of that module can be used.

Other examples are in common software packages by Microsoft. The font section found in word has been designed by coders and reused in other software such as excel, PowerPoint, publisher……



Another good example of external reuse is the captchas found on registration forms online that prevent computers from auto filling account details. The algorithms are reused across sites globally rather than being coded again each time.


Levelers critics
Type the two words:
reCAPTCHA™
stop spam.
read books.


□ I'm not a robot
reCAPTCHA
Privacy - Terms