Year Nine Programming

Data types, structures, string handling and iteration

Data types: Data used by programming languages must be of a certain type. This means what kind of data it is. *Boolean* data is `True` or `False`. *Integers* are whole numbers. *Float* (or real) are any numbers. *Strings* can contain any characters. Data input is a string by default and must be converted before it can be used like another data type.

An **array** is like a variable, but can have more than one item. Think if it as a list. The first item ("Melissa") is item 0. "Liam" is item 5.

Iteration: repeat sections of code as long as a condition is met.

len(array) will give you the number of items in an array.

String handling: Manipulation of string variables:

name = 'John'

name.upper()

>>> 'JOHN'

name[0]

>>> 'J'

Name[0:2].lower()

>>> 'jo'

```
from random import randint

# array data structure
pupils = ["Melissa","Gabe","Corey","Alanah","Darcy","Liam"]

# Boolean variable
var_again = True

print("Welcome to the random pupil picker")
input("Press enter to get a random name")

# iteration - repeat the indented code as long as var_again is 'y'
while var_again == True:

    # get a random number between 0 and the number of items in the array - 1
    random_number = randint(0,len(pupils)-1)

    # output the corresponding name
    print("Selected student: " + pupils[random_number])

    print("Pick another pupil? Enter 'y'")
    ans = input()

    # convert answer to lower case
    ans = ans.lower()

    # selection statement(if/else). If the user doesn't enter 'y'...
    if ans.lower() != "y":
        var_again = False

# After the loop
print("Thanks for using the random pupil picker")
```

Boolean variables can be True or False

You can refer to an item in an array by typing the name of the item, followed by it's position in square brackets. For example, pupils[0] is 'Melissa'

pupils array

Position	0	1	2	3	4	5
Data	'Melissa'	'Gabe'	'Corey'	'Alanah'	'Darcy'	'Liam'

Computer Science Basics

Binary Number System

Converting **binary to denary** numbers. Example: Convert 01110101 to denary.

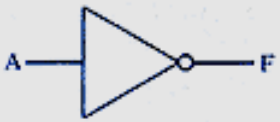
1. Write the place values above each bit (1 or 0).

Place value	128	64	32	16	8	4	2	1
	0	1	1	1	0	1	0	1

2. Add up the place values where there is a 1 below it.

64 + 32 + 16 + 4 + 1 = **117**

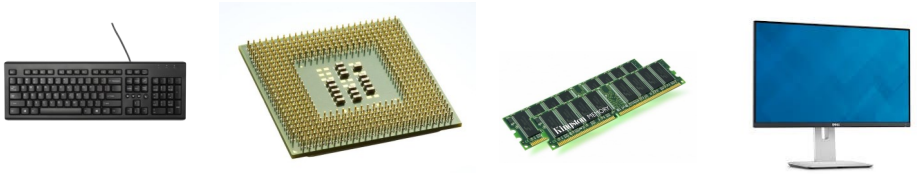
Logic Gates

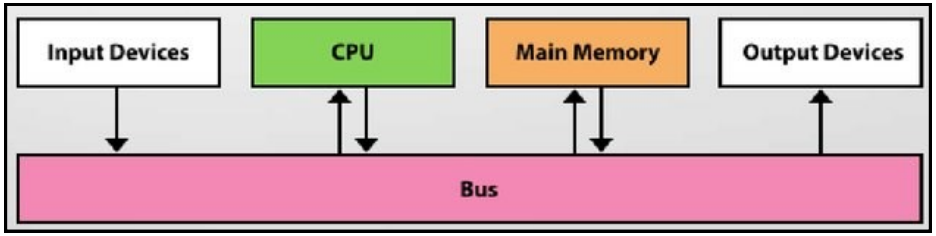
Name	Graphic Symbol	Algebraic Function	Truth Table
AND		F = A • B or F = AB	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		F = A + B	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		F = $\overline{A}$ or F = A′	A | F 0 | 1 1 | 0

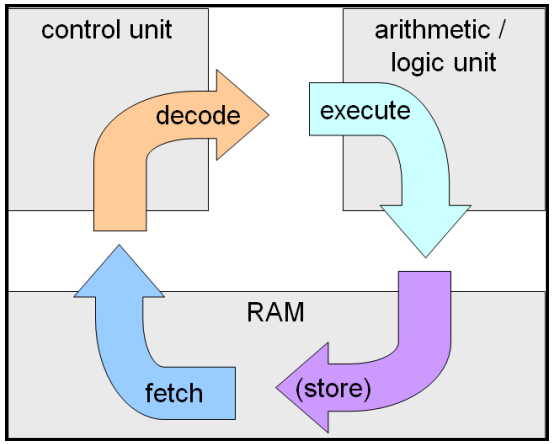
von Neumann Architecture

In 1945, John von Neumann proposed a design for computer systems which is still used in most computers today.

Computer programs (instructions) and data are stored in **memory**. Instructions are **fetched** in sequence by the **central processing unit** (CPU). They are then executed **one-at-a-time**. Instructions and data travel between components on wires called a **bus**.







Above: data and instructions move between computer components on buses.

Left: Instructions are fetched in sequence from memory (RAM). They are then carried out (executed) by the CPU. Data (your work) may be stored in RAM.