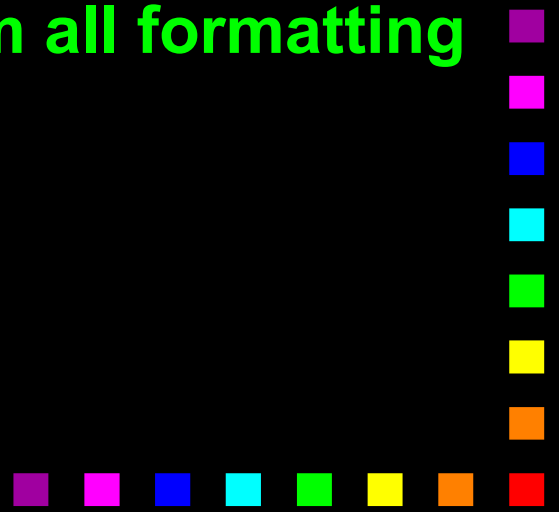
COBOL Basics 1

COBOL coding rules

- Almost all COBOL compilers treat a line of COBOL code as if it contained two distinct areas. These are known as; Area A and Area B
- When a COBOL compiler recognizes these two areas, all division, section, paragraph names, FD entries and 01 level numbers must start in Area A. All other sentences must start in Area B.
- Area A is four characters wide and is followed by Area B.
- In Microfocus COBOL the compiler directive `$ SET SOURCEFORMAT"FREE"` frees us from all formatting restrictions.



COBOL coding rules

- Almost all COBOL compilers treat a line of COBOL code as if it contained two distinct areas. These are known as;
  Area A and Area B
- When a COBOL compiler recognizes these two areas, all division, section, paragraph names, FD entries and 01 level numbers must start in Area A. All other sentences must start in Area B.
- Area A is four characters wide and is followed by Area B.
- In Microfocus COBOL the compiler directive `$ SET SOURCEFORMAT"FREE"` frees us from all formatting restrictions.

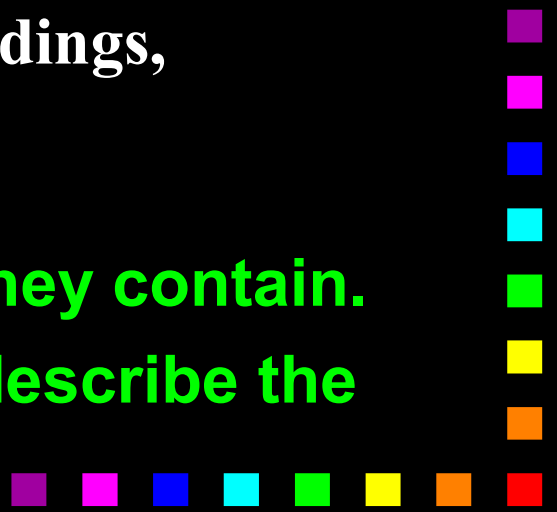
```
      $ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID.  ProgramFragment.
* This is a comment. It starts
* with an asterisk in column 1
```

Name Construction.

- All user defined names, such as data names, paragraph names, section names and mnemonic names, must adhere to the following rules;
  - They must contain at least one character and not more than 30 characters.
  - They must contain at least one alphabetic character and they must not begin or end with a hyphen.
  - They must be contructed from the characters A to Z, the number 0 to 9 and the hyphen.

    e.g. TotalPay, Gross-Pay, PrintReportHeadings, Customer10-Rec

- All data-names should describe the data they contain.
- All paragraph and section names should describe the function of the paragraph or section.

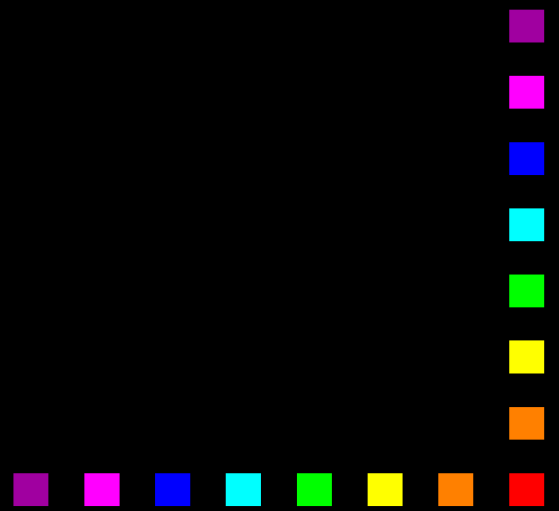


Describing DATA.

There are basically three kinds of data used in COBOL programs;

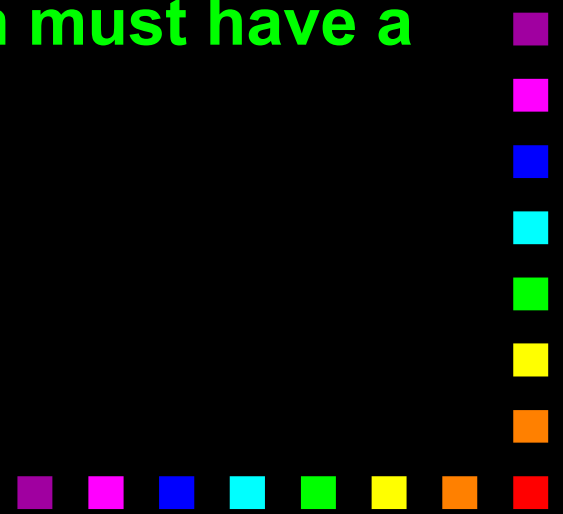
- Variables.
- Literals.
- Figurative Constants.

Unlike other programming languages, COBOL does not support user defined constants.



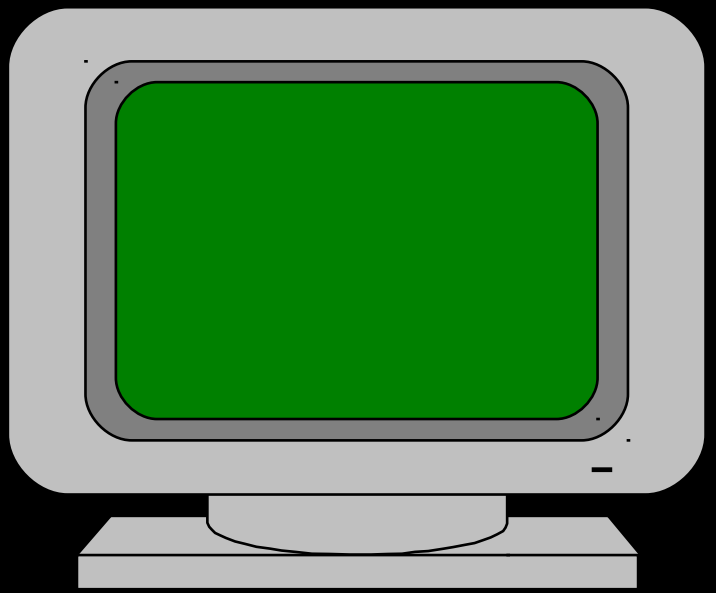
Data-Names / Variables

- A variable is a named location in memory into which a program can put data and from which it can retrieve data.
- A data-name or identifier is the name used to identify the area of memory reserved for the variable.
- Variables must be described in terms of their type and size.
- Every variable used in a COBOL program must have a description in the DATA DIVISION.



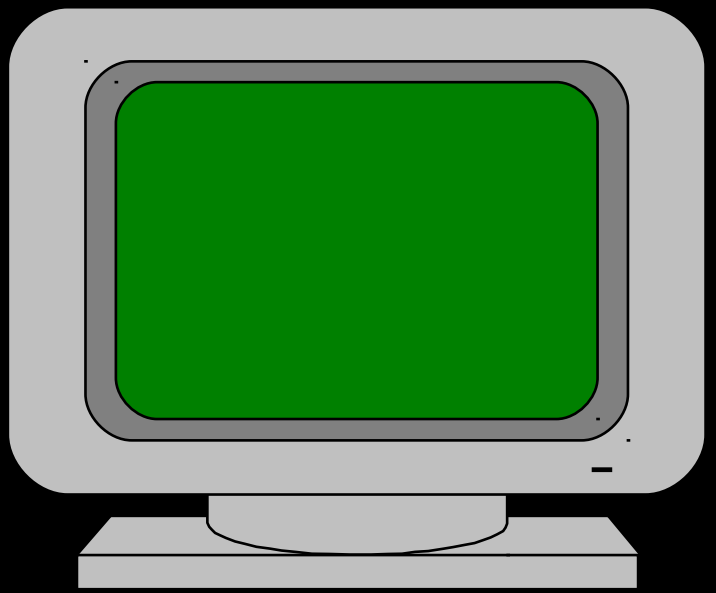
Using Variables

```
01 StudentName   PIC X(6) VALUE SPACES.

MOVE "JOHN" TO StudentName.
DISPLAY "My name is ", StudentName.
```



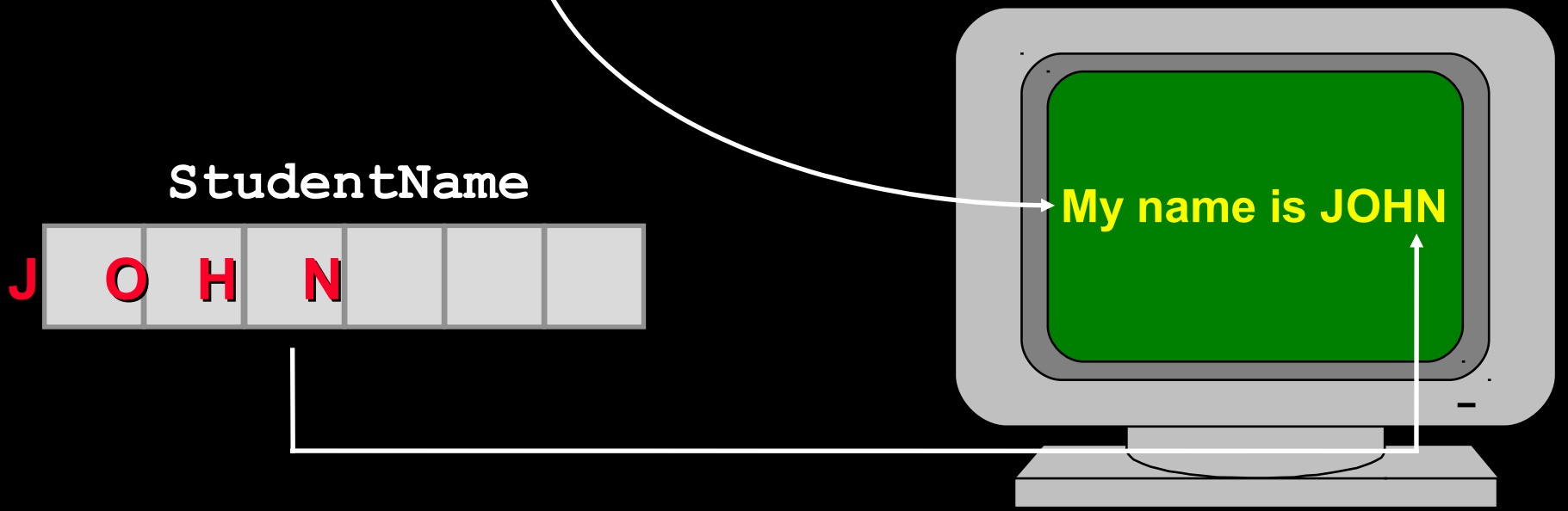
Using Variables

```
01 StudentName   PIC X(6) VALUE SPACES.

MOVE "JOHN" TO StudentName.
DISPLAY "My name is ", StudentName.
```



Using Variables

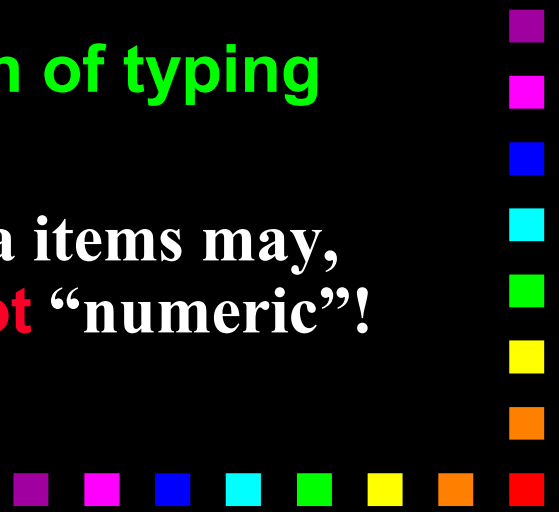
```
01 StudentName   PIC X(6) VALUE SPACES.

MOVE "JOHN" TO StudentName.
DISPLAY "My name is ", StudentName.
```



COBOL Data Types

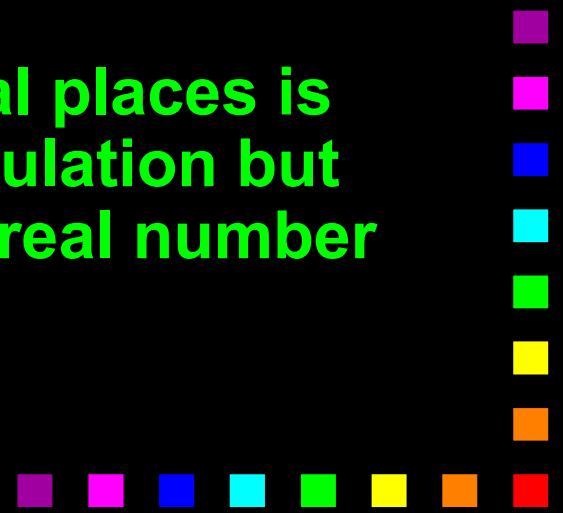
- COBOL is not a “typed” language and the distinction between some of the data types available in the language is a little blurred.
- For the time being we will focus on just two data types,
  - numeric
  - text or string
- Data type is important because it determines the operations which are valid on the type.
- COBOL is not as rigorous in the application of typing rules as other languages.

  For example, some COBOL “numeric” data items may, from time to time, have values which are not “numeric”!



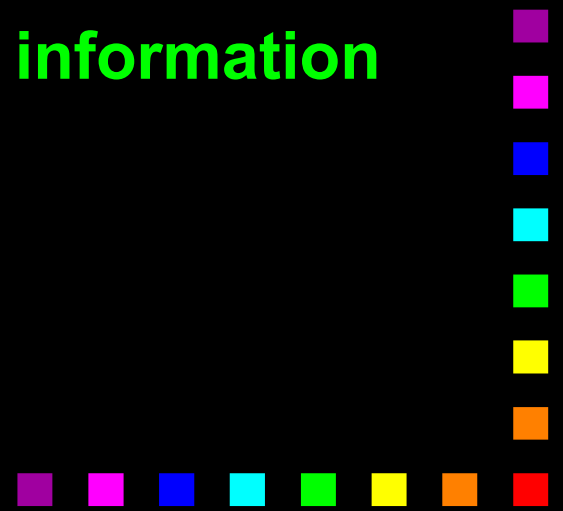
Quick Review of "Data Typing"

- In "typed" languages simply specifying the type of a data item provides quite a lot of information about it.
- The type usually determines the range of values the data item can store.
  - For instance a CARDINAL item can store values between 0..65,535 and an INTEGER between -32,768..32,767
- From the type of the item the compiler can establish how much memory to set aside for storing its values.
- If the type is "REAL" the number of decimal places is allowed to vary dynamically with each calculation but the amount of the memory used to store a real number is fixed.



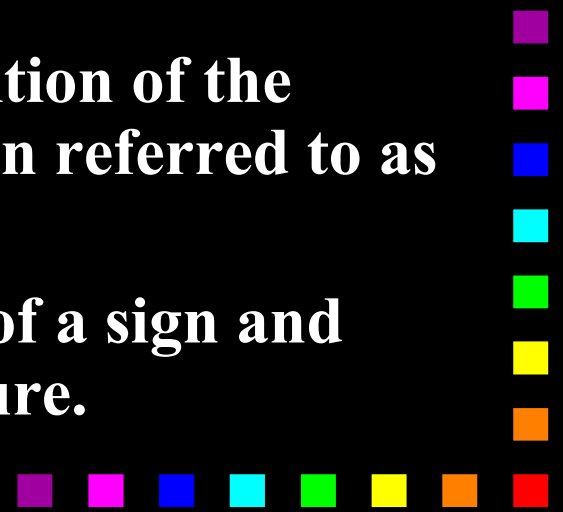
COBOL data description.

- Because COBOL is not typed it employs a different mechanism for describing the characteristics of the data items in the program.
- COBOL uses what could be described as a “declaration by example” strategy.
- In effect, the programmer provides the system with an example, or template, or PICTURE of what the data item looks like.
- From the “picture” the system derives the information necessary to allocate it.



COBOL 'PICTURE' Clause symbols

- To create the required 'picture' the programmer uses a set of symbols.
- The following symbols are used frequently in picture clauses;
  - 9 (the digit nine) is used to indicate the occurrence of a digit at the corresponding position in the picture.
  - X (the character X) is used to indicate the occurrence of any character from the character set at the corresponding position in the picture
  - V (the character V) is used to indicate position of the decimal point in a numeric value! It is often referred to as the "assumed decimal point" character.
  - S (the character S) indicates the presence of a sign and can only appear at the beginning of a picture.

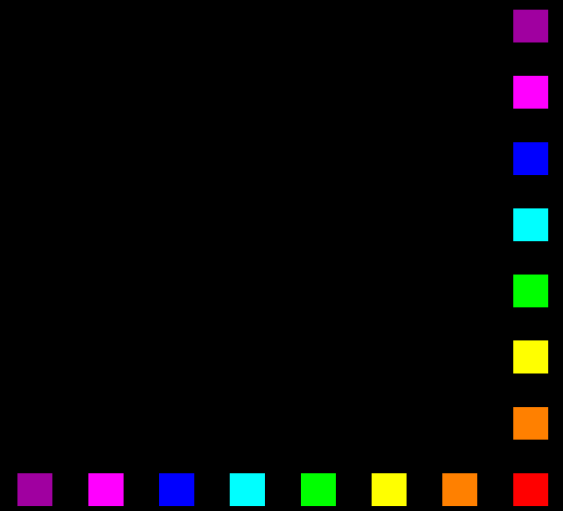


COBOL PICTURE Clauses

- Some examples
  - PICTURE 999 — a three digit (+ive only) integer
  - PICTURE S999 — a three digit (+ive/-ive) integer
  - PICTURE XXXX — a four character text item or string
  - PICTURE 99V99 — a +ive ‘real’ in the range 0 to 99.99
  - PICTURE S9V9 — a +ive/-ive ‘real’ in the range ?
- If you wish you can use the abbreviation PIC.
- Numeric values can have a maximum of 18 (eighteen) digits (i.e. 9’s).
- The limit on string values is usually system-dependent.

Abbreviating recurring symbols

- Recurring symbols can be specified using a ‘repeat’ factor inside round brackets
  - PIC 9(6) is equivalent to PICTURE 999999
  - PIC 9(6)V99 is equivalent to PIC 999999V99
  - PICTURE X(10) is equivalent to PIC XXXXXXXXXX
  - PIC S9(4)V9(4) is equivalent to PIC S9999V9999
  - PIC 9(18) is equivalent to PIC 999999999999999999



Declaring DATA in COBOL

- In COBOL a variable declaration consists of a line containing the following items;

  Œ A level number.

  ❷ A data-name or identifier.

  Ž A PICTURE clause.

- We can give a starting value to variables by means of an extension to the picture clause called the value clause.

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01   Num1            PIC 999     VALUE ZEROS.
01   VatRate         PIC V99     VALUE .18.
01   StudentName     PIC X(10)   VALUE SPACES.
```

DATA

Num1	VatRate	StudentName
000	.18	

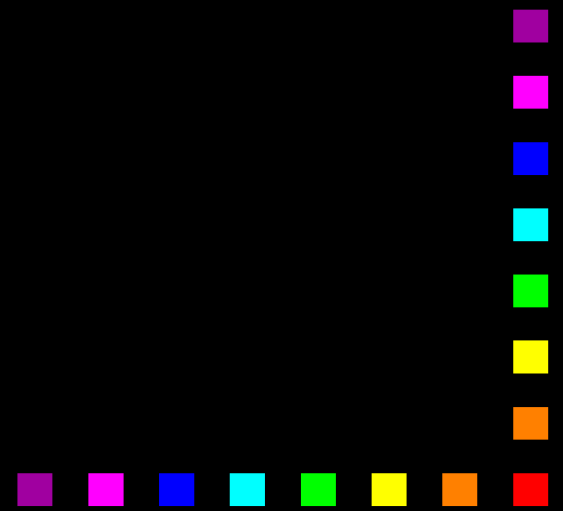
COBOL
Literals.

- String/Alphanumeric literals are enclosed in quotes and may consists of alphanumeric characters

  e.g. "Michael Ryan", "-123", "123.45"

- Numeric literals may consist of numerals, the decimal point and the plus or minus sign. Numeric literals are not enclosed in quotes.

  e.g. 123, 123.45, -256, +2987



Figurative Constants

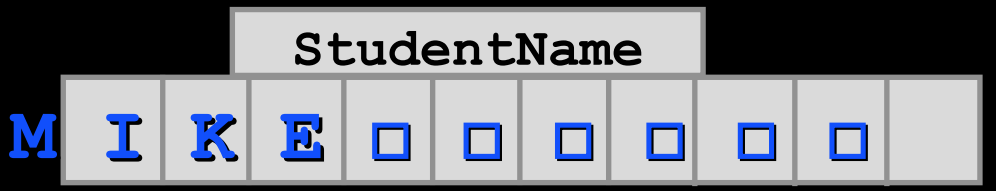
- COBOL provides its own, special constants called Figurative Constants.

SPACE or SPACES	=	□
ZERO or ZEROS or ZEROS	=	0
QUOTE or QUOTES	=	"
HIGH-VALUE or HIGH-VALUES	=	Max Value
LOW-VALUE or LOW-VALUES	=	Min Value
ALL *literal*	=	Fill With Literal

Figurative Constants - Examples

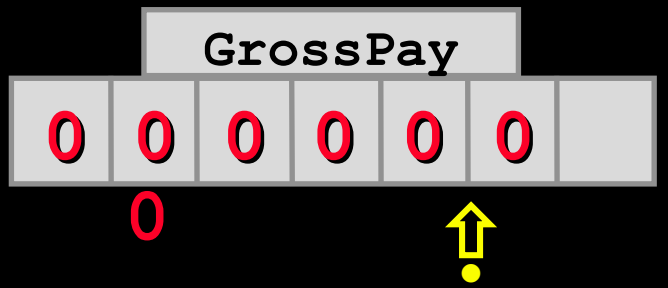
```
01   GrossPay    PIC 9(5)V99 VALUE 13.5.

       ZERO
MOVE   ZEROS       GrossPay.
       ZEROES
```


```
01   StudentName    PIC X(10) VALUE "MIKE".

MOVE ALL "-" TO StudentName.
```



Figurative Constants - Examples

```
01   GrossPay    PIC 9(5)V99 VALUE 13.5.

       ZERO
MOVE   ZEROS       GrossPay.
       ZEROES
```




```
01   StudentName    PIC X(10) VALUE "MIKE".

MOVE ALL "-" TO StudentName.
```