Introduction to Programming with Python

code LAGOS
An Initiative of the Lagos State Ministry of Education

SPARK I.T. Boot Camp ...ignite the passion

Python Programming

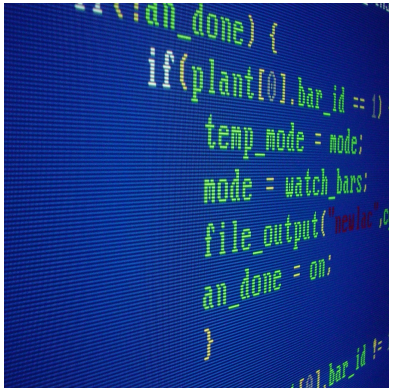
Adedipo Olagbegi
dipo@delphitechnologies.com

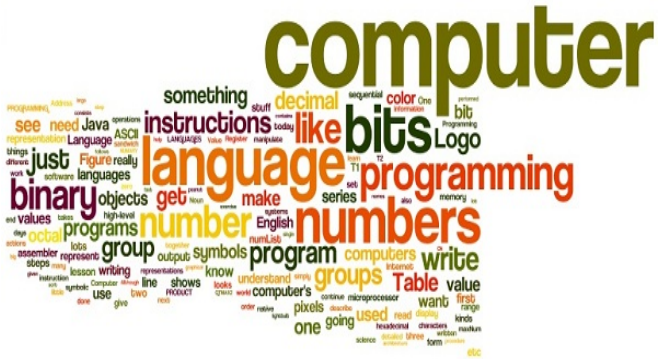
AIM

To know what programming languages a

To know why Python

To learn some Python programming





Programming objectives

"design, write and debug programs that accomplish specific goals, including controlling or simulating physical systems; solve problems by decomposing them into smaller parts "

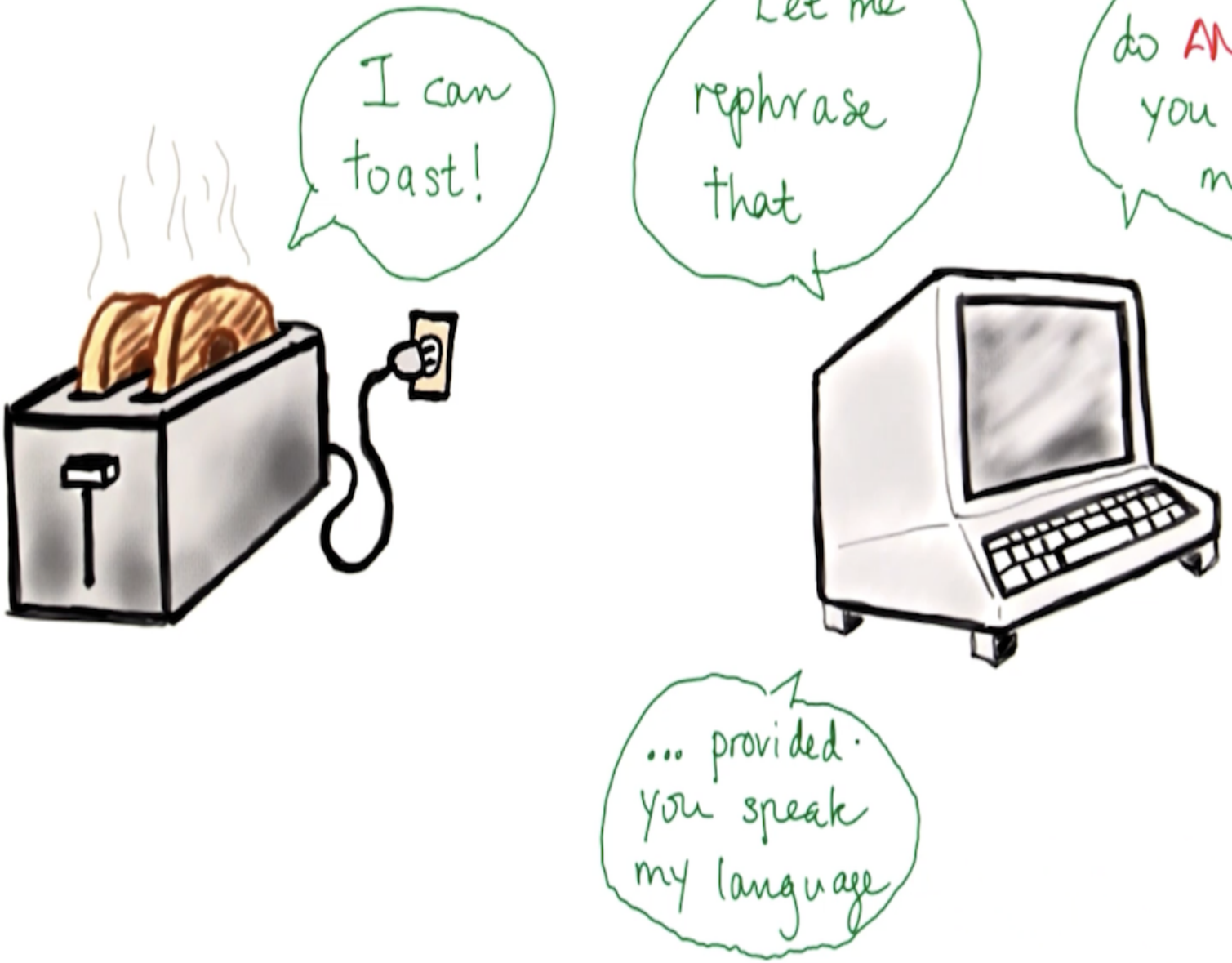
"use sequence, selection, and repetition in programs; work with variables and various forms of input and output "

"use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs "

SS3 Competency Target

"use two or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions"

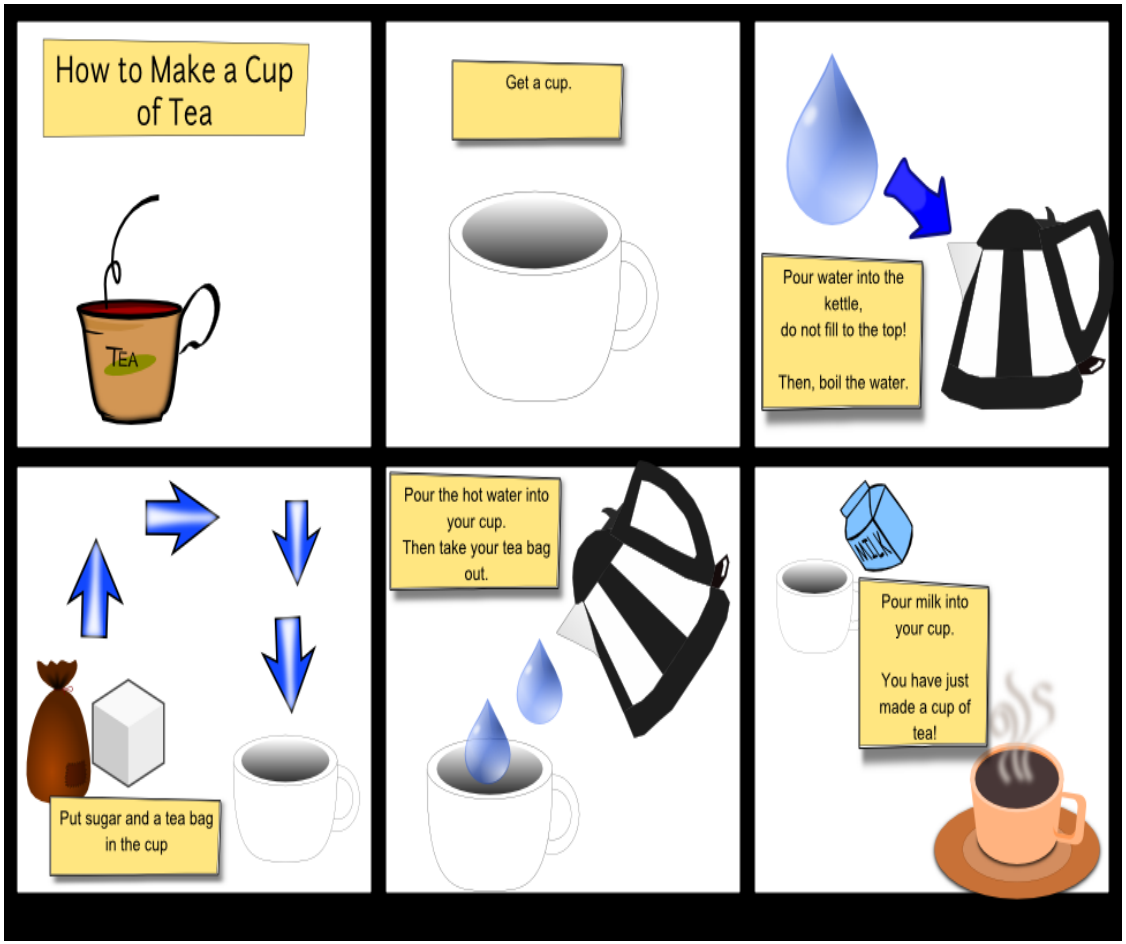
What is Programming?



What is software and what is programming ?

- **Computers understand 1 and 0's (binary).**
- **They can do things very simple things in binary very fast.**
- **Very difficult for and time consuming to do anything at this level.**
- **We need to know the detail and connection in the comp**

Please make me a cup of tea?



Programming languages

- A program running on the computer
- Takes instructions that we write
  - E.g Print Hello
  - Convert them into code the computer can understand and execute
- Lots of different languages

Programming and Software

- The art of creating these programs is programming.

Software

m we produce are c

I

Hardware

Why learn programming

- Get a job programming
- It teaches you to be a logical thinker
- Computers are everywhere these days, so there's a good chance you'll use computers at work, at school, or at home—probably all three. Learning about programming will help you understand computers better in general.
- Where are computer programs used. ?

Why Python ?

- Easy to read and learn
- Free and open source
- Is a real language used in industry
- Runs on many platforms

Interpreted v Compiled Languages

Which ever language we write in, our program must be converted into code that the computer hardware can understand (machine code).

- Compiled Languages convert the whole program into machine code to produce an executable file. You can then run the '.exe' file.

- Interpreted Languages take one line of the program at a time and convert it into machine code. They then execute the code for that line.

Python is an Interpreted Language.

Flowcharts and Pseudocode

Programming Tools

- Three tools are used to convert algorithms into computer programs:
- **Flowchart** - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- **Pseudocode** - Uses English-like phrases with some Visual Basic terms to outline the program.

Problem solving example

- How many stamps do you use when mailing a letter?
- One rule of thumb is to use one stamp for every five sheets of paper or fraction thereof.

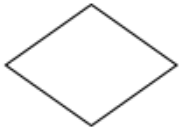
Algorithm

1. Request the number of sheets of paper; call it Sheets. *(input)*
2. Divide Sheets by 5. *(processing)*
3. Round the quotient up to the next highest whole number; call it Stamps. *(processing)*
4. Reply with the number Stamps. *(output)*

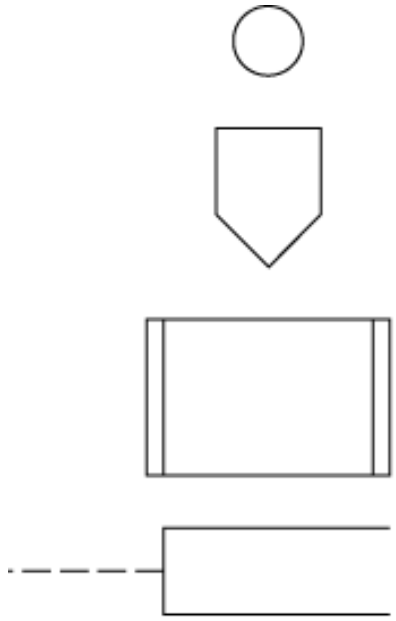
Flowcharts

- Graphically depict the logical steps to carry out a task and show how the steps relate to each other.

Flowchart symbols

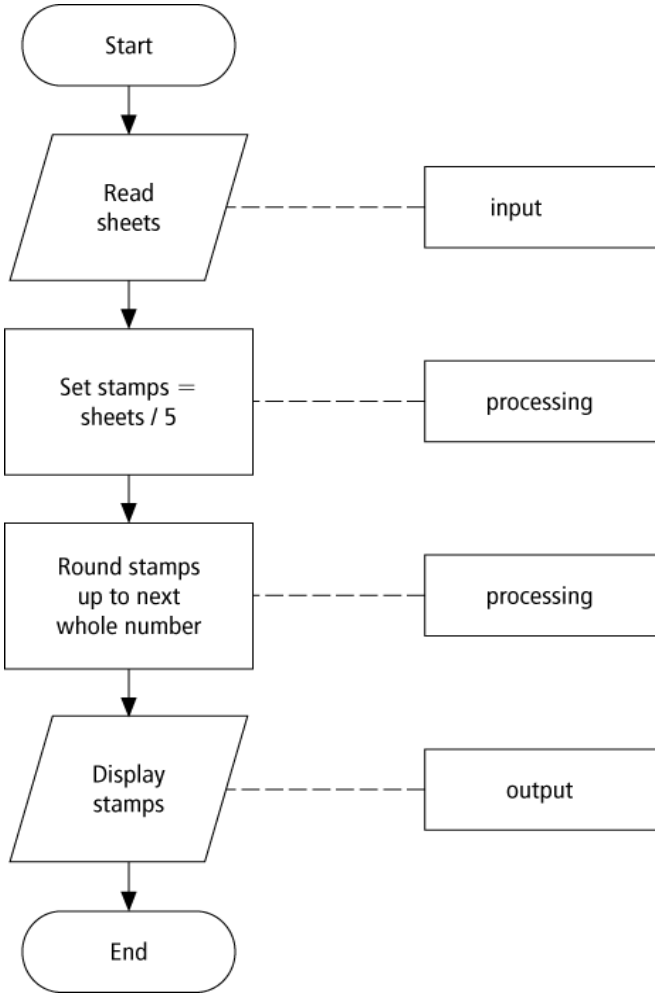
Symbol	Name	Meaning
	Flowline	Used to connect symbols and indicate the flow of logic.
	*Terminal*	Used to represent the beginning (Start) or the end (End) of a task.
	*Input/Output*	Used for input and output operations, such as reading and displaying. The data to be read or displayed are described inside.
	*Processing*	Used for arithmetic and data-manipulation operations. The instructions are listed inside the symbol.
	*Decision*	Used for any logic or comparison operations. Unlike the input/ouput and processing symbols, which have one entry and one exit flowline, the decision symbol has one entry and two exit paths. The path chosen depends on whether the answer to a question is "yes" or "no."

Flowchart symbols continued



Connector	Used to join different flowlines.
Offpage Connector	Used to indicate that the flowchart continues to a second page.
Predefined Process	Used to represent a group of statements that perform one processing task.
Annotation	Used to provide additional information about another flowchart symbol.

Flowchart example



Pseudocode

- Uses English-like phrases to outline the task.

Pseudocode example

Determine the proper number of stamps for a letter

Read Sheets *(input)*

Set the number of stamps to Sheets / 5 *(processing)*

Round the number of stamps up to the next whole number *(processing)*

Display the number of stamps *(output)*

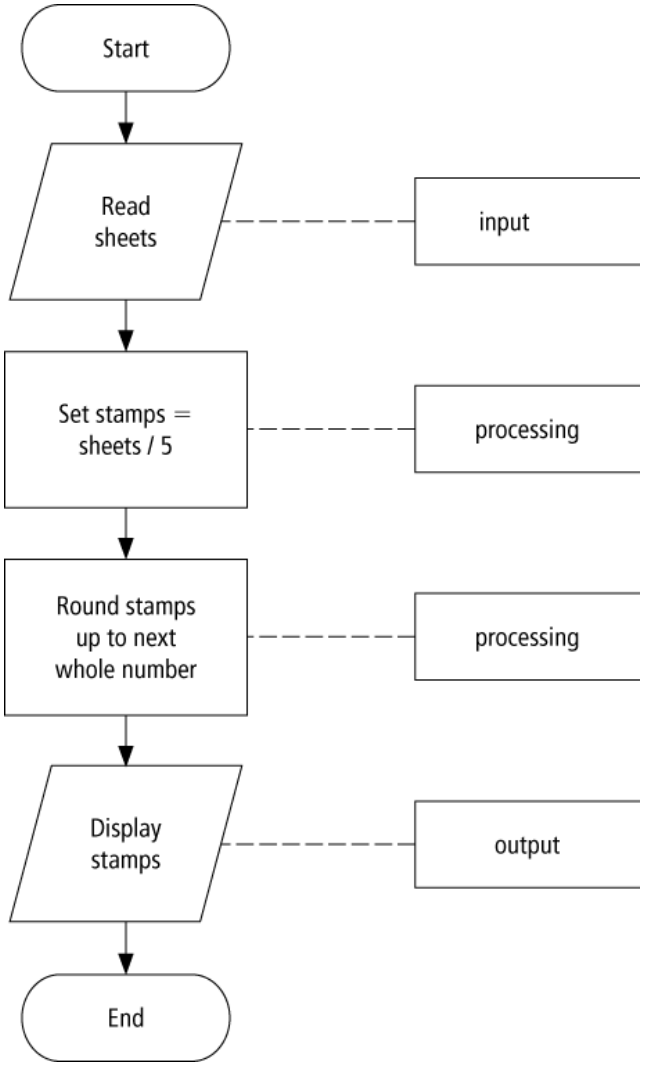
Divide-and-conquer method

- Used in problem solving – take a large problem and break it into smaller problems solving the small ones first
- Breaks a problem down into modules

Statement structures

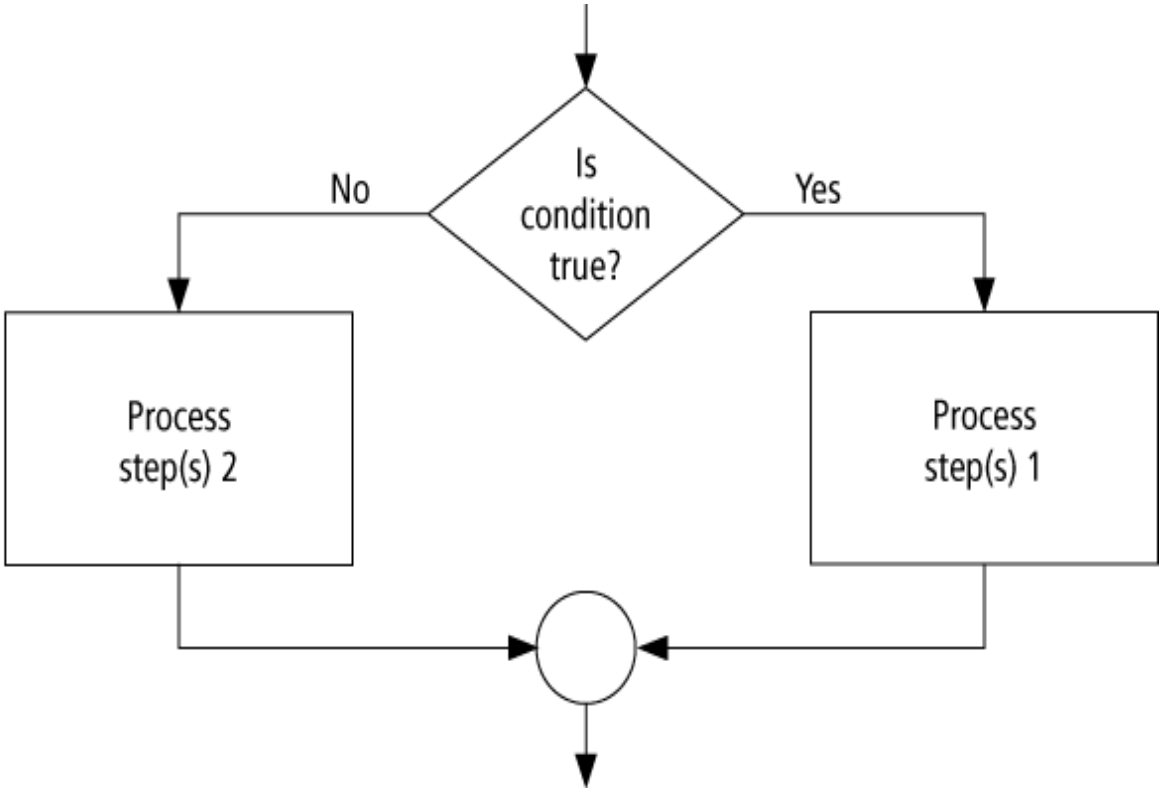
- Sequence – follow instructions from one line to the next without skipping over any lines
- Decision - if the answer to a question is “Yes” then one group of instructions is executed. If the answer is “No,” then another is executed
- Looping – a series of instructions are executed over and over

Sequence flow chart



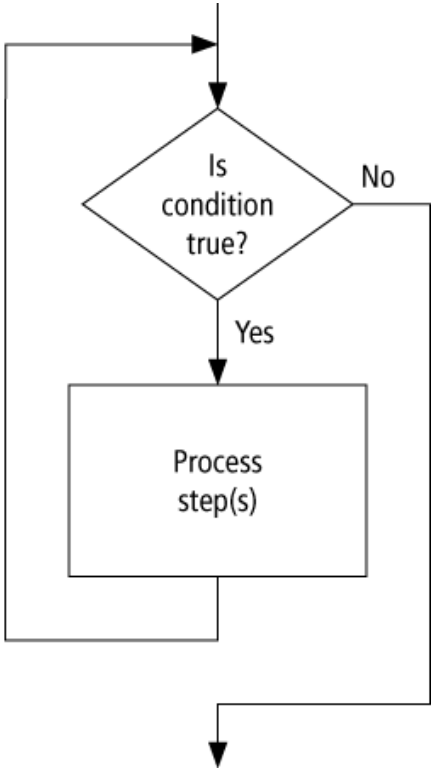
Decision flow chart

```
If condition is true Then
    Process step(s) 1
  Else
    Process step(s) 2
End If
```



Looping flow chart

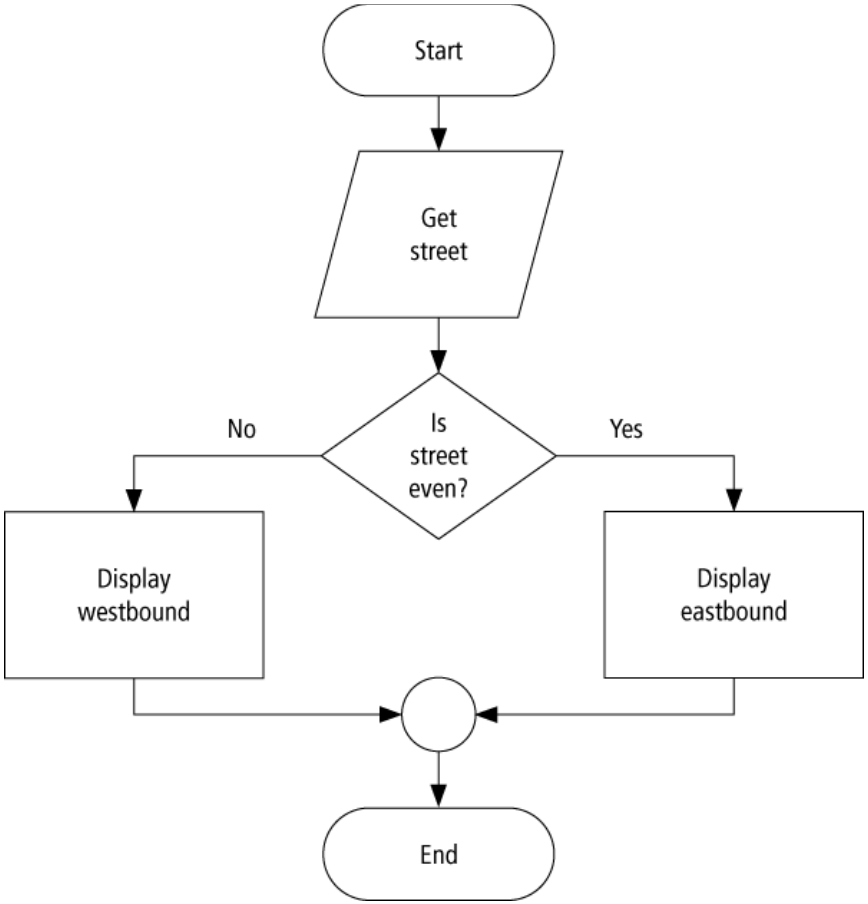
```
Do While condition is true
    Process step(s)
Loop
```



Direction of Numbered NYC Streets Algorithm

- ***Problem:*** Given a street number of a one-way street in New York City, decide the direction of the street, either eastbound or westbound
- ***Discussion:*** in New York City even numbered streets are Eastbound, odd numbered streets are Westbound

Flowchart


Start
Get street
Is street even?
No
Yes
Display westbound
Display eastbound
End

Pseudocode

Program: Determine the direction of a numbered NYC street

```
Get street
If street is even Then
  Display Eastbound
Else
  Display Westbound
End If
```

Class Average Algorithm

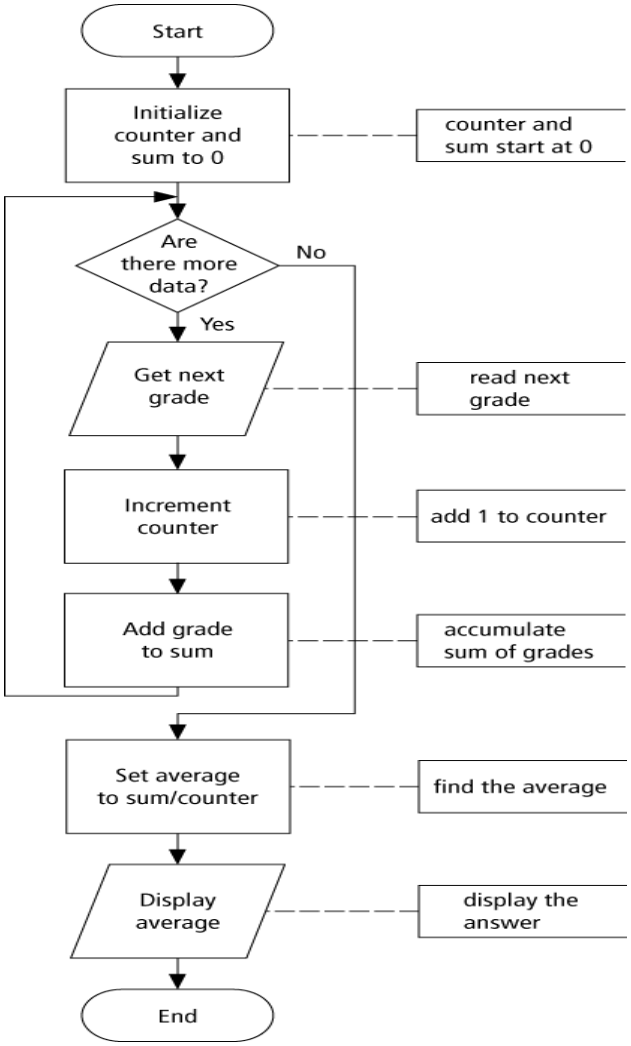
- ***Problem:*** Calculate and report the grade-point average for a class
- ***Discussion:*** The average grade equals the sum of all grades divided by the number of students

Output: Average grade

Input: Student grades

Processing: Find the sum of the grades; count the number of students; calculate average

Flowchart


Start
Initialize counter and sum to 0
counter and sum start at 0
Are there more data?
No
Yes
Get next grade
read next grade
Increment counter
add 1 to counter
Add grade to sum
accumulate sum of grades
Set average to sum/counter
find the average
Display average
display the answer
End

Pseudocode

Program: Determine the average grade of a class

Initialize Counter and Sum to 0

Do While there are more data

    Get the next Grade

    Add the Grade to the Sum

    Increment the Counter

Loop

Computer Average = Sum / Counter

Display Average

Tips and tricks of flowcharts

- Flowcharts are time-consuming to write and difficult to update
- For this reason, professional programmers are more likely to favor pseudocode and hierarchy charts
- Because flowcharts so clearly illustrate the logical flow of programming techniques, they are a valuable tool in the education of programmers

Tips and tricks of pseudocode

- There are many styles of pseudocode
- Some programmers use an outline form
- Some use a form that looks almost like a programming language
- The pseudocode in the case studies of this text focus on the primary tasks to be performed by the program and leaves many of the routine details to be completed during the coding process

URL's for free Flowcharting software

- ***www.smartdraw.com***
- ***www.gliffy.com/uses/flowchart-software/***
- ***www.breezetree.com/flowcharting-software/***

***Flowchart Software*, FREE Flowchart Examples and Templates ...**

- ***www.edrawsoft.com/flowchart.php***


LET'S
GET
GOING

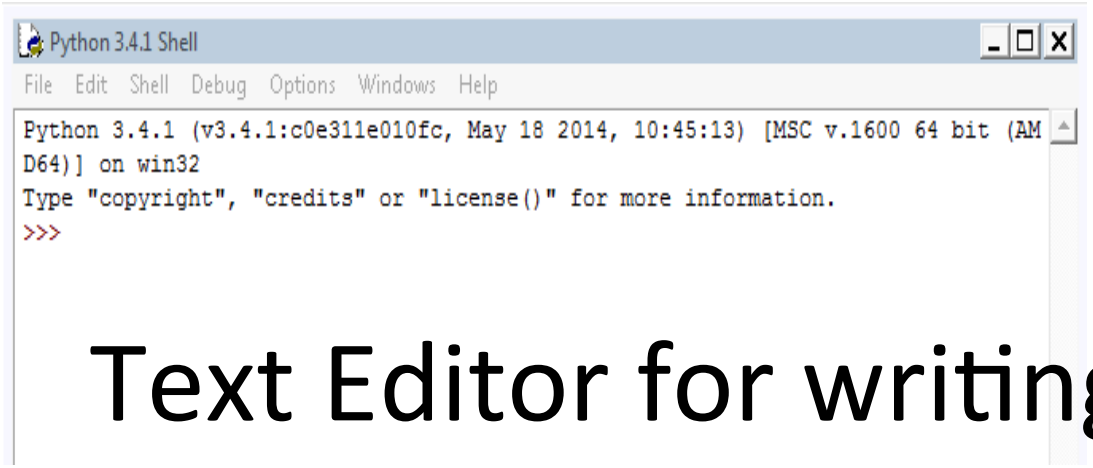
Programming Environment

What you need

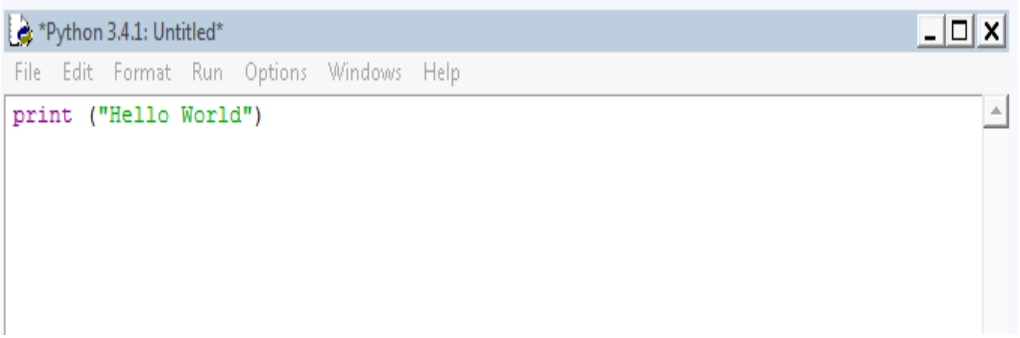
- A text editor (NOT a word processor)
- A python interpreter.
- http://www.python.org

- What you get is IDLE (Integrated Development Environment)
- And the python interpreter (in Windows in its own root folder)

Python Shell for interactive input.

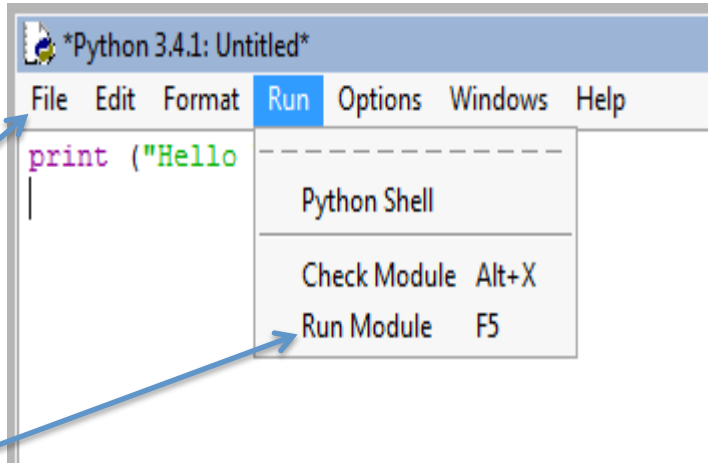


Text Editor for writing and running scripts



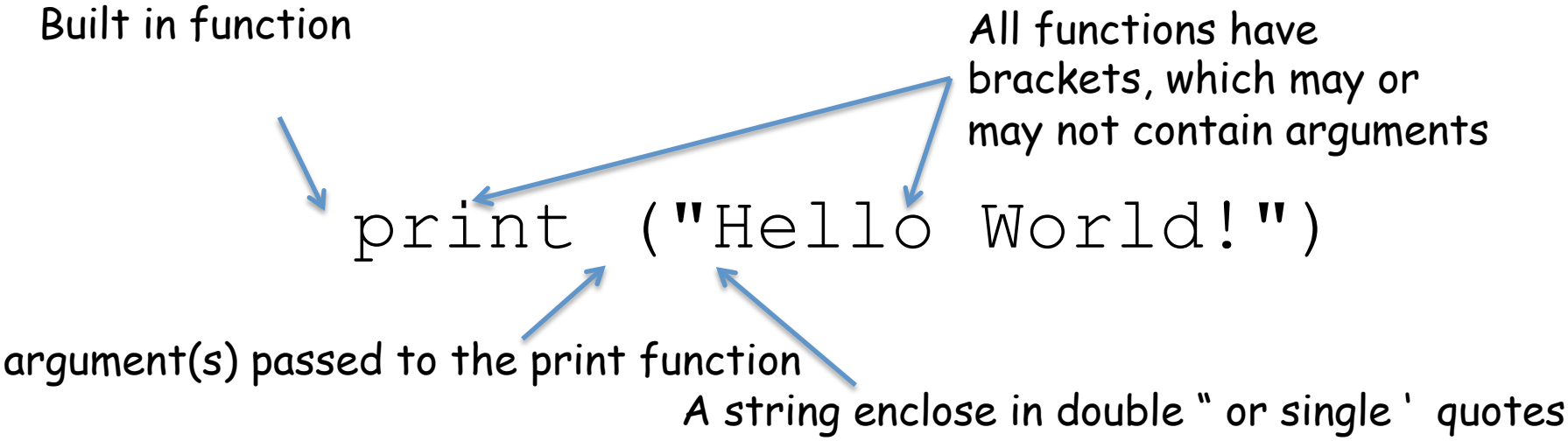
Right of passage 01_Hello_world

```
print ("Hello World!”)
```



Save it and then run it.
Look in the shell window for your output

Right of passage Hello_world



What would this do ?

```
print('So what do "you" think')
```

Python and Numbers

```
print("OK Dipo !")
print(5 + 11)
print("5 + 11")
```

What happened and why ?

Python numbers, strings and arithmetic

```
print("Dipo" + "Maidens”)
print("Dipo","Maidens”)
print("5" + "11”)
print("Python is Ace" * 5)
```

The asterisk ‘*’ is a multiplication

What about ?
04_strings_and_maths_2

```
print("Dipo" + 5)
print("5" *  "11")
```

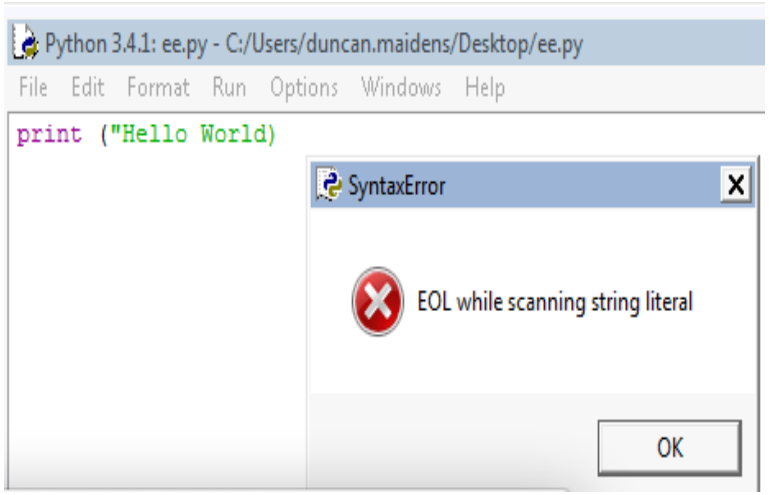
Don't worry about the messages for now. !

Would this work ?

```
print("Dipo",5)
```

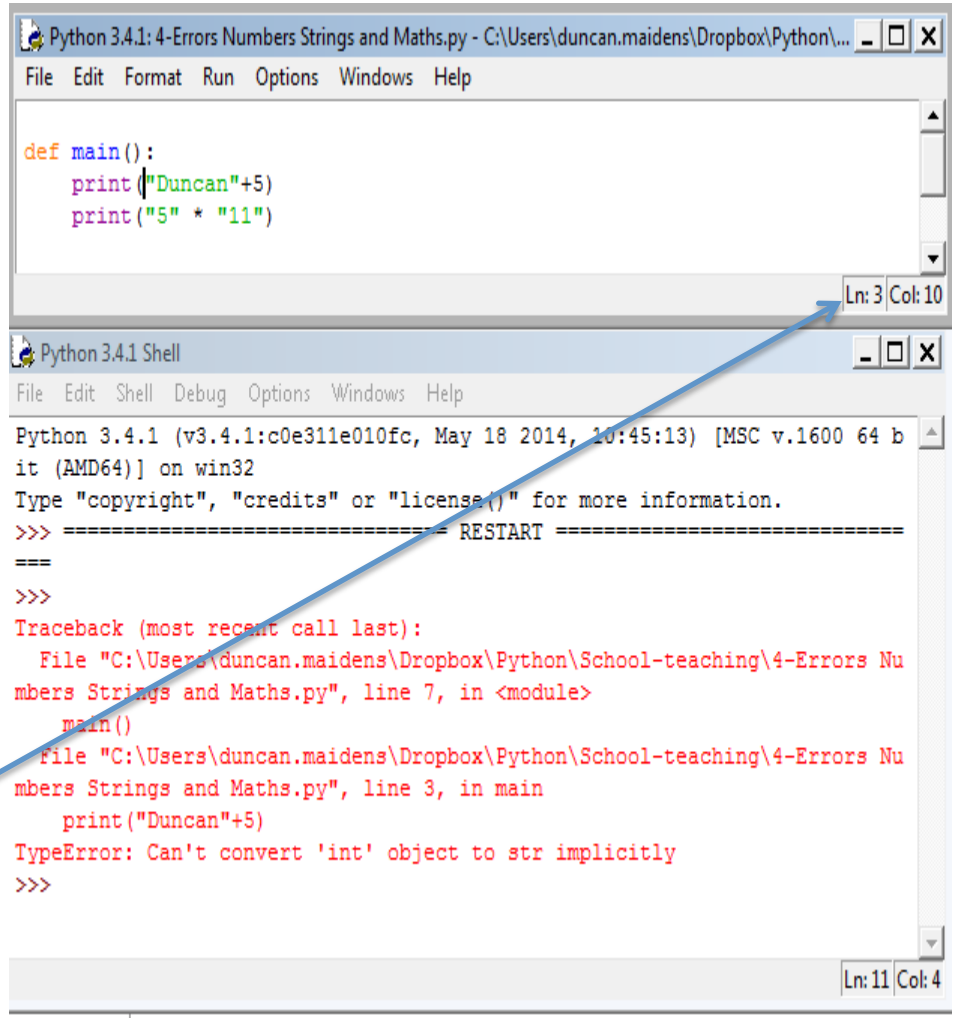
Syntax Errors

- Errors in the way you have set out your program.
- Missing quotes, brackets or colons
- Incorrect indentation
- Highlighted in the program by IDLE
- Picked up before the program has started to run



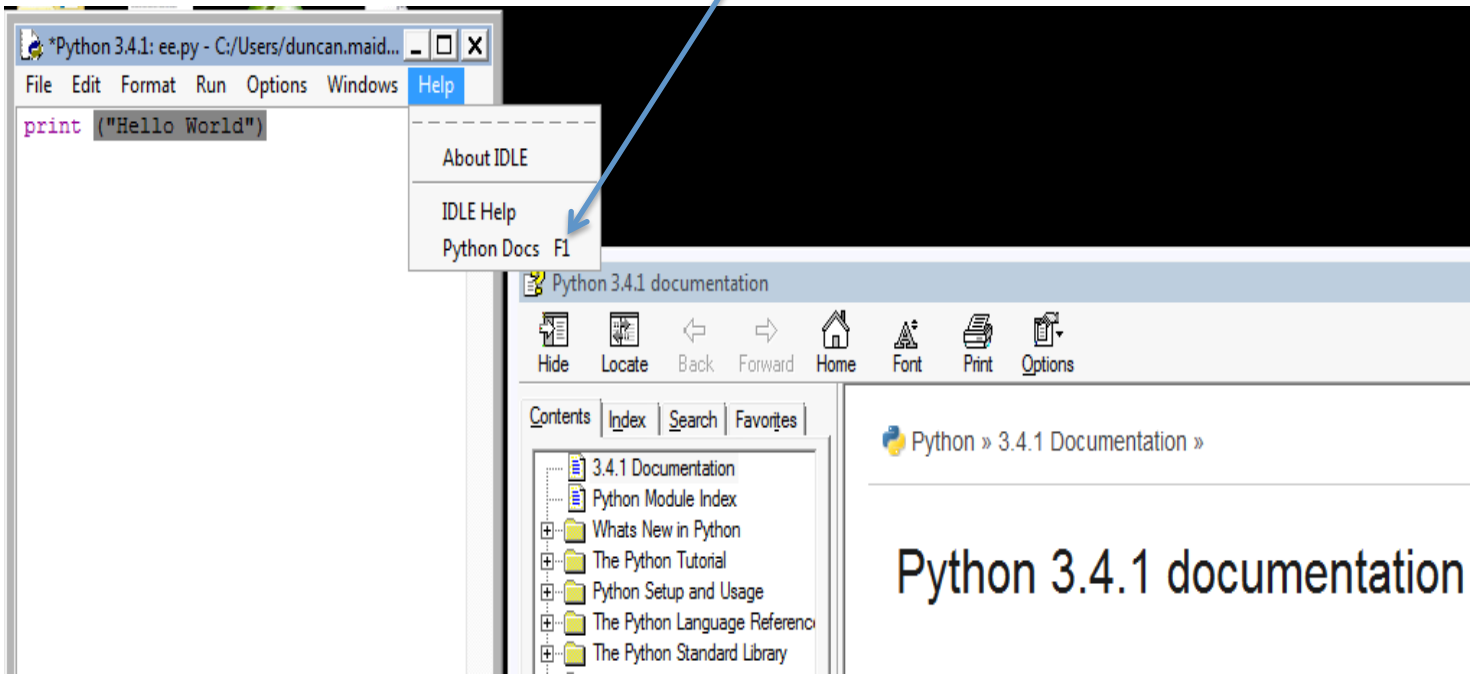
Runtime Errors

- Errors in the way you expected the program to work.
- Things that don't make sense to python when it runs
- A lack of understanding of what the instructions do.
- Failure in your logic !
- Look for the line the error is in.



Where to go for help !

- Google it ! (Watch the version !!!)
- Python Docs (part of the install)



Variables and types of variables

```
name = "Dipo"
age = 24
print(name,"is",age)
```

- Variables are containers with labels .
- We can refer to the contents by using the label
- The containers ore of different types, depending on the type of variable.
- '=' is an assignment operator. It assigns the values on the right to the variable on the left.

Data Types

- So far we have looked at 3 types of data:-
  - integers – whole numbers (+ve and –ve) no limit
  - strings – sequence of ASCII characters
  - floats – decimal numbers – watch precision
- Boolean Type
  - True or False
  - Usually generated as the result of a test
    - legal = age >= 18
  - Can be statically signed
    - brown_eyes = True

Getting data in from the keyboard.

- The 'input()' function reads the keyboard up to a return character, and returns the characters typed as a string.
- A string is a sequence of characters.

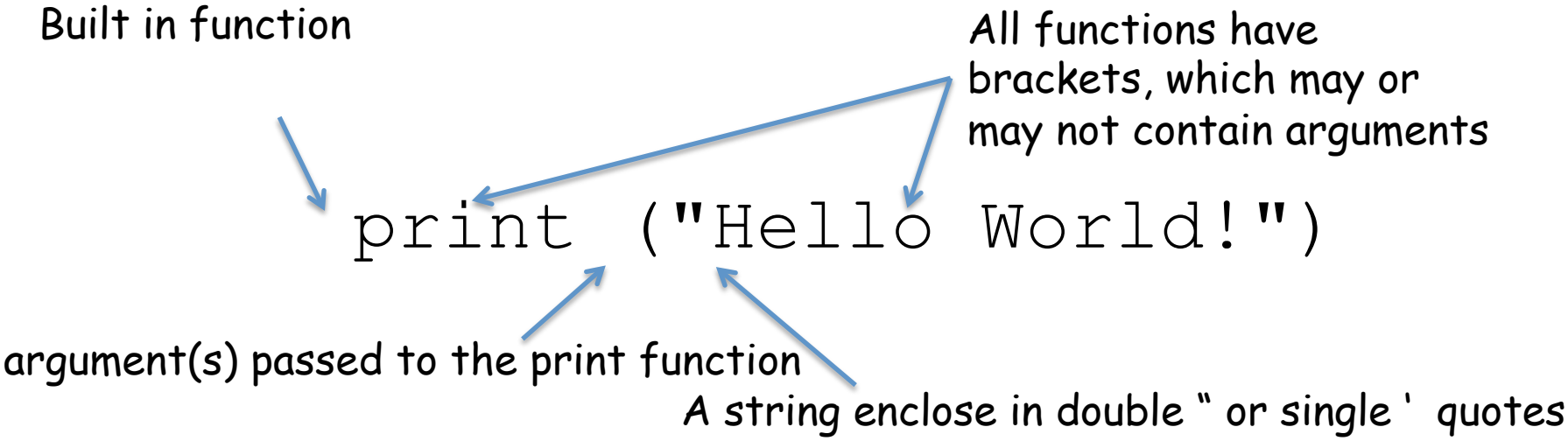
```
print("Type something in")
value = input()
print ("You typed", value)
```

- What would happen if you typed in 23 ?

Recap

- print() and input()
- Variables and data types.
- "=" is an assignment NOT 'EQUALS'
- Type conversion (strings to integers)
- Simple numeric and string operations

A function



What would this do ?

```
print('So what do "you" think')
```

Python and Numbers

```
print("OK Dipo !")
print(5 + 11)
print("5 + 11")
```

What happened and why ?

Python numbers, strings and arithmetic

```
print("Dipo" + "Maidens”)
print("Dipo","Maidens”)
print("5" + "11”)
print("Python is Ace" * 5)
```

The asterisk ‘*’ is a multiplication

What about ?
04_strings_and_maths_2

```
print("Dipo" + 5)
print("5" *  "11")
```

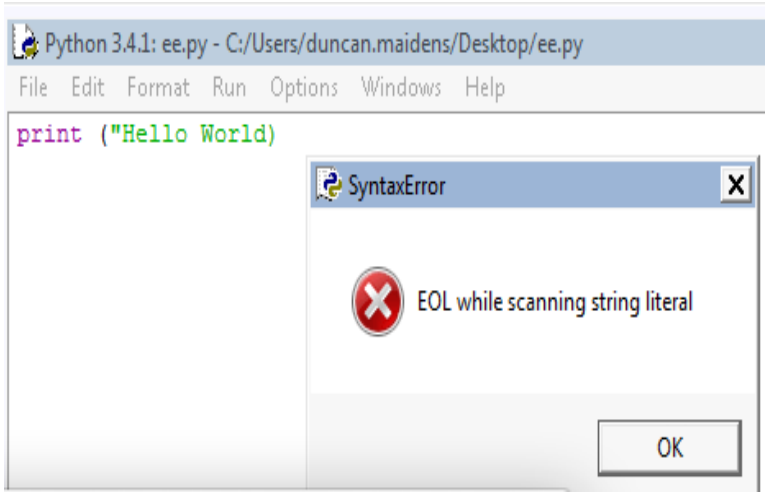
Don't worry about the messages for now. !

Would this work ?

```
print("Dipo",5)
```

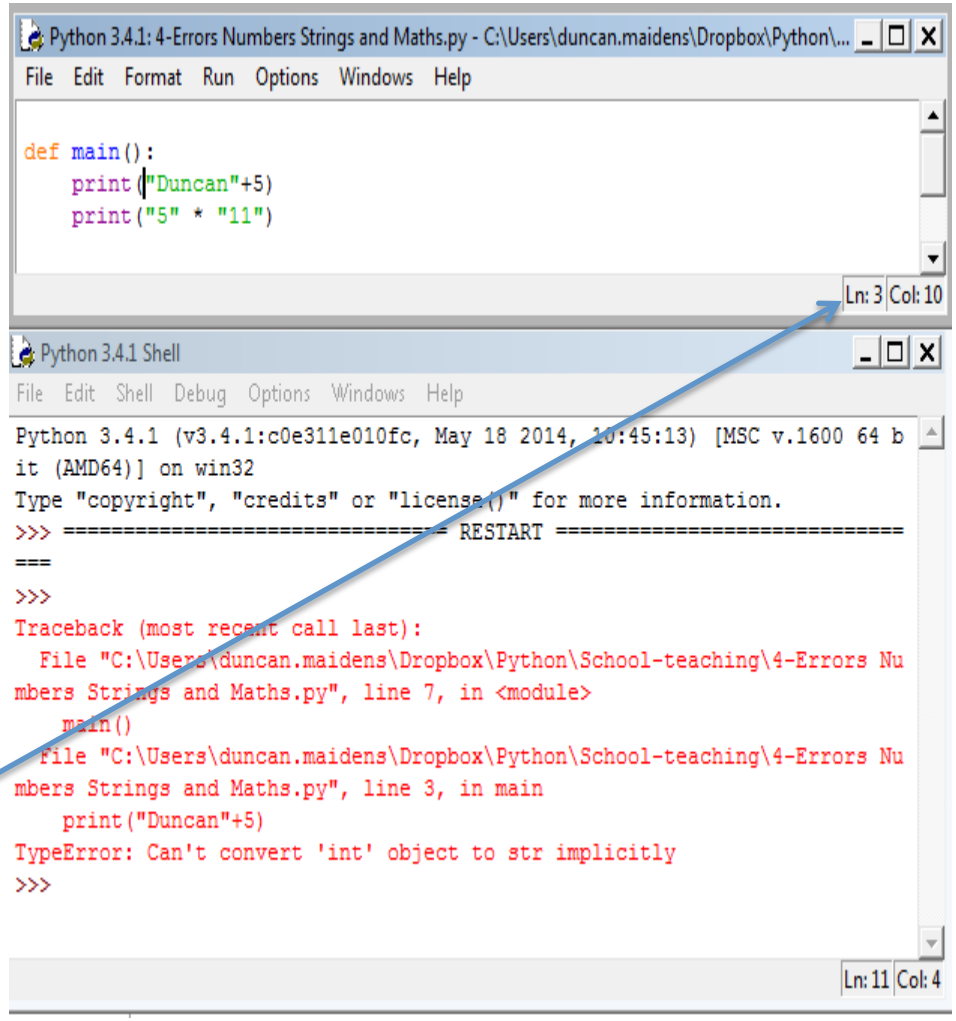
Syntax Errors

- Errors in the way you have set out your program.
- Missing quotes, brackets or colons
- Incorrect indentation
- Highlighted in the program by IDLE
- Picked up before the program has started to run



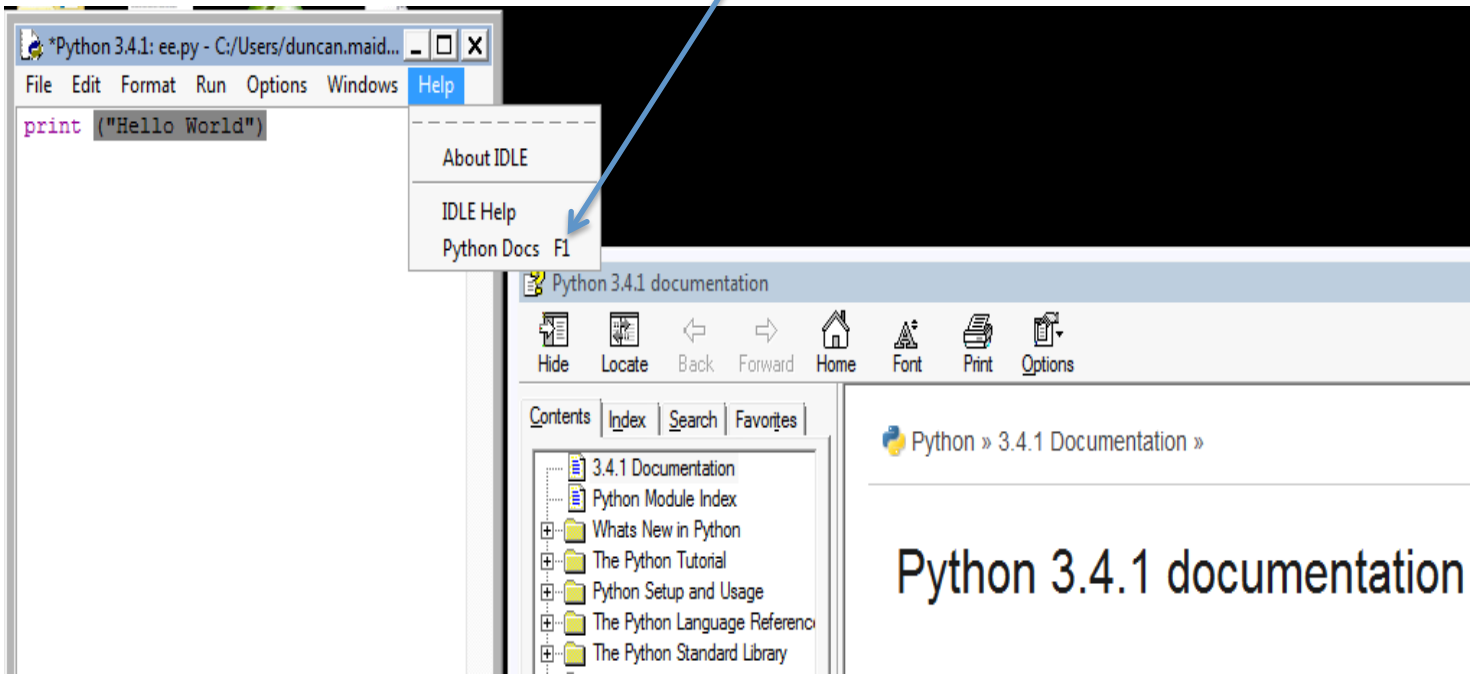
Runtime Errors

- Errors in the way you expected the program to work.
- Things that don't make sense to python when it runs
- A lack of understanding of what the instructions do.
- Failure in your logic !
- Look for the line the error is in.



Where to go for help !

- Google it ! (Watch the version !!!)
- Python Docs (part of the install)



Basic Math functions 1

- Add '+'
  - Can be used with integers to add
  - Can be used with strings to concatenate (join)

- Subtract '-'
  - Used with integers for subtraction
  - Integers can handle –ve numbers
  - Makes no sense with strings

Basic Math functions 2

- Multiply '*' - **NOTE** this is a 'star' not 'x'
  - Can be used with integers to multiply
  - It makes no sense to multiply strings.
  - What does `"Hello " * 5` result in ?

- Divide '/' and '//'
  - Used with integers for division
    - '/' is normal division (5/4 = 1.25)
    - '//' is whole number and ignores the remainder. (5 //4 = 1)
  - '/' results in a value of type 'float', where as '//' returns an integer
  - Makes no sense with strings

Basic Math functions 3

- FLOATS
  - '/' always returns a 'float' (even if it is a whole number)
  - Numeric type for decimals.
  - 2 (as a float) = 2.0
    - Python will do math with any combination of integers and floats.
    - It will convert as appropriate, but usually to a float.
- MODULUS '%'
  - Can be used with floats and integers to find the remainder after division.
  - Makes no sense with strings
  - Uses
    - If today is Monday in 29 days time it will be (29%7 = 1 (one day on)) Tuesday.

Repeating blocks of code.

- In programming we often want to repeat the same block of code several times.
- It may be exactly the same instructions, or by using variables, a value (or values) may change each time we repeat the instructions.
- If we know how many times to repeat, we can use a 'for' loop.
- We can set a variable that increments (or decrements) each time we go round the loop.
  - By setting the start value, the end value and the inc / dec, we can control how many times we repeat.
  - We can use the variable within the loop.
- The correct terminology is iterating through the loop counter (variable)

for loop syntax

```
for num in range (1,10):
    print("2 x",num,"=",
2*num)
  print("What a lot of
output !!!")
```

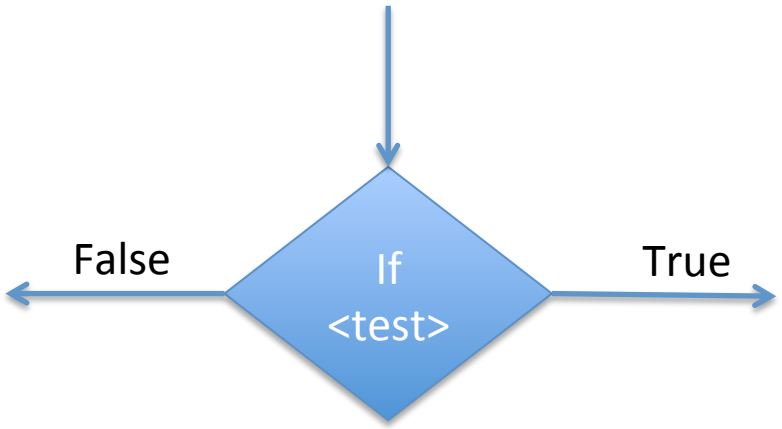
For each of the values in the list

Variable num

range (1,10) generates a list of numbers 1,2,… 10

Block to repeat is indented under the :

Not part of the repeat block



Conditional statements

```
if <I can understand this>:
   i will be happy
else:
   i will be sad ☹
```

Boolean Types

- So far we have looked at 3 types of data:-
  - integers – whole numbers (+ve and –ve) no limit
  - strings – sequence of ASCII characters
  - floats – decimal numbers – watch precision
- Boolean Type
  - True or False
  - Usually generated as the result of a test
    - legal = age >= 18
  - Can be statically signed
    - brown_eyes = True

Tests compare data items..so. Comparators in Python

- ==

  Used to test if the two operands are equal

  ```
  name == "George"
  ```

- > or >=

  Used to test if the two operands are 'greater than' or 'greater than or equals' to another item.)

  ```
  height >= 2000
  ```

  Strings are compared letter by letter starting from the first letter using each characters ASCII value

  Thus

  ```
  "Bob" > "Bill"
  ```

Comparators in Python

- < or <=

  Used to test if the two operands are 'less than' or 'less than or equals' to another item.)

  ```
  item_cost < 450
  ```

  Strings are compared letter by letter starting from the first letter using each characters ASCII value

  Thus

  ```
  "Big" < "Small"  returns True
  ```

- !=

  Used to test if two operands are not equal

  ```
  passcode  !=  "12$A*djm"
  ```

Logic and multiple comparators

We can combine comparators using logical operators.

- A and B
  - True when A and B are both True
    - `age > 21 and sex == "F"`

- A or B
  - True when either A or B are True (or both of then are True)
    - `temp > 100 or pressure > 5`

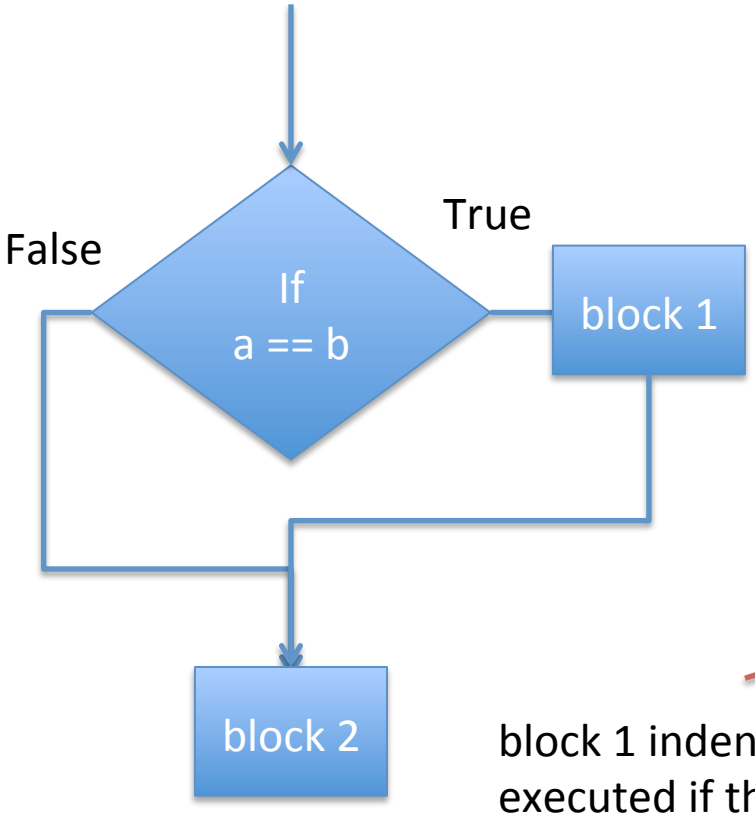
- not A
  - True when A is False
    - `not passed_exam`

- Can be combined – use () for clarity

The if statement

- Thus we can use comparators and combine them with logical operators to evaluate a set of conditions.
- The overall expression MUST return True or False and we can execute different bits of code depending on the result.

The simplest 'if'



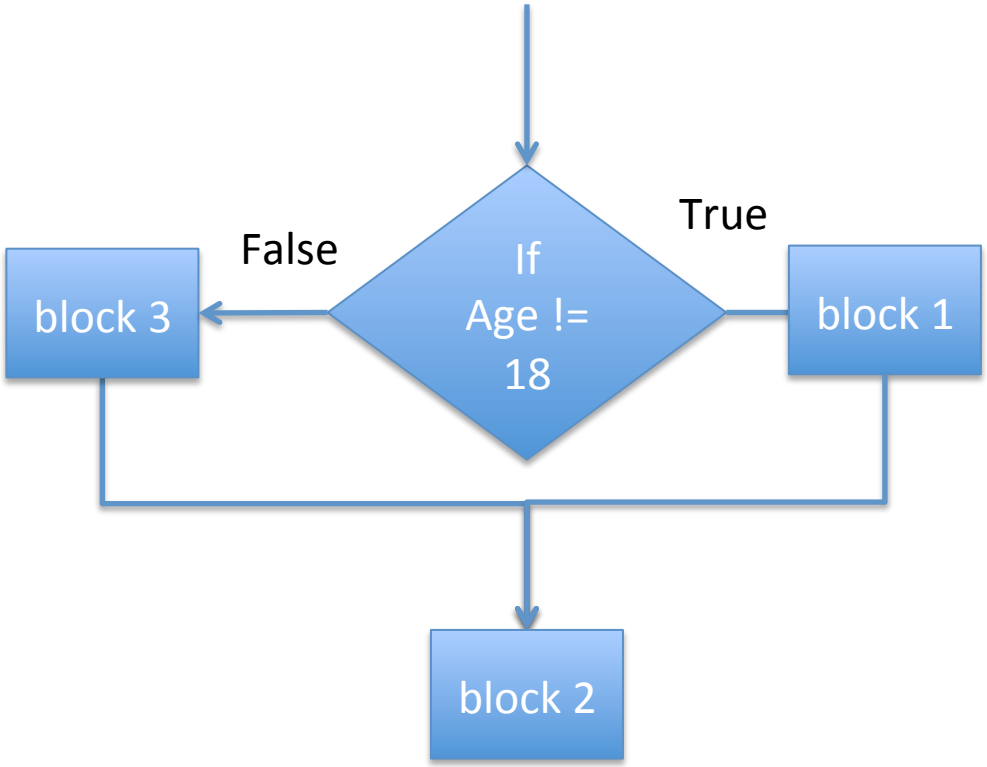
Logical test

```
if  a == b:
    block 1
block2
```

block 1 indented and only executed if the test returned True

block 2 aligned with the if. It is executed independently of the test no matter what happens

if, else



block 1 indented and only executed if the test returned True

Logical test

```
if  a == b:
  block 1
else:
  block 3
block2
```

Block 3 indented and only executed if the test returned False

block 2 aligned with the if. It is executed independently of the test no matter what happens