Worksheet 1: Introductory Exercises

- The **print** function
    - Used to display messages to the person running the Python program
        1. print(“example”)
        2. example
    - Multiple statements, each on its own line are executed in order. A group of statements run in order can be called a *block* of statements.
    - Can be used to compute mathematical equations (Python preferences * and /)
- Comments
    - Comments are “plain language” description which are only read by people, the Python program ignores these comments.
        1. # comment here
        2.
- Errors
    - **Syntax error**
        - Occurs when you type something that is not legible in the programming language
- Strings
    - Any words are assumed to be Python commands, so if you want to type words but not as commands you have to use “” to define the string. The quotation marks are **delimiters** and mark the start and end limits of text.
    - Strings are defined as a piece of text inside quotes.
    - You can use ‘ or “, as long as it’s the same at the start and end.
    - The “safest” approach is to use triple quotation marks “”” or ‘’’
- String operations
    - Can “add” two strings together
        - Called **concatenation**
            1. print(“Hello” + “World”)
            2. HelloWorld
    - Can “multiply” strings by numbers
        1. print(‘Hi’ * 3)
        2. HiHiHi

- Variables
    - Can store a value using a **variable.** Variables (identifiers) are created using an operation called **assignment**. "=" is the **assignment operator.** When printing variables the value (literals) is printed

        1. message = "Hello world"
        2. print(message)
        3. Hello world

        - If you assign a variable a new value it forgets its old value.
        - Variable names must start with an alphabetic letter or an underscore _
        - Variable names can only be made up of digits, alphabetic letters, and underscores
        - These words have specific meanings, and are keywords in Python, so cannot be variable names

            1. and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield.
                - Print is not a keyword, but it would be a bad idea to assign a new value as it would lose the original meaning of the function.

        - If a statement continues over multiple lines you can use a backslash \ at the end of the line and continue writing.

- Input function
    - Used to get a keyboard input from the user as a string.

        1. name = input("What is your name? ")
        2. age = input("What is your age? ")
        3. print("I know that", name, "is", age, "years old!")
        4. What is your name? *Sam*
        5. What is your age? *17*
        6. I know that Sam is 17 years old!

- Print function can have multiple strings (variables + “literals”) separated by commas. Function has multiple arguments, and print combines them together, separating each one with a space.

Worksheet 2: Numerical Expressions

- Expressions
  - Make up the most of how we do everything. “Hel” + “lo”, 5.0 + 3.2 are examples of **simple expressions.**
  - Simple expressions are made up of two **operands** (things to be operated on), and one **operator** (performs the actions on the operands).
  - Expressions always return a value
- Types of Operands

Type	Description
`int`	For whole numbers eg: `-3`, `-5`, or `10`
`float`	For real numbers eg: `-3.0`, `0.5`, or `3.14159`
`bool`	The Boolean type. For storing `True` and `False` (only those two values; Booleans allow for no grey areas!).
`str` (= "string")	For chunks of text, e.g.: `"Hello, I study Python"`
`tuple`	For combinations of objects, e.g.: `(1, 2, 3)` or `(1.0, "hello", "frank")`
`list`	A more powerful way of storing lists of objects, e.g. `[1, 3, 4]` or `[1.0, "hello", "frank"]`
`dict`	We will see this later ... maybe you can guess what it does, e.g. `{"bob": 34, "frankenstein": 203}`

- Representing numbers: integers
  - Data type **int** is a type for all integers, such as 1 or -1832
  - They can be added, subtracted multiplied and divided

1. print(1523 - 329)
2. print(3929204 *2)