# Software Development 1
## Printing (Output)

TUDublin- Tallaght Campus

E. Costelloe

# Printing to the screen in Python

- A basic form of printing is as follows:

```
print("hello world")
```

- This prints the text “hello world” to the screen.
- Why do you think the parenthesis either side are not displayed?
- The print function(3.x)simply prints objects to the standard output stream which usually maps to the window where you started your Python program
- Python is case sensitive!! Print() is not the same as print()

# Printing

- The quotation marks represent a ***string*** (We will discuss these in detail later on)

```
print("hello world")

print('hello world')
```

- In Python you can use single or double quotation marks to represent a string.
- There is no strict convention on what to use, but we will use double quotation marks.

*(This is due in part to other language conventions/ syntax where a string only uses double quotation marks)*

## Printing

- What do you think the following output will be?

```
print("hello world")
print("hello world")
```

## Printing

- What do you think the following output will be?

```
print("hello world")
print("hello world")
```

```
Output:
    hello world
    hello world
```

- Why is this? And why not => hello worldhello world
- By default, print adds a linefeed at the end of the current output line
- i.e. At the end of each print, the function adds => \n

## Printing

- We can change the default end of a print method by adding an **end** argument:
- **end** is a string added at the end of the printed text, passing an empty string avoids dropping down to the next output line at the end of the printed text, the next **print** will keep adding to the end of the current output line

```
print("hello world", end="")
print("hello world")
```

```
Output:
    hello worldhello world
```

```
print("hello world", end=".")
print("hello world")
```

```
Output:
    hello world.hello world
```

- This first example removes the default end of line \n.
- The second example changes it to what we want, for example a full stop.
- With no arguments at all, the print function simply prints a newline character to the standard output stream, which usually displays a blank line.

```
print()
```

## Escape sequences

Backslashes are used to introduce special character codings known as *escape sequences,* escape sequences let us embed characters in strings that cannot be easily typed on a keyboard.

The character \, and one or more characters following it in the string literal, are replaced with a single character in the resulting string object.

For example, here is a five-character string that embeds a newline and a tab:

```
print(“a\nb\tc”)
```

The two characters \n stand for a single character—the binary value of **the newline character** in your character set (in ASCII, character code 10).

Similarly, the sequence \t is replaced with the tab character.

***Output:***

```
a
b	c
```

## String backslash characters

| Escape | Meaning |
|---|---|
| \\ | Backslash (stores one \) |
| \‘ | Single quote (stores ') |
| \" | Double quote (stores ") |
| \a | Bell |
| \n | Newline (linefeed) |
| \t | Horizontal tab |

# Printing

- We can use these special characters (Escape Sequences)

```
print("hello world\nhello world")
```

- The interpreter ignores the first character after a “\”

```
Output:
    hello world
    hello world
```

# Printing

- We can use these escape characters and special characters to format outputs:

```
print("*************\n*\t\t\t*\n*\t\t\t*\n*************")
```

**Output:**

?

**Output:**

```
*************
*           *
*           *
*************
```

# Printing

- The print method will print blank spaces (these are also characters):

```
print("  *************\n*\t\t\t*\n*\t\t\t*\n*************")
```

**Output:**

```
      *************
    *           *
    *           *
    *************
```

## Printing

- We can also print numbers, Python recognises the difference (no quotes)

```
print(5)
print(5+2)
```

```
Output:
    5
    7
```

- The second line, the Interpreter completes the mathematical operation first.
- We will look into this more in the next section (variables).

- What if I wanted to print a number and a sentence (assuming it is a number)
- print(**"5 + 2 ="**, 5+2) **Outputs** → 5 + 2 = 7

## Printing

- We can also print numbers, Python recognises the difference (no quotes)

```
print(“Answer: " + str(5))
print(“Answer:", 5)
print(“Answer {0}".format(5))
```

```
Output:
    Answer: 5
    Answer: 5
    Answer: 5
```

- The convention generally used is the second line. The third line has its advantage with regards to security, this will be discussed at a later stage.

# Comments:

- In industry programs will be divided up into segments (divide and conquer methodology)
- This means that you will have to develop with several other developers.
- Updates and legacy will also require you to work with other developers.
- Developers have their own style and algorithms.
- We need to be able to understand it!

# Comments:

- Python allows us to comment in two ways:

```
"""
This is a multi line comment
"""

print("hello")                # This is a single line comment


# This is also a single line comment
print("world")
```

- Comments are ignored by the interpreter and are for documentation purposes only
- It is your preference / style, but you should stick to one type.
- For this course all Python files must have at the top as a comment:
  - *Student Number*
  - *A line describing the purpose of the program*

## Printing

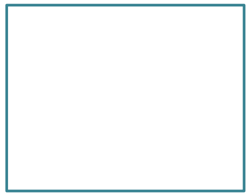
- There is more to printing:
  - Printing Variables (next section)
  - Printing Lists
  - Printing Objects

*We will develop these more as we reach each topic.*

*Mastering printing will help in UI design, and usability.*

# In class

- Try print the following (hint you can use stars to draw the shapes):
- Your name and address on separate lines
- Square



- Triangle