DOCUMENTING

An Introduction to Computer Science

Let's learn about Documenting.

Documenting Code

1. For sharing code
2. For re-reading old code
3. To organize our current code

Programmers write documentation in order to explain what a function does.
This is useful not only for sharing code with other programmers, but when we come back to code we wrote later.
To document code, you create comments that are ignored when the code is executed.
These comments are for the benefit of humans, not the computer.
Python has two ways of writing comments.

Single-line Comment

```
# This line is ignored
a = 0 # The rest of this line is ignored
b = 0
```

"Octothorpe"
"Pound"
"Hash"
#

To make a single-line comment in Python, you should use the hash symbol (also known as "pound" or "octothorpe", written as #).
Everything after the hash symbol on the same line is completely ignored by your program.

Multi-line Comment

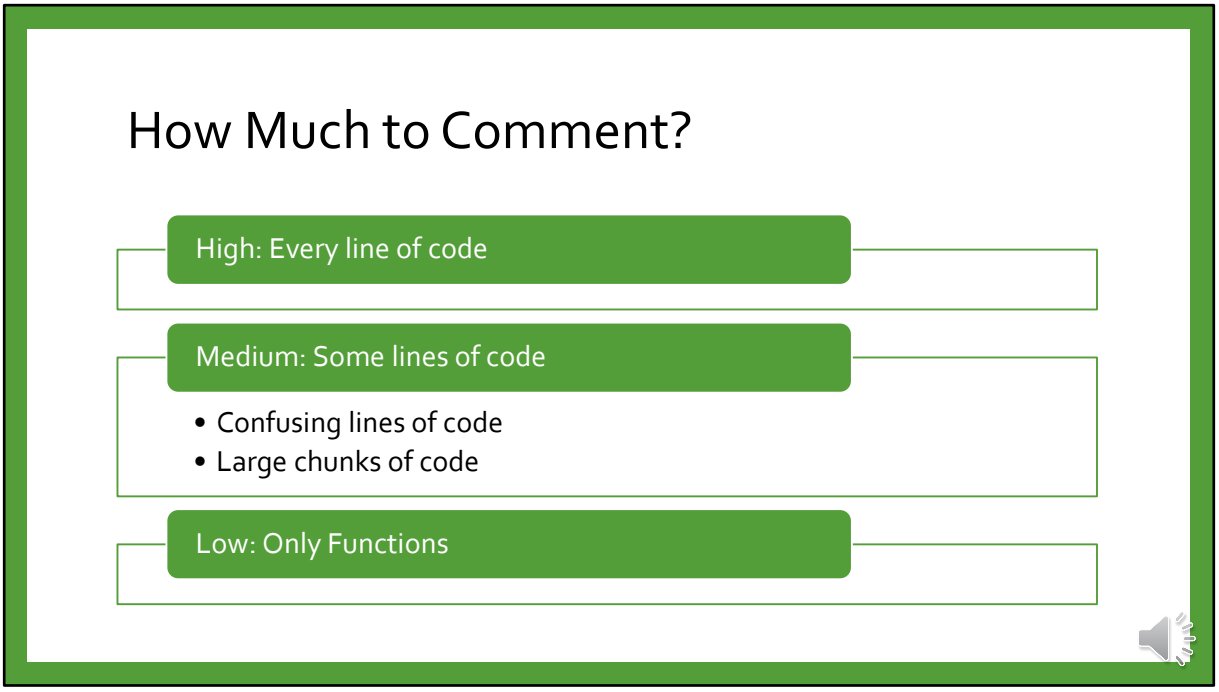
```
'''
All of this code has been commented out!
Or rather, turned into a string.
age = 0
name = "Klaus Bart"
'''


def my_function(x:int, y:int)-> int:
'''
The start of this string
needed to be indented!
'''
```

Safely commented out

IndentationError!

To write multi-line comments in Python, you actually create a triple-quoted string. When Python evaluates this string, the value is unused, so it is safely ignored by the computer.
However, unlike the single-line comment, that means that the start of a multi-line comment must respect indentation rules as shown here.



There are many opinions on when to write comments, and no "correct answer".
Some people believe you need to document every single line.
Some people believe you should only document functions and programs as a whole.
You will need to find your own balance - for example, rather than documenting every line, you might only document lines that are particularly confusing, or document a chunk of code.

Documenting functions

```
def fix_capitalization(title:str)->str:
    '''
    This function capitalizes the first letter of each
    word in the title, correctly ignoring prepositions.

    Args:
        title (str): A book or movie title
            that we want to fix.
    Returns:
        str: The corrected title
    '''
```

1) What it does

2) Argument types

3) Argument description

4) Return type

5) Return description

Many people agree that, at a minimum, you should document all your functions.Here is the convention that you will need to follow.Use a triple-quoted string as the first line in the definition of the function.Begin with a quick description of what the function does.Leave a blank line, and then type "Args:".Indent the next line, and then type the name of the first parameter followed by its type in parentheses. On the same line, write a colon and then describe the parameter. If you need to continue onto the next line, make sure it is indented past the previous line.You should repeat this for all the parameters of your function.Finally, write "Returns:" and then on the next line write the name of the return type, followed by a colon and a description of the returned value.