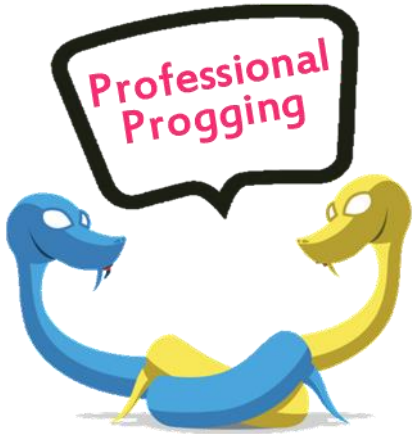
COMP1811
Paradigms of Programming



Adapted from https://www.cleanpng.com

Professional Programming 1

Dr. Yasmine Arafa

Professional Programming

- One of the first steps to professional programming is consistency:
  - in terms of the style of writing, as well as
  - conforming to the conventions of the language you are coding in.
- Coding convention are a set of f coding rules, guidelines, and best practices that determine the programming style, procedures, and methods for a programming language.

Why are coding conventions important?

- Without the coding conventions, individual programmers will come up with their own style. It will not be easy to maintain and debug the code in the near future.

- Benefits of a convention:
  - gives a uniform appearance to the codes written by programmers,
  - improves readability, and maintainability of the code,
  - it helps in code reuse and helps to detect error easily,
  - it promotes sound programming practices and increases efficiency of the programmers.

Common Aspects of a Coding Convention

- Naming convention
  - the naming convention is how your variables, functions, classes, etc. should be named.
- Code formatting and indentation
  - the code should be written in a standardised format with correct indentation.
- Commenting and documenting
  - this helps the reviewer of your code or someone who needs to update it better understand the code and its methods/functions and logic.

Common Aspects of a Coding Convention
Cntd.

- Classes, methods and functions:
  - this specifies how classes, methods and functions should behave.
- File and folder naming and organisation
  - this is how your files and packages should be named and structured.
- Testing:
  - this specifies which approach and tools should be used to test the code.

Python Variable Naming
some rules

- **Python is case-sensitive:**
  - e.g. `module_code`, `module_Code`, and `Module_Code` are three different names!
- **Variable names can only contain alphanumeric characters, and `_`:**
  - must start with a letter,
  - names should be meaningful (e.g. `age` ✓, `a` ✗),
  - avoid names starting with `_` (e.g. `_code` - it is legal but best to avoid ✓),
  - white spaces are not allowed (`module code = 1811` ✗),
  - special characters (%, $, £, etc.) are not allowed (`%total = 58` ✗),
  - digits at start are not allowed (`10ml_price = 12` ✗).
- **Variable names must not be a reserved or keyword word such as:**
  `True`, `False`, etc. (`True = "Good result"` ✗)

Reserved Words (Keywords) in Python
avoid the use of keyword as a variable name

- Keywords are have predefined meaning and syntax in the language, and
  - cannot be used as identifiers for variables, functions, classes, methods, etc.

- Python keywords:

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	raise
break	except	in		

Common Variable Naming Conventions

- Variable names (identifiers) should describe the purpose of the variable:
  - they should be meaningful within context, and
  - should avoid names like: `var1`, `button1`, etc.
  - names such as `save_btn`, or `exit_btn` are better names.
- Variable names should start with a lowercase letter:
  - beginning with an uppercase is legal BUT can be confused with a class name.
- With single word variable names, all characters are lowercase
  - e.g.: `grades, name`
- Multiple words are separated with an underscore
  - e.g.: `module_code`, `savings_acct`, `current_acct`

Common Variable Naming Conventions
Cntd.

- CamelCase name can be used BUT this is not strictly Python Style
  - camelCase ("dromedary case") is where Multiple words are separated by capitalising the first letter of each word except for the first word,
  - that is, the first letter of each word in a compound word is capitalised;
  - variables should have lowerCamelCase names (ie the first compound word is lowercase) e.g. `firstName = "James"`
- Constant names are typically all capital
  - e.g.: `MAX_SIZE, PI, VALUE_ADDED_TAX`
- Details of Python best practice and naming convention can be found at PEP8.

Comments

- Comments are text notes about the code and are used for
  - for providing explanatory notes about the code:
    - *this helps other programmers understand your code and its purpose, and*
    - *also helps you remember your own code if you've left it for a while;*
  - for keeping versions of parts of the code that are not working properly but you want to keep.
- A comment is any statement in Python that begins with a #
  - the Python interpreter ignores comments (any text beginning with #)
- Single-line comments denoted by #
  - lines beginning with #, comment the entire line, and
  - partial lines beginning with #, comment the line from the # to the end of line.

Comments
professional programming

- It is good practice to include in all your program files:
  - a **header comment** (such as in the previous slide) that includes details of who and when the code was created, its version and a brief description of its purpose; and
  - **in-line comments** that describe line or blocks of code.
- Comments should express information that cannot be expressed in the code – do not restate code.

Multi-line Comments

- Multi-line comments can be a collection of separate lines, each starting with #:

```
##
# Author: Some Programmer
# Date created: 04/10/21
# Date updated: no updates
# Version: 1.0
# Description: Demonstrates
#      the use of comments.
##
```

Multi-line Comments
cntd.

- Alternatively, define multi-line comments by enclosing them between three quotes at the beginning and end of the lines:

```
"""
  Multi-line documentation
  more text
  and more text.
"""
```

  - this technique doesn't create *true* comments - it simply inserts a text constant that does nothing,
  - be careful where you place them in the code - they could turn into **docstrings** and what was intended as a simple comment will become associated with an object and take up memory. They are better avoided.

Some Python Fundamentals
whitespace matters!

- Whitespace is significant in Python.
  - adding whitespaces to the beginning of a line of code (indentation) in the main body of the code will cause error

- Example:

```
print("Hello World!")
    print("AND a special hello to COMP1811...")
```

causes error!

Common Errors

- General rule of thumb: there will always be some errors in the code!!!
  - that's fine!
  - few programs will run successfully from the first time.
- Common errors:
  - **Name Errors:**
    misspelt names or misused upper and lowercase characters;
  - **Trying to use a variable before it is defined will cause syntax error:**

```
print(x)
x = 10
```

causes error!

Common Errors
Cntd.

- Common errors, *Cntd.*:
  - **Indentation Errors:**
    spaces or tabs used at the beginning of a line where they shouldn't be;
  - **EoF Errors:**
    End of File error, which means there is an open parenthesis somewhere without a matching closing parenthesis;
  - **EoL Errors:**
    End of Line error, which means either there is a string with open quotes without closing quotes, or a string extends past one line without using triple quotes.