Coding Style

Code

- Developers **read code** much more than they **write code**
- In the long run, there is no "your code" and "my code"
- Often, the code is the only documentation of the design
- Common coding style/coding standard is necessary

Essential Parts of Coding Style

- Indentation
- Newlines
- Whitespace
- Comments/DocStrings
- Naming conventions

Characteristics of a Coding Style

- Consistency - does our code adhere to regular expected stylistic patterns
- Scalability - does the chosen style work as the size of the code increases
- Maintainability - does the style support clarity in the code such that making changes or bug fixes can be done efficiently

Inconsistency

```
1
2   def named_greeting(name):
3       print(f"Hello {name}!")
4
5
6   def shoutNamedGreeting(name):
7       print(f"Hello {name.upper()}")
8
9   def main():
10      Firstperson="Maya"
11      second_person ="Addison"
12      named_greeting(Firstperson)
13      shoutNamedGreeting(second_person)
14
15
16
17  if __name__ == "__main__":
18
19
20      main()
21
22
23
```

Scalability

```
/* NOW INITIALIZATION TO FILL DUMMY LEVELS, TOP LEVEL, AND UNUSED PART OF TOP*/
/* LEVEL AS REQUIRED.                                                        */

(7) INIT: MINF= (48) '0'B;
          PINF= (48) '1'B;

          DO L=   0 TO 4094;  T(L) = MINF;        END;
          DO L=   0 TO 2499;  T(L+4095) = V(L);   END;
          DO L=6595 TO 8190;  T(L) = PINF;        END;

(8) K0:   K = -1;                                                    (10)
    K1:   I = 0;                  /*          (11)                  <------| */
    K3:   J = 2*I+1;              /*SET J TO SCAN BRANCHES FROM NODE I.  <-----|| */
    K7:   IF T(J) <= T(J+1)       /*PICK SMALLER BRANCH                  __>___|| */
            THEN                  /*                                        ||| */
          (9) DO;      (12)       /*                                        ||| */
    K11:      T(I) = T(J);        /*REPLACE                                 ||| */
    K13:      IF T(I) = PINF THEN GO TO K16; /*IF INFINITY, REPLACEMENT_+∞___||| */
                                  /* IS FINISHED                            |||| */
    K12:      I = J;              /*SET INDEX FOR HIGHER LEVEL              |||| */
            END;                  /*                                        |||| */
          ELSE                    /*                                   <---+-|| */
            DO;                   /*                                       | || */
    K11A:     T(I) = T(J+1);      /*                                       | || */
    K13A:     IF T(I) = PINF THEN GO TO K16;       /*                 _+∞___| || */
    K12A:     I = J+1;            /*                                       | || */
            END;                  /*                                     <-| || */
    K14:  IF 2*I < 8191 THEN GO TO K3;   /*GO BACK IF NOT ON TOP LEVEL  ----+-|| */
    K15:  T(I) = PINF;            /*IF TOP LEVEL, FILL WITH INFINITY        |  | */
    K16:  IF T(0) = PINF THEN RETURN;    /*TEST END OF SORT             <---|  | */
    K17:  IF T(0) = MINF THEN GO TO K1;  /*FLUSH OUT INITIAL DUMMIES     -∞____| */
    K18: K = K+1;                        /*STEP STORAGE INDEX                | */
         V(K) = T(0);  GO TO K1; (12)    /*STORE OUTPUT ITEM                 | */
END QLTSRT7;
```

Scalability

```
    auto value=*begin; // pivot value is the first element

    auto left = begin;
    auto right= std::prev(end);
    while (std::distance(left,right)>0) {

            // move left-to-right while value is greater than elements
            while(value>= *left && std::distance(left, end) > 0){
               if (std::next(left) == end)
                        break;

                    left=std::next(left);
              }

            // move right-to-left while value is less than elements
            while (value <*right)
            {
                    right= std::prev(right);
            }

    // exchange so elements less than value are on the left
    // and elements greater than value are on the right
            std::swap(*left, *right);
    }


    std::swap(*begin,*right); // exchange pivot and final location

// post-condition
assert(std::all_of(begin, right, [right](int n){ return n <= *right; }));
assert(std::all_of(right, end,   [right](int n){ return n >= *right; }));
```

Difficult to Maintain

```
/**********************************************************
 *                                                        *
 * myCoolFunction                                         *
 *                                                        *
 * This is my cool function. Isn't it cool?               *
 *                                                        *
 **********************************************************/

 /*
  * My Cool Function
  *
  * This is my cool function. Isn't it cool?
  *
  */
```

Difficult to Maintain

```
① //QLT4 JOB ...

② QLTSRT7: PROCEDURE (V);

   /*********************************************************************/
③  /*A SORT SUBROUTINE FOR 2500 6-BYTE FIELDS, PASSED AS THE VECTOR V.  A     */
   /*SEPARATELY COMPILED, NOT-MAIN PROCEDURE, WHICH MUST USE AUTOMATIC CORE   */
   /*ALLOCATION.                                                              */
   /*                                                                         */
④  /*THE SORT ALGORITHM FOLLOWS BROOKS AND IVERSON, AUTOMATIC DATA PROCESSING,*/
   /*PROGRAM 7.23, P. 350.  THAT ALGORITHM IS REVISED AS FOLLOWS:             */
⑤  /*  STEPS 2-12 ARE SIMPLIFIED FOR M=2.                                     */
   /*  STEP 18 IS EXPANDED TO HANDLE EXPLICIT INDEXING OF THE OUTPUT VECTOR.  */
   /*  THE WHOLE FIELD IS USED AS THE SORT KEY.                               */
   /*  MINUS INFINITY IS REPRESENTED BY ZEROS.                                */
   /*  PLUS INFINITY IS REPRESENTED BY ONES.                                  */
   /*  THE STATEMENT NUMBERS IN PROG. 7.23 ARE REFLECTED IN THE STATEMENT     */
   /*    LABELS OF THIS PROGRAM.                                              */
   /*  AN IF-THEN-ELSE CONSTRUCTION REQUIRES REPETITION OF A FEW LINES.       */
   /*                                                                         */
   /*TO CHANGE THE DIMENSION OF THE VECTOR TO BE SORTED, ALWAYS CHANGE THE    */
   /*INITIALIZATION OF T.  IF THE SIZE EXCEEDS 4096, CHANGE THE SIZE OF T,TOO.*/
   /*A MORE GENERAL VERSION WOULD PARAMETERIZE THE DIMENSION OF V.           */
   /*                                                                         */
   /*THE PASSED INPUT VECTOR IS REPLACED BY THE REORDERED OUTPUT VECTOR.     */
   /*********************************************************************/

⑥ /* LEGEND  (ZERO-ORIGIN INDEXING)                                         */

  DECLARE
    (H,                  /*INDEX FOR INITIALIZING T                        */
     I,                  /*INDEX OF ITEM TO BE REPLACED                    */
     J,                  /*INITIAL INDEX OF BRANCHES FROM NODE I           */
     K) BINARY FIXED,    /*INDEX IN OUTPUT VECTOR                          */

    (MINF,               /*MINUS INFINITY                                  */
     PINF) BIT (48),     /*PLUS INFINITY                                   */

     V (*)   BIT (*),    /*PASSED VECTOR TO BE SORTED AND RETURNED         */

     T (0:8190) BIT (48); /*WORKSPACE CONSISTING OF VECTOR TO BE SORTED, FILLED*/
                          /*OUT WITH INFINITIES, PRECEDED BY LOWER LEVELS   */
                          /*FILLED UP WITH MINUS INFINITIES                 */
```

Problems with a Coding Standard

- Lack of formal training
- Programming language differences
- Difficult to formally define/check/correct
- Difficult to maintain
- Lots of corner cases
- Preference arguments
- “bike shed painting”

- NOTE: Python helps to alleviate some of these issues with PEP 8, but it still allows enough room for judgement calls

Indentation and Whitespace

- Indentation must be consistent
- Indentation should appear consistent, even when moved out of the IDE, e.g., Gists, web examples etc.
- Indentation is based on "flow of control", not importance/difficulty

Indentation Composition

- How much to indent for each level? 2? 4? 8?
  - PEP 8 says 4
- What is the indentation made up of? spaces? tabs? tabs/spaces?
  - PEP 8 says spaces
- Problem: Most of the time, can't tell by looking
- Problem: Lack of understanding of what a tab (and tab stop) is.
- Confusion: Developers who use spaces to indent often have the editor expand tabs to spaces

Inconsistent Indentation

- Don't mix tabs and spaces for the indentation on different lines or the same line (with Python this is actually a problem for the interpreter)
- Many of the advantages of tabs have been replaced by IDE features.
  - Just use spaces llvm coding standard, especially with code that may get used in multiple projects
- If you are going to use tabs, only use them for flow-of-control indentation, and not to line things up
- Indentation level: 4 is PEP 8 but Google is 2…why might that be?

Inconsistent Indentation

```
1    # PEP 8
2    foo = long_function_name(var_one, var_two,
3                             var_three, var_four)
4
```

```
# PEP 8
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

Newlines

- Single newlines are sufficient to break up related code content
  - PEP 8 even states that each Python file must end with one blank newline
- Multiple newlines are not needed, except in the case of function definitions
  - PEP 8 dictates two newlines before each function definition
- Again, consistency

Whitespace

```
area = width * height;
```

- Spaces before parentheses
  - For control flow statements
  - **NOT** for functions
    - print(“hello”) not print (“hello”)

- Space around operators
  - PEP 8 States: assignment, comparison, and Boolean operators
  - Arithmetic operators is officially left to person preference (BUT BE CONSISTENT)

Comments

- Code is written in paragraphs, chunks/hunks of code
- Comments appear before, indented the same, space after ”#"
- Prefer line comments
- Use of DocStrings for File Headings/Classes/Functions
  - There are multiple styles of DocString markup
    - I use Google, but ReST is also popular

Summary

- Consistency, Consistency, Consistency
- Coding styles often reflect a compromise
- With existing code, adapt to the coding style of the existing code/project
- Be open to updating your coding style over time
- There are much more important things in a project than minor coding style decisions