Exploring Software Metrics with Python

Samantha M. Hipple

School of Technology and Computing, City University of Seattle

CS 504: Software Engineering

Kendra Schraml

June 7, 2022

Exploring Software Metrics with Python

Software metrics are data collected based on the analysis of source code that are used to quantify the quality of a software program's overall design. Good software design is expected to be consistent and complete. A well-designed program should be as easy to understand, modify, reuse, test, integrate, and code as possible. The degree of simplicity within a system can be measured by two characteristics: cohesion and coupling. Cohesion focuses on maintaining a single purpose at the component, class, and modular level. Coupling is the interaction and interdependence between two software units. Good software designs focus on having high degrees of cohesion and low degrees of coupling (Tsui et al., 2014).

Unfortunately, when searching for Python modules that have been designed to analyze programs for both cohesion and coupling metrics, there are only a few outdated libraries that exist at this point in time. However, there are plenty of libraries available that focus on legacy characterizations of design attributes such as Halstead's Complexity Metric and McCabe's Cyclomatic Complexity (CC), as well as basic raw metrics such as lines of code, comment usage, and proper lexical conventions analyses. The rest of this report tests three commonly used software metric generating Python modules: **pylama**, **pylint**, and **radon** against an example Tic Tac Toe program that is preparing for its second release. From there we are able to dive into how the generated metrics can be used to improve our program and our programming habits.

Getting Started with Pylama

Pylama is a code audit tool that wraps nine software metric tools in one: **pycodestyle**, **pydocstyle**, **pyflakes**, **mccabe**, **pylint**, **radon**, **eradicate**, **mypy**, and **vulture**. We began our software metrics analysis by running all of the metric tools available in Pylama against the four modules of our Tic Tac Toe program: *main.py*, *ttt_menus.py*, *tic_tac_toe.py*, and *test.py*. and saved the results to *report.txt* - which can be reviewed in **Appendix A**.

This initial Pylama report is sorted first by our example program modules, then by the line number of the code being flagged. The tool used to generate each flag is recorded at the end of each message as well. In order to break down these results further, we also generated individual reports for each of the metric tools available in Pylama. In doing this, we discovered that the **vulture** module, which is designed to find unused and untested code; and the **eradicate** module, which is designed to remove all dead code (commented-out code) from your source code; return no analysis for our example program (nor make any changes). Additionally, the **mypy** module, which is a static type-checker that requires your code to be written using static-typing, is unable to analyze our Tic Tac Toe program as well, due to its standard of dynamic-typing throughout.

Further Metrics Exploration with Radon

Next, we used the **radon** module itself to provide a more in-depth look at our source code using its four software metric tools: raw metrics, cyclomatic complexity, Halstead complexity metrics, and maintainability index (MI) (Lacchia, 2020). This section will break down each of the four tools initial results by providing a short summary of what the tool does, visuals of the quantitative metrics and statistics each tool generates, and an explanation of what each dataset represents and why it may, or may not, be important for the overall improvement of our example program’s design.

Raw Metrics

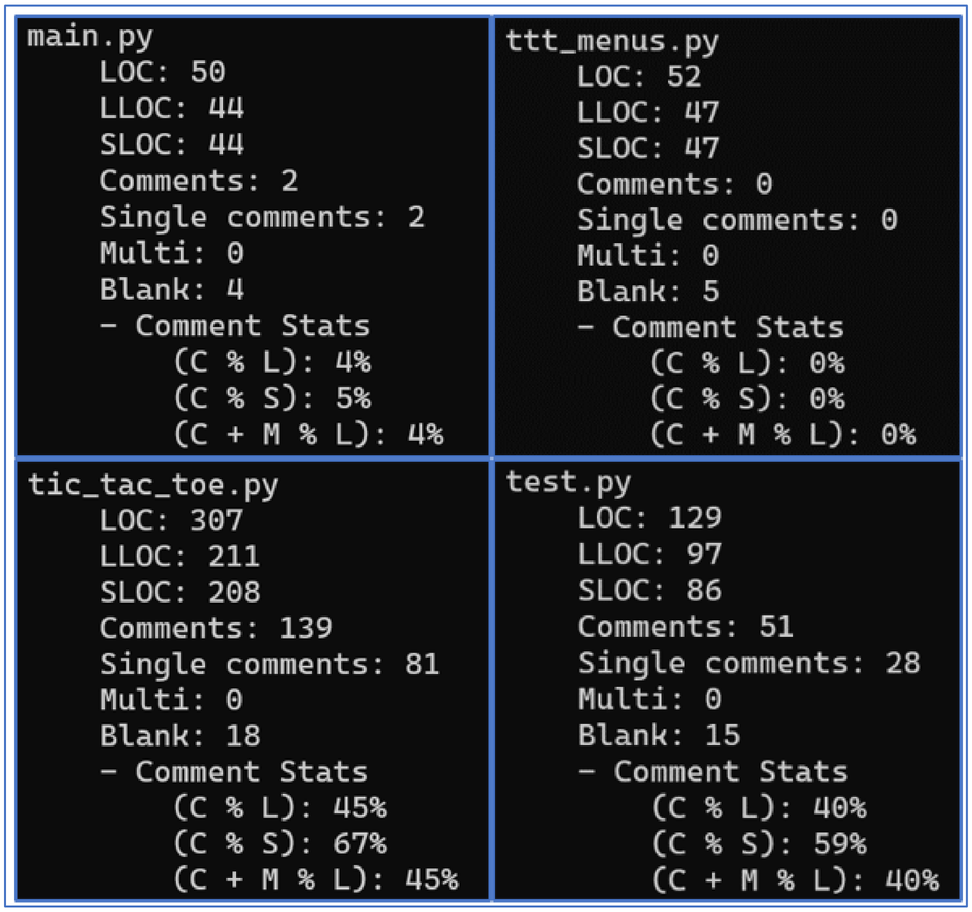
Radon uses the following data within its raw metrics reports: total lines of code (**LOC**) – which is equivalent to (**SLOC + Multi + Single comments + Blank**); logical lines of code (**LLOC**) – which includes all code instructions; and source lines of code (**SLOC**) – which include all lines of code, minus any comments and blank lines. Next, Radon records: the number of comment lines (**Comments**); the total number of comments that are not multi-line strings (**Single comments**); the total number of comments that are multi-line strings (**Multi**); and the total number of blank lines (**Blank**). Lastly, Radon uses the above metrics to compute three comment statistics: percentage of comments per total lines of code (**C % L**); percentage of total comments per lines of source code (**C % S**); and percentage of comments and multi-line strings per total lines of code (**M + C % L**) (Lacchia, 2020).

Looking at the results displayed in **Figure 1** on the next page, we can see that our tic_tac_toe module contains the most comments per lines of code at 45%. Additionally, we can see that our ttt_menus module contains zero comments – which has the potential to make the code more difficult to maintain or be understood by others. Our main module could potentially use more comments as well, while our test module appears well documented.

These raw metrics are able to establish common patterns that a programmer may not realize in their coding. One pattern in our example program is that of commenting larger modules well, while essentially ignoring comment usage in the simpler, shorter modules. Fixing this unintentional oversight would improve this programmer’s ability to share, maintain, and refractor their code more readily.

Figure 1.

Radon Raw Metrics



```
main.py
    LOC: 50
    LLOC: 44
    SLOC: 44
    Comments: 2
    Single comments: 2
    Multi: 0
    Blank: 4
    - Comment Stats
        (C % L): 4%
        (C % S): 5%
        (C + M % L): 4%
```

```
ttt_menus.py
    LOC: 52
    LLOC: 47
    SLOC: 47
    Comments: 0
    Single comments: 0
    Multi: 0
    Blank: 5
    - Comment Stats
        (C % L): 0%
        (C % S): 0%
        (C + M % L): 0%
```

```
tic_tac_toe.py
    LOC: 307
    LLOC: 211
    SLOC: 208
    Comments: 139
    Single comments: 81
    Multi: 0
    Blank: 18
    - Comment Stats
        (C % L): 45%
        (C % S): 67%
        (C + M % L): 45%
```

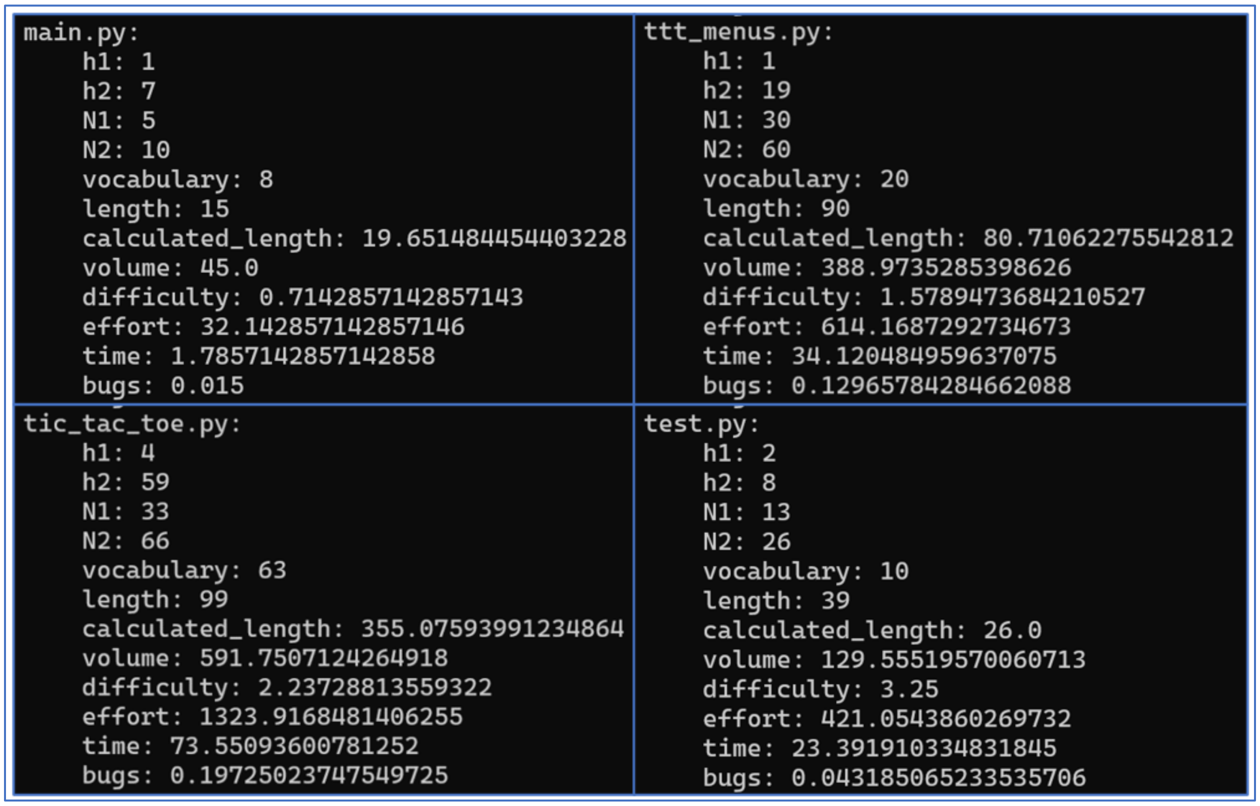
```
test.py
    LOC: 129
    LLOC: 97
    SLOC: 86
    Comments: 51
    Single comments: 28
    Multi: 0
    Blank: 15
    - Comment Stats
        (C % L): 40%
        (C % S): 59%
        (C + M % L): 40%
```

Halstead Metrics

The Halstead Complexity Metric uses four metric units when analyzing a program's source code: (1) number of *distinct operators*, (2) number of *distinct operands*, (3) total number of *operators*, and (4) total number of *operands*. Halstead then applied these values to define two more measurements: (1) unique program vocabulary (sum of unique operands and operators), and (2) program length (sum of all operands and operators) (Tsui et al., 2014). From there, Radon takes the generated Halstead metrics and creates six statistics for each module: (1)

Figure 2.

Radon Halstead's Complexity Metrics



```
main.py:
    h1: 1
    h2: 7
    N1: 5
    N2: 10
    vocabulary: 8
    length: 15
    calculated_length: 19.651484454403228
    volume: 45.0
    difficulty: 0.7142857142857143
    effort: 32.142857142857146
    time: 1.7857142857142858
    bugs: 0.015
```

```
ttt_menus.py:
    h1: 1
    h2: 19
    N1: 30
    N2: 60
    vocabulary: 20
    length: 90
    calculated_length: 80.71062275542812
    volume: 388.9735285398626
    difficulty: 1.5789473684210527
    effort: 614.1687292734673
    time: 34.120484959637075
    bugs: 0.12965784284662088
```

```
tic_tac_toe.py:
    h1: 4
    h2: 59
    N1: 33
    N2: 66
    vocabulary: 63
    length: 99
    calculated_length: 355.07593991234864
    volume: 591.7507124264918
    difficulty: 2.23728813559322
    effort: 1323.9168481406255
    time: 73.55093600781252
    bugs: 0.19725023747549725
```

```
test.py:
    h1: 2
    h2: 8
    N1: 13
    N2: 26
    vocabulary: 10
    length: 39
    calculated_length: 26.0
    volume: 129.55519570060713
    difficulty: 3.25
    effort: 421.0543860269732
    time: 23.391910334831845
    bugs: 0.043185065233535706
```

calculated length, (2) volume, (3) difficulty, (4) effort, (5) time, and (6) bugs as shown in **Figure 2** above.

The **calculated_length** statistic represents Halstead's first hypothesis of software science - that the length of a well-structured program can be determined solely using the number of unique operators and operands in this formula: **N^ = $n_1 \log_2 n_1 + n_2 \log_2 n_2$**. The idea is to compare the actual and estimated lengths as a way of measuring whether or not your program is well-structured, however, this particular hypothesis is often inaccurate for small programs that are considered individually (Mall, 2018). Our example program's tic_tac_toe module has the greatest variance between its actual length and its estimated length. This is also our largest module and the most likely to benefit from modifications based on this report. The other three modules all have estimated lengths within +-10 of their actual length.

The **volume** statistic represents the minimum number of bits required to encode the program while compensating for the programming language used. To calculate the volume, Halstead uses the expression **$V = N*\log_2(ŋ)$** where **N** is the program's length and **ŋ** is the program's vocabulary (Mall, 2018). Looking at *main.py*, we see the volume is 45 = 15 * log2(8). This means that our main module requires at least 45 binary bits to build.

The **difficulty** statistic represents how error prone our program is proportionate to the number of unique operators in the program. This stat is calculated using the expression **D = (n1 / 2) * (N2 / n2)**. This metric is then applied to calculate Halstead's **effort** statistic (**E = D * V**). According to IBM (2021), "[t]he lower the value of this metric, the simpler the program would be to change". That is, the less mental effort it should take to develop or maintain the program. We can see in our Tic Tac Toe program's results above, that our main module has a small effort value of approximately **32**. However, every other module is well above **400** in value, with our tic_tac_toe module exceeding a value of **1300**.

The **time** statistic estimates the amount of time it takes to implement the program. The equation is **T = E / S** where **E** is the effort metric from the previous step and S is the *Stroud Number*. According to VanSuetendael & Elwell (1991):

The Stroud Number is a value the implementor arbitrarily chooses from a range of 5 <= S <= 20 per second. For concentrating programmers who are fluent in the language used and who are provided with nonprocedural problem statements, a value of **18** discriminations per second has been used successfully (Funami and Halstead 1976). The range is obtained from psychological studies that determined the total number of elementary mental discriminations a person could perform per second.

In reference to our Tic Tac Toe program's results in Figure 2, we see that our test module should take less time for an experienced programmer to write than our rather simple ttt_menus module based on this evaluation. In reality, the test module took much longer – however, there were no pre-defined problem statements, and the programmer did not have experience writing test cases at the time. It makes sense, based on the complexity of the methods within those two modules, that the test module could be created quickly by an experience Python programmer based on the simplicity of the solutions compared to our ttt_menus module methods – which present more of a challenge when attempting to describe them in plain English to the point of replication.

Finally, the number of potential **bugs** within a program can be estimated using the following equation: $\mathbf{B = V / E_0}$ where **V** is the program volume and $\mathbf{E_0}$ is the average number of discriminations a programmer is likely to make per error. Halstead originally determined, through experimentation, that an experienced programmer codes approximately one bug per 3,200 discriminations (VanSuetendael & Elwell, 1991). Radon uses a value of 3000 for this expression (Lacchia, 2020). This particular metric does not compute well when used against a program the size of our Tic Tac Toe game due to our program volume being far less than the 3000 estimated discriminations it is divided by, causing all of our results to equate to less than one bug per module. "This equation gives meaningful results for real-life applications where the volume is much greater than the examples provided here" (VanSuetendael & Elwell, 1991).

McCabe's Cyclomatic Complexity

The CC metric was designed based on McCabe's observation that program quality is directly related to the number of branches in source code. The measure is calculated from a control flow diagram representation of that program using this expression: **CC = E – N + 2p**, where **E** is the number of edges of the graph, **N** is the number of nodes, and **p** is the number of connected components (*usually p=1*). Another way to express McCabe's CC metric is *CC = number of binary decisions + 1* or *CC = number of closed regions + 1* (Tsui et al., 2014). In **Table 1** below, we see the effects of particular constructs on our CC metric:

Table 1.

Constructs and their effects on the cyclomatic complexity metric (Lacchia, 2020)

Constructs	CC Effect	Reasoning
`if`	+1	An **if** statement is a single decision.
`elif`	+1	The **elif** statement adds another decision.
`else`	+0	The **else** statement does not cause a new decision – the decision is in the **if** statement.
`for`	+1	There is a decision at the start of the **for** loop.
`while`	+1	There is a decision at the **while** statement.
`except`	+1	Each **except** branch adds a new conditional path of execution.
`finally`	+0	The **finally** block is unconditionally executed.
`with`	+1	The **with** statement roughly corresponds to a **try**/**except** block.
`assert`	+1	The **assert** statement internally, roughly equals a conditional statement.
`comprehension`	+1	A list/set/dictionary comprehension of generator expression is equivalent to a **for** loop.
`boolean operator`	+1	Every boolean operator (**and**, **or**) adds a decision point.

Table 2.

Radon’s CC Ranking Method (Lacchia, 2020)

CC Score	Rank	Risk	
1-5	A	*low*	a simple block
6-10	B	*low*	a well-structured and stable block
11-20	C	*moderate*	a slightly complex block
21-30	D	*more than moderate*	a more complex block
31-40	E	*high*	a complex block, alarming
41+	F	*very high*	an error-prone, unstable block

When running Radon’s CC feature against our Tic Tac Toe program, the default results generate with a letter grade after each class, method, or function within each of our modules. Table 2 above explains each ranks score range and risk level. The resulting CC metrics for each module in our program can be found in **Appendix B**. All classes and methods within our ttt_menus and test modules were given **A** rankings. Both of the functions in our main module were rated rank **B**. Both of these functions use **while** loops with **try**/**except** blocks for input error catching, and **if** statements for user selections. According to Table 2, this means that our

Figure 3.

Pylama’s CC Results

```
Namespace(paths=['C:/Users/Saman/Desktop/CS 504 (software engineering)/05. Independent Project/CS504-Project-Samantha-
Hipple/Submission 4/TicTacToe v2.0'], verbose=True, options=None, linters=['mccabe'], from_stdin=False, concurrent=False,
format='pycodestyle', abspath=False, max_line_length=100, select=set(), ignore=set(), skip=[], sort=None, report='report_mccabe.txt',
hook=False, max_complexity=10, pydocstyle_convention=None, pylint_confidence=None, radon_no_assert=False,
radon_show_closures=False, file_params={}, linters_params={})

Run [mccabe] main.py
Run [mccabe] test.py
Run [mccabe] tic_tac_toe.py
Run [mccabe] ttt_menus.py

tic_tac_toe.py:120:5 C901 'TicTacToe.is_winner' is too complex (15) [mccabe]
```

main module functions consist of well-structured, stable blocks of code.

Our tic_tac_toe module had the most variety in its rankings with the class overall receiving an **A**, the two gameplay loop methods both receiving **B**s, and our **is_winner()** method receiving a **C**. **Figure 3** on the previous page, displays our Pylama mccabe analysis on the same program. Instead of returning a rank for every component in every module, Pylama simply let us know about our **is_winner()** method. The Radon module suggests that a rank of **C**, which reflects a CC score between 11 and 20, contains a *moderate* risk due to the component being slightly complex. Meanwhile, our Pylama result gave us the exact score for the method (**15**) and states that the method "is too complex" – which is more likely to prompt a programmer to re-examine the method in question and potentially make changes (such as refactoring) in order to remove the call-out altogether.

The Maintainability Index

The MI is a key metric that attempts to quantify how easily a program can be maintained and updated by blending together four separate metrics: (1) Halstead's program volume, (2) McCabe's CC score, (3) total lines of code, and (4) percentage of comments. The original MI expression had an upper bound of 171. Over the years, this has shifted to a return range between 1 and 100. Radon uses a formula that combines the derivate expressions used by SEI and Visual Studio.

Figure 4.

Radon's maintainability index expression

$$\boxed{MI = \max\left[0, 100\,\frac{171 - 5.2\ln V - 0.23G - 16.2\ln L + 50\sin(\sqrt{2.4C})}{171}\right]}$$

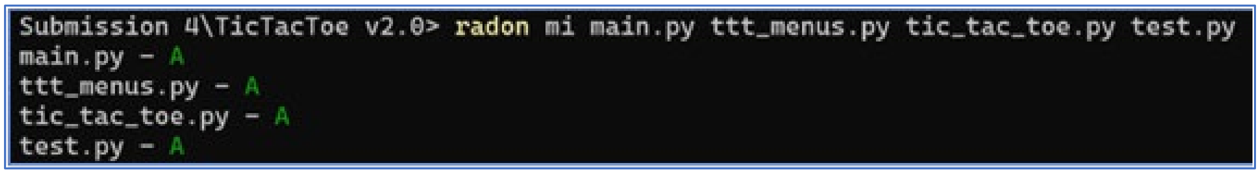
Looking to **Figure 4** on the previous page - **V** represents Halstead's volume, **G** is the total CC score, **L** is the number of source code lines, and **C** is the percent of comment lines converted to radians (Lacchia, 2020). Despite being adopted by Microsoft's Visual Studio IDE, MI is a rather controversial software metric. In Radon's documentation there is even a note that states "Maintainability Index is still a very experimental metric and should not be taken into account as seriously as the other metrics." This is due to at least four reasons according to Gilboy (2022):

1. *Over-reliance on lines of code* – The raw metric, lines of code, is not only a direct component of MI's formula, it is also a part of the expression that determines a program's volume and is heavily correlated with a program's CC score.
2. *Built on averages* – Despite evidence that both complexity and maintainability follow a power law, we calculate MI using volume and CC averages instead causing a large discrepancy when attempting to calculate the true cost of extremely complex functions, classes, and files in a codebase.
3. *Designed for a single company* – MI was created by HP engineers for HP projects.
4. *Not built for all languages* – The original MI formula was designed for projects written in C and is not reconsidered when applied to other programming languages.

Demonstrated by **Figure 5** on the next page, when running Radon's MI analysis, the default results present as a rank of **A** (score: 20-100), **B** (score: 10-19), or **C** (score: 0-9), with **A** equating to "very high" maintainability (Lacchia, 2020).

Figure 5.

Radon's maintainability index results



```
Submission 4\TicTacToe v2.0> radon mi main.py ttt_menus.py tic_tac_toe.py test.py
main.py - A
ttt_menus.py - A
tic_tac_toe.py - A
test.py - A
```

A Final Analysis with Pylint

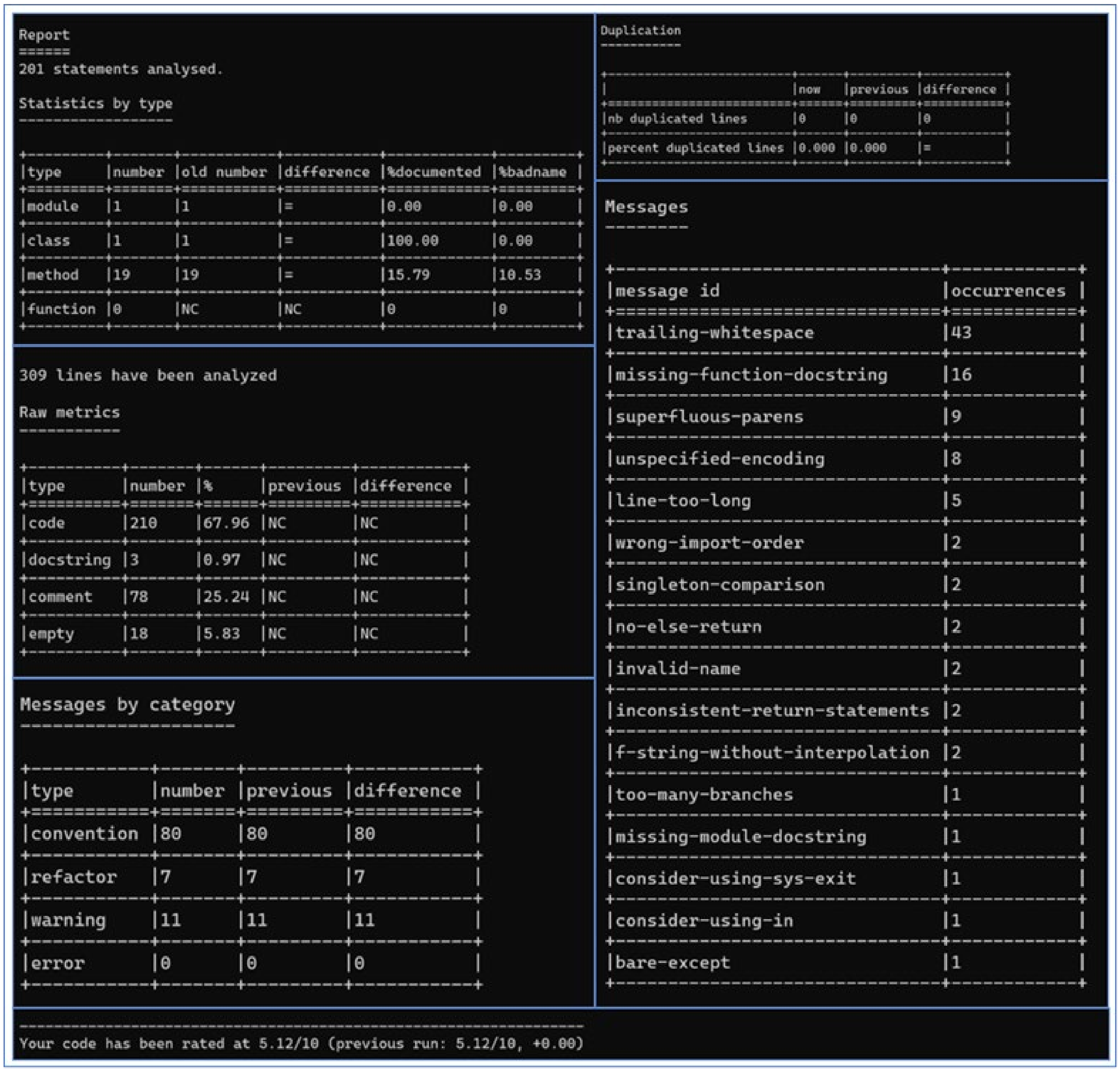
The third and final Python library we used to explore the use of software metrics is called **pylint**. Pylint is a static code analyzer that checks for errors, enforced a coding standard, looks for code smells (surface indications of deeper issues within source code), and makes suggestions on code refactoring if complex methods or functions are found (*What is Pylint?* 2022).

When running Pylint against out Tic Tac Toe program, we used the library's report feature to provide some basic statistics about the issues found for each of our modules. Each report is divided into six sections: (1) statistics by type, (2) raw metrics, (3) duplications, (4) messages by category, (5) messages (message ID & count), and (6) an overall code rating out of 10. This section will focus solely on our tic_tac_toe module's results and corrections due to the overwhelming amount of information Pylint provides.

The report generated for our tic_tac_toe module can be found as **Figure 6** on the next page. The actual results that printed to console can be found in **Appendix C** due to the sheer number of messages (98). The messages for our tic_tac_toe module are sorted by message type from using a short Python script; they are not presented this way by default. It is here that we will begin correcting some of the code flagged by our different tools. Figure 6 shows that our tic_tac_toe module has **80** potential issues concerning basic Python conventions – we can reduce this number quickly as most of them are due to the programmer adding an extra space at the end of a line of code.

Figure 6.

Pylint report for the tic_tac_toe module



```
Report
======
201 statements analysed.

Statistics by type
------------------

+---------+-------+-----------+-----------+------------+---------+
|type     |number |old number |difference |%documented |%badname |
+=========+=======+===========+===========+============+=========+
|module   |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|class    |1      |1          |=          |100.00      |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|method   |19     |19         |=          |15.79       |10.53    |
+---------+-------+-----------+-----------+------------+---------+
|function |0      |NC         |NC         |0           |0        |
+---------+-------+-----------+-----------+------------+---------+
```

```
309 lines have been analyzed

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |210    |67.96 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |3      |0.97  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |78     |25.24 |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |18     |5.83  |NC       |NC         |
+----------+-------+------+---------+-----------+
```

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |80     |80       |80         |
+-----------+-------+---------+-----------+
|refactor   |7      |7        |7          |
+-----------+-------+---------+-----------+
|warning    |11     |11       |11         |
+-----------+-------+---------+-----------+
|error      |0      |0        |0          |
+-----------+-------+---------+-----------+
```

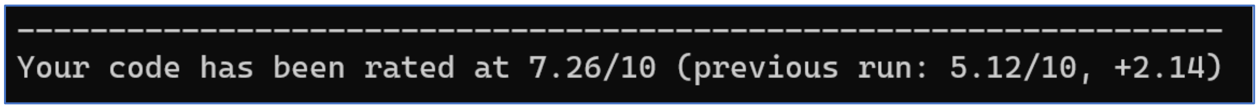
```
Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |0        |0          |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |0.000    |=          |
+-------------------------+------+---------+-----------+
```

```
Messages
--------

+-------------------------------+------------+
|message id                     |occurrences |
+===============================+============+
|trailing-whitespace            |43          |
+-------------------------------+------------+
|missing-function-docstring     |16          |
+-------------------------------+------------+
|superfluous-parens             |9           |
+-------------------------------+------------+
|unspecified-encoding           |8           |
+-------------------------------+------------+
|line-too-long                  |5           |
+-------------------------------+------------+
|wrong-import-order             |2           |
+-------------------------------+------------+
|singleton-comparison           |2           |
+-------------------------------+------------+
|no-else-return                 |2           |
+-------------------------------+------------+
|invalid-name                   |2           |
+-------------------------------+------------+
|inconsistent-return-statements |2           |
+-------------------------------+------------+
|f-string-without-interpolation |2           |
+-------------------------------+------------+
|too-many-branches              |1           |
+-------------------------------+------------+
|missing-module-docstring       |1           |
+-------------------------------+------------+
|consider-using-sys-exit        |1           |
+-------------------------------+------------+
|consider-using-in              |1           |
+-------------------------------+------------+
|bare-except                    |1           |
+-------------------------------+------------+
```

```
------------------------------------------------------------------
Your code has been rated at 5.12/10 (previous run: 5.12/10, +0.00)
```

Figure 7.

Pylint score after removing excess white spaces in code in tic_tac_toe module



```
-------------------------------------------------------------------
Your code has been rated at 7.26/10 (previous run: 5.12/10, +2.14)
```

Figure 7 shows the updated Pylint score for our tic_tac_toe module after simply deleting the blank spaces at the end of lines of code (msg id: C0303). We ran Pylint against our code without including the report parameter this time – after we fix the following set of convention issues, we will run Pylint again with reports set to yes in order to demonstrate how Pylint visualizes the changes made.

After our initial edits, we reduced our total number of convention messages from 80 to 37. The other convention messages found in Appendix C include m*ethod name not conforming to snake_case* (msg id: C0103), *missing module docstring* (msg id: C0114), *missing method or function docstring* (msg id: C0116), *line too long* (msg id: C0301), *unnecessary parens* (msg id: C0325) and *wrong import order* (msg id: C0411).

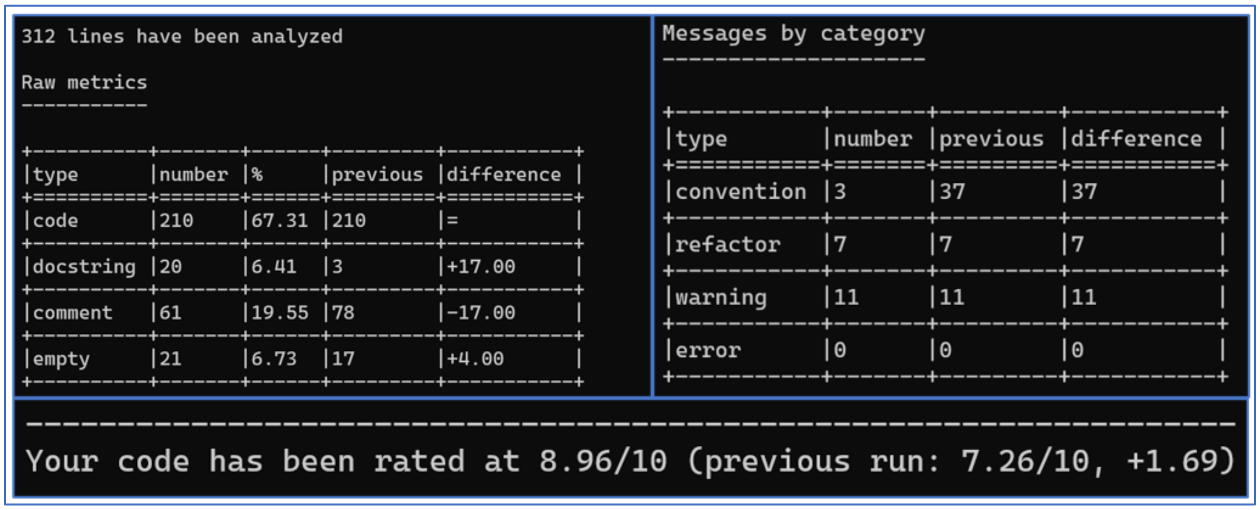
In order to address the rest of these results, we start by changing the name of the two methods with inconsistent styles from **game_mode_1_PvP()** to **game_mode_1()** and **game_mode_2_PvE()** to simply **game_mode_2()**, then update their docstrings to include what game mode they each represent instead. Next, we add the missing docstrings for our tic_tac_toe module and methods by simply shifting the program's comments to docstrings where appropriate.

Then, we can consider whether we care about lines of code that are in excess of 100 characters – five of these lines were called out in our results. All but one of these lines included long comments that were easily shortened by rephrasing. The second message however, called out a line of code that both prompted and captured a human player's input when it is their turn to make a move. We will leave this line as is – with the expectation that it will remain on our Pylint list. Pylint does offer options to ignore specific msg ids, however, we will not be exploring this feature here. Instead, we shall move on to fixing the next two sets of convention messages that call out unnecessary parentheses used after keywords such as **if** and **return** and the improper import orders before rerunning our Pylint analysis to review the results of our modifications.

The results in **Figure 8** below show us that we still have three convention messages, when we only expected one. This is because we missed a sub-category that was calling out the way the replay option is coded in our game mode methods under msg id: C0121. We can make these changes during the next section. Figure 8 also shows us that our tic_tac_toe module has gone from an original score of

Figure 8.

Pylint results after working through convention messages for tic_tac_toe module



```
312 lines have been analyzed

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |210    |67.31 |210      |=          |
+----------+-------+------+---------+-----------+
|docstring |20     |6.41  |3        |+17.00     |
+----------+-------+------+---------+-----------+
|comment   |61     |19.55 |78       |-17.00     |
+----------+-------+------+---------+-----------+
|empty     |21     |6.73  |17       |+4.00      |
+----------+-------+------+---------+-----------+
```

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |3      |37       |37         |
+-----------+-------+---------+-----------+
|refactor   |7      |7        |7          |
+-----------+-------+---------+-----------+
|warning    |11     |11       |11         |
+-----------+-------+---------+-----------+
|error      |0      |0        |0          |
+-----------+-------+---------+-----------+
```

```
-----------------------------------------------------------------
Your code has been rated at 8.96/10 (previous run: 7.26/10, +1.69)
```

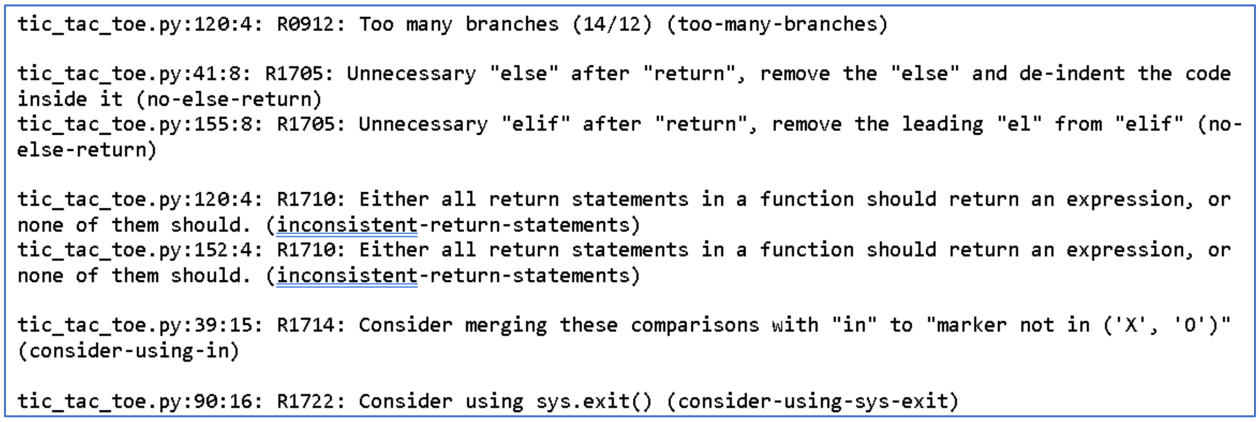
5.12 out of **10** to **8.96** just from making the simple convention changes Pylint found. In fact, our resulting console printout is now able to fit entirely on screen!

Before we end this section of our report, we are going to briefly review the **refractor** and **warning** messages from our tic_tac_toe module's Pylint analysis as well. The first refractor message, displayed in **Figure 9** below, points to our **is_winner()** method, which we can recall received a **C** rank in cyclomatic complexity. In order to modify this method, we could theoretically break up the different win patterns into separate methods – however, we are not going to follow through on this action at this point in time.

The next two messages want a simple removal of the **else** keyword prior to **return** statements. These can be easily rectified within the program. The messages concerning *inconsistent return statements*, however, are confusing. They both point to our moderately complex **is_winner()** method, which has the same return variable (**win**) throughout the entire method. So, we are going to keep these two flags as they are for now and decide later if such a convention is worth further

Figure 9.

Pylint refractor messages for tic_tac_toe module


```
tic_tac_toe.py:120:4: R0912: Too many branches (14/12) (too-many-branches)

tic_tac_toe.py:41:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code
inside it (no-else-return)
tic_tac_toe.py:155:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-
else-return)

tic_tac_toe.py:120:4: R1710: Either all return statements in a function should return an expression, or
none of them should. (inconsistent-return-statements)
tic_tac_toe.py:152:4: R1710: Either all return statements in a function should return an expression, or
none of them should. (inconsistent-return-statements)

tic_tac_toe.py:39:15: R1714: Consider merging these comparisons with "in" to "marker not in ('X', 'O')"
(consider-using-in)

tic_tac_toe.py:90:16: R1722: Consider using sys.exit() (consider-using-sys-exit)
```

exploration.

Next, we are asked to consider modifying line 39 from:

while (marker != ‘X’ and marker != ‘O’):

to

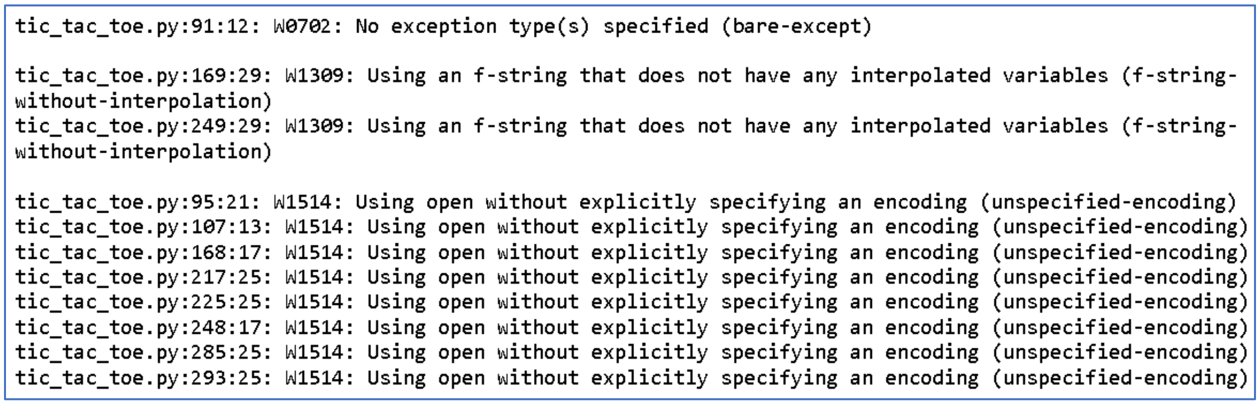
while marker not in (‘X’, ‘O’):

which is a simple enough request to complete now. The last refractor message suggests using **sys.exit()** in line 90 which currently calls just **exit()**. This change would require us to add **sys** as an imported module and would not change the functionality of our program at all as far as I can tell, so we are going to ignore it.

Finally, **Figure 10** shows our warning messages. The first message wants us to define an exception type within our **player_move()** method. Since we are making these changes without play testing, however, we will be skipping this flag for now as it is possible this was a conscious choice to catch all errors under one line of code. The next two warnings concerning f-strings, however, should be a simple modification.

Figure 10.

Pylint warning messages for the tic_tac_toe module

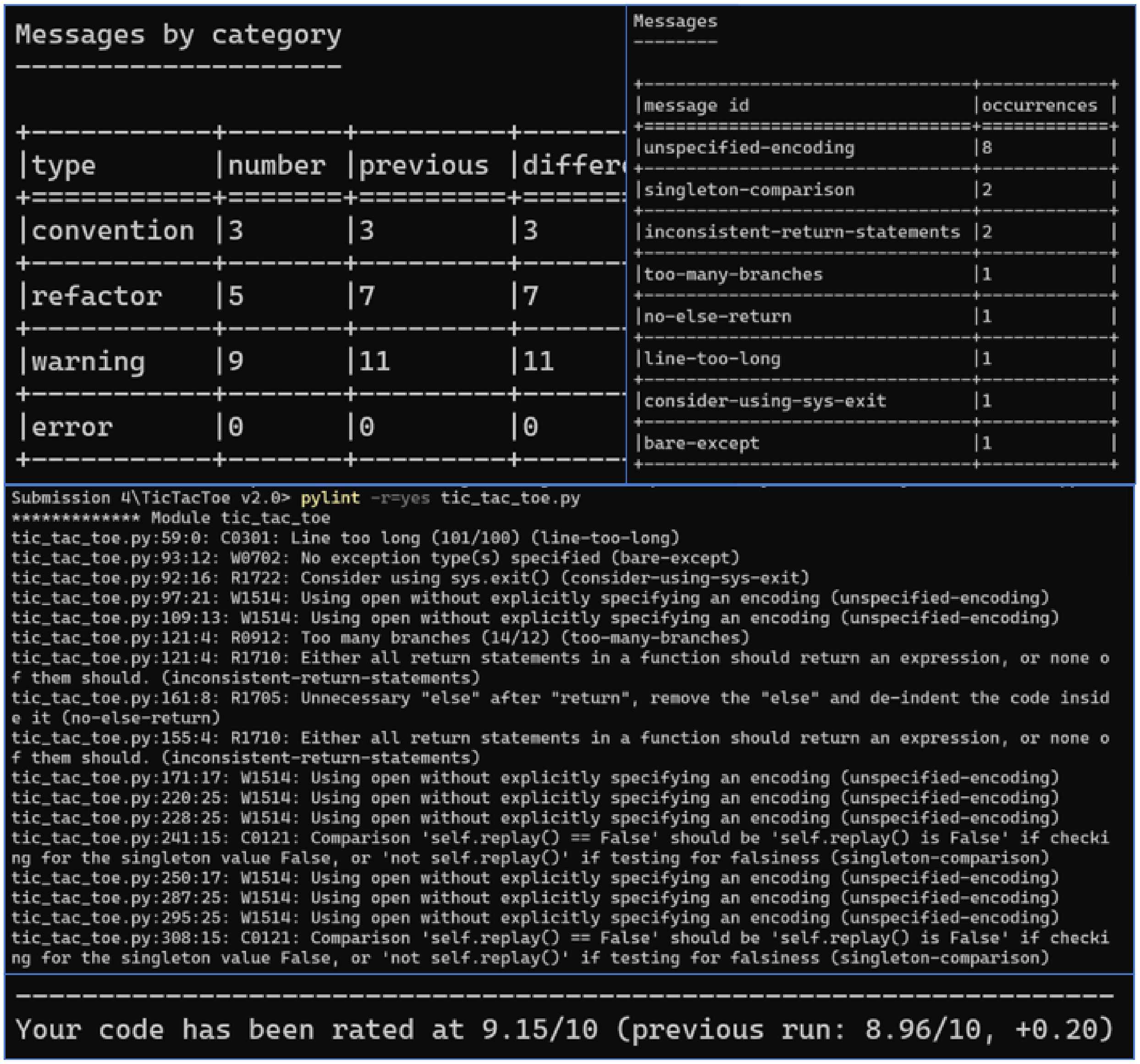


```
tic_tac_toe.py:91:12: W0702: No exception type(s) specified (bare-except)

tic_tac_toe.py:169:29: W1309: Using an f-string that does not have any interpolated variables (f-string-
without-interpolation)
tic_tac_toe.py:249:29: W1309: Using an f-string that does not have any interpolated variables (f-string-
without-interpolation)

tic_tac_toe.py:95:21: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:107:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:168:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:217:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:225:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:248:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:285:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:293:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

The rest of the warnings have to do with the way our Tic Tac Toe programs tracks the player movements and match outcomes in a text document. There is not really a good reason to enforce this warning as the default works fine for our use case at this time. Therefore, we are now moving on to our final Pylint analysis!

Figure 11.

Final Pylint results for the tic_tac_toe module



```
Messages by category
--------------------

+-----------+-------+---------+------
|type       |number |previous |differe
+===========+=======+=========+======
|convention |3      |3        |3
+-----------+-------+---------+------
|refactor   |5      |7        |7
+-----------+-------+---------+------
|warning    |9      |11       |11
+-----------+-------+---------+------
|error      |0      |0        |0
+-----------+-------+---------+------
```

```
Messages
--------

+-------------------------------+------------+
|message id                     |occurrences |
+===============================+============+
|unspecified-encoding           |8           |
+-------------------------------+------------+
|singleton-comparison           |2           |
+-------------------------------+------------+
|inconsistent-return-statements |2           |
+-------------------------------+------------+
|too-many-branches              |1           |
+-------------------------------+------------+
|no-else-return                 |1           |
+-------------------------------+------------+
|line-too-long                  |1           |
+-------------------------------+------------+
|consider-using-sys-exit        |1           |
+-------------------------------+------------+
|bare-except                    |1           |
+-------------------------------+------------+
```

```
Submission 4\TicTacToe v2.0> pylint -r=yes tic_tac_toe.py
************* Module tic_tac_toe
tic_tac_toe.py:59:0: C0301: Line too long (101/100) (line-too-long)
tic_tac_toe.py:93:12: W0702: No exception type(s) specified (bare-except)
tic_tac_toe.py:92:16: R1722: Consider using sys.exit() (consider-using-sys-exit)
tic_tac_toe.py:97:21: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:109:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:121:4: R0912: Too many branches (14/12) (too-many-branches)
tic_tac_toe.py:121:4: R1710: Either all return statements in a function should return an expression, or none o
f them should. (inconsistent-return-statements)
tic_tac_toe.py:161:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code insid
e it (no-else-return)
tic_tac_toe.py:155:4: R1710: Either all return statements in a function should return an expression, or none o
f them should. (inconsistent-return-statements)
tic_tac_toe.py:171:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:220:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:228:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:241:15: C0121: Comparison 'self.replay() == False' should be 'self.replay() is False' if checki
ng for the singleton value False, or 'not self.replay()' if testing for falsiness (singleton-comparison)
tic_tac_toe.py:250:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:287:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:295:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:308:15: C0121: Comparison 'self.replay() == False' should be 'self.replay() is False' if checki
ng for the singleton value False, or 'not self.replay()' if testing for falsiness (singleton-comparison)
```

```
------------------------------------------------------------------
Your code has been rated at 9.15/10 (previous run: 8.96/10, +0.20)
```

Displayed in **Figure 11** on the previous page, is our final Pylint printout for our tic_tac_toe module, with few enough flags to capture them all with a single screenshot! Compared to the length of Appendix C, we have clearly improved the quality of our module a lot by correcting the simplest of flags. One interesting note, from the message list above, is that we have a new **refractor** message pointing to another method in our module that contains the keyword **else** after a **return** statement. The reason this was not caught in the previous analyses is unclear.

Additionally, it appears that the **difference** column in our **Messages by category** reports does not behave as intended. Instead of displaying the number of changes made by giving us the difference between the **number** and **previous** columns, it appears to simply clone the **previous** column. Other than that, the rest of the Pylint results for our tic_tac_toe module are now ones that we have considered and chosen to keep as they are. If you wish to review the initial Pylint reports for our other three Tic Tac Toe modules, they can be found in **Appendix D**.

Conclusion

The software metrics covered in this report are some of the most basic examples of quantifying software quality. In using more than one method to determine our system's software metrics, we were able to better understand which portions were considered complex and which parts simply failed to meet standard Python conventions. Completing this project has demonstrated how easily one can use metrics testing to improve the quality of their code's readability and maintainability.

Future research on this topic would include reviewing the effects of our modifications on our Radon and Pylama reports (**Appendix E** contains a second

Pylama analysis that was run after our work through the Pylint reports – the resulting message list was cut from six to three pages worth of flags), as well as further exploration into more complex software metrics such as those described in our introduction concerning module cohesion and coupling.

References

Federal Aviation Administration, VanSuetendael, N., & Elwell, D., Software Quality Metrics (1991). Pleasantville, NJ; Computer Resource Management, Inc. Retrieved June 8, 2022, from https://apps.dtic.mil/sti/pdfs/ADA241510.pdf.

Gilboy, T. (2022, March 7). [web log]. Retrieved June 8, 2022, from https://sourcery.ai/blog/maintainability-index/.

IBM. (2021, April 22). *Halstead Complexity Report*. Halstead Complexity Report - IBM Documentation. Retrieved June 8, 2022, from https://www.ibm.com/docs/en/addi/5.1.0?topic=SSRR9Q_5.1.0%2FIBM_AD_Analyze_User_Guide_OUT_KC%2FHalsteadReport.html

Lacchia, M. (2020, January 28). *Welcome to Radon's documentation!* Welcome to Radon's documentation! - Radon 4.1.0 documentation. Retrieved June 8, 2022, from https://radon.readthedocs.io/en/latest/index.html

Logilab and PyCQA. (2022, June 8). *What is Pylint?* Pylint 2.15.0-dev0 documentation. Retrieved June 8, 2022, from https://pylint.pycqa.org/en/latest/

Mall, R. (2018, June). *Fundamentals of Software Engineering, Fifth Edition*. Google Books. Retrieved June 8, 2022, from https://books.google.com/books?id=-JNuDwAAQBAJ&pg=PA125&lpg=PA125&dq=halstead%27s%2Bfirst%2Bhypothesis%2Bis%2Binaccurate%2Bfor%2Bsmaller%2Bprograms&source=bl&ots=PAKi2QTdqc&sig=ACfU3U18ejYzQ5hMg77hMADTcS5Ly4TrHQ&hl=en&sa=X&ved=2ahUKEwizmMvc6Zv4AhXknI4IHfKFA6kQ6AF6BAgOEAM#v=onepage&q=halstead's%20first%20hypothesis%20is%20inaccurate%20for%20smaller%20programs&f=false

Tsui, F. F., Bernal, B., & Karam, O. (2014). Chapter 8: Design Characteristics and Metrics. In *Essentials of Software Engineering* (3rd ed., pp. 165–186). Jones & Bartlett Learning.

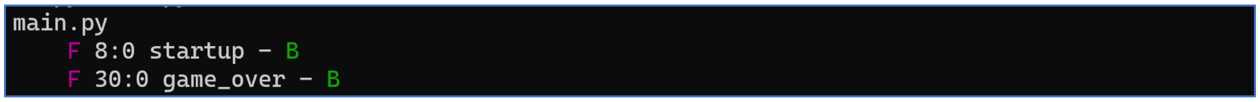
Appendix A

Initial Pylama Report for All Four Tic Tac Toe Modules

```
Namespace(paths=['C:/Users/Saman/Desktop/CS 504 (software engineering)/05. Independent Project/CS504-
Project-Samantha-Hipple/Submission 4/TicTacToe v2.0'], verbose=True, options=None, linters=['pylint',
'mypy', 'mccabe', 'pycodestyle', 'pyflakes', 'pydocstyle', 'isort'], from_stdin=False, concurrent=False,
format='pycodestyle', abspath=False, max_line_length=100, select=set(), ignore=set(), skip=[], sort=None,
report='report.txt', hook=False, max_complexity=10, pydocstyle_convention=None, pylint_confidence=None,
radon_no_assert=False, radon_show_closures=False, file_params={}, linters_params={})

Run [pylint] main.py
Run [mypy] main.py
Run [mccabe] main.py
Run [pycodestyle] main.py
Run [pyflakes] main.py
Run [pydocstyle] main.py
Run [isort] main.py

Run [pylint] test.py
Run [mypy] test.py
Run [mccabe] test.py
Run [pycodestyle] test.py
Run [pyflakes] test.py
Run [pydocstyle] test.py
Run [isort] test.py

Run [pylint] tic_tac_toe.py
Run [mypy] tic_tac_toe.py
Run [mccabe] tic_tac_toe.py
Run [pycodestyle] tic_tac_toe.py
Run [pyflakes] tic_tac_toe.py
Run [pydocstyle] tic_tac_toe.py
Run [isort] tic_tac_toe.py

Run [pylint] ttt_menus.py
Run [mypy] ttt_menus.py
Run [mccabe] ttt_menus.py
Run [pycodestyle] ttt_menus.py
Run [pyflakes] ttt_menus.py
Run [pydocstyle] ttt_menus.py
Run [isort] ttt_menus.py

main.py:0:1  Incorrectly sorted imports. [isort]
main.py:1:1 C0114 Missing module docstring [pylint]
main.py:1:1 D100 Missing docstring in public module [pydocstyle]
main.py:8:1 C0116 Missing function or method docstring [pylint]
main.py:8:1 E302 expected 2 blank lines, found 1 [pycodestyle]
main.py:8:1 D103 Missing docstring in public function [pydocstyle]
main.py:19:32 C0303 Trailing whitespace [pylint]
main.py:19:1 C0325 Unnecessary parens after 'if' keyword [pylint]
main.py:19:13 R1723 Unnecessary "elif" after "break", remove the leading "el" from "elif" [pylint]
main.py:19:32 W291 trailing whitespace [pycodestyle]
main.py:22:34 C0303 Trailing whitespace [pylint]
main.py:22:1 C0325 Unnecessary parens after 'elif' keyword [pylint]
main.py:22:34 W291 trailing whitespace [pycodestyle]
main.py:25:1 C0325 Unnecessary parens after 'elif' keyword [pylint]
main.py:27:17 R1722 Consider using sys.exit() [pylint]
main.py:29:1 C0303 Trailing whitespace [pylint]
main.py:29:1 W293 blank line contains whitespace [pycodestyle]
main.py:30:1 C0116 Missing function or method docstring [pylint]
main.py:30:1 E302 expected 2 blank lines, found 1 [pycodestyle]
main.py:30:1 D103 Missing docstring in public function [pydocstyle]
main.py:39:13 R1722 Consider using sys.exit() [pylint]
main.py:41:32 C0303 Trailing whitespace [pylint]
main.py:41:1 C0325 Unnecessary parens after 'if' keyword [pylint]
main.py:41:32 W291 trailing whitespace [pycodestyle]
main.py:43:1 C0325 Unnecessary parens after 'if' keyword [pylint]
main.py:45:17 R1722 Consider using sys.exit() [pylint]
main.py:48:1 E305 expected 2 blank lines after class or function definition, found 1 [pycodestyle]
main.py:50:1 C0304 Final newline missing [pylint]
main.py:50:12 W292 no newline at end of file [pycodestyle]

test.py:0:1  Incorrectly sorted imports. [isort]
test.py:1:1 C0114 Missing module docstring [pylint]
test.py:1:1 D100 Missing docstring in public module [pydocstyle]
test.py:4:1 C0411 standard import "from io import StringIO" should be placed before "import numpy as np"
[pylint]
test.py:5:1 C0411 standard import "from unittest.mock import patch" should be placed before "import numpy
as np" [pylint]
test.py:6:1 C0411 third party import "import pytest" should be placed before "from tic_tac_toe import
TicTacToe" [pylint]
test.py:6:1 W0611 Unused import pytest [pylint]
test.py:12:1 E302 expected 2 blank lines, found 1 [pycodestyle]
test.py:13:1 D203 1 blank line required before class docstring [pydocstyle]
test.py:13:1 D300 Use """triple double quotes""" [pydocstyle]
test.py:18:27 E261 at least two spaces before inline comment [pycodestyle]
test.py:20:48 E261 at least two spaces before inline comment [pycodestyle]
test.py:28:78 C0303 Trailing whitespace [pylint]
test.py:28:78 W291 trailing whitespace [pycodestyle]
test.py:28:32 E261 at least two spaces before inline comment [pycodestyle]
test.py:32:23 E222 multiple spaces after operator [pycodestyle]
test.py:33:24 E222 multiple spaces after operator [pycodestyle]
test.py:34:1 C0301 Line too long (115/100) [pylint]
test.py:35:24 E222 multiple spaces after operator [pycodestyle]
test.py:36:24 E222 multiple spaces after operator [pycodestyle]
test.py:37:24 E222 multiple spaces after operator [pycodestyle]
test.py:38:1 C0301 Line too long (115/100) [pylint]
test.py:39:24 E222 multiple spaces after operator [pycodestyle]
test.py:40:24 E222 multiple spaces after operator [pycodestyle]
test.py:41:24 E222 multiple spaces after operator [pycodestyle]
test.py:42:1 C0301 Line too long (115/100) [pylint]
test.py:43:24 E222 multiple spaces after operator [pycodestyle]
test.py:44:24 E222 multiple spaces after operator [pycodestyle]
test.py:50:1 D400 First line should end with a period [pydocstyle]
test.py:50:1 D415 First line should end with a period, question mark, or exclamation point [pydocstyle]
test.py:51:26 E261 at least two spaces before inline comment [pycodestyle]
test.py:52:24 E261 at least two spaces before inline comment [pycodestyle]
test.py:56:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:59:1 C0301 Line too long (102/100) [pylint]
test.py:60:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:61:37 E261 at least two spaces before inline comment [pycodestyle]
test.py:62:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:64:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:65:37 E261 at least two spaces before inline comment [pycodestyle]
test.py:66:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:68:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:69:31 C0303 Trailing whitespace [pylint]
test.py:69:31 W291 trailing whitespace [pycodestyle]
test.py:72:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:73:68 E261 at least two spaces before inline comment [pycodestyle]
test.py:74:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:75:46 E261 at least two spaces before inline comment [pycodestyle]
test.py:77:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:78:68 E261 at least two spaces before inline comment [pycodestyle]
test.py:79:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:80:46 E261 at least two spaces before inline comment [pycodestyle]
test.py:82:27 C0303 Trailing whitespace [pylint]
test.py:82:27 W291 trailing whitespace [pycodestyle]
test.py:84:1 D300 Use """triple double quotes""" [pydocstyle]
test.py:85:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:86:35 E261 at least two spaces before inline comment [pycodestyle]
test.py:87:1 C0301 Line too long (103/100) [pylint]
test.py:87:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:89:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:94:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:95:25 E261 at least two spaces before inline comment [pycodestyle]
test.py:96:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:97:52 E261 at least two spaces before inline comment [pycodestyle]
test.py:98:38 E261 at least two spaces before inline comment [pycodestyle]
test.py:99:65 E261 at least two spaces before inline comment [pycodestyle]
test.py:101:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:106:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:107:25 E261 at least two spaces before inline comment [pycodestyle]
test.py:108:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:109:44 E261 at least two spaces before inline comment [pycodestyle]
test.py:110:49 E261 at least two spaces before inline comment [pycodestyle]
test.py:111:65 E261 at least two spaces before inline comment [pycodestyle]
test.py:121:5 E265 block comment should start with '# ' [pycodestyle]
test.py:124:36 C0303 Trailing whitespace [pylint]
test.py:124:36 W291 trailing whitespace [pycodestyle]
test.py:128:1 E305 expected 2 blank lines after class or function definition, found 1 [pycodestyle]
test.py:129:1 C0304 Final newline missing [pylint]
test.py:129:21 W292 no newline at end of file [pycodestyle]
test.py:129:20 W291 trailing whitespace [pycodestyle]

tic_tac_toe.py:0:1  Incorrectly sorted imports. [isort]
tic_tac_toe.py:1:1 C0114 Missing module docstring [pylint]
tic_tac_toe.py:1:1 D100 Missing docstring in public module [pydocstyle]
tic_tac_toe.py:2:1 C0411 standard import "import random" should be placed before "import numpy as np"
[pylint]
tic_tac_toe.py:3:1 C0411 standard import "from time import sleep" should be placed before "import numpy as
np" [pylint]
tic_tac_toe.py:6:1 E302 expected 2 blank lines, found 1 [pycodestyle]
tic_tac_toe.py:7:1 D203 1 blank line required before class docstring [pydocstyle]
tic_tac_toe.py:7:1 D204 1 blank line required after class docstring [pydocstyle]
tic_tac_toe.py:7:1 D400 First line should end with a period [pydocstyle]
tic_tac_toe.py:9:1 D107 Missing docstring in __init__ [pydocstyle]
tic_tac_toe.py:10:24 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:11:68 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:12:1 C0301 Line too long (113/100) [pylint]
tic_tac_toe.py:12:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:15:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:15:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:16:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:17:33 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:18:52 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:21:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:21:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:25:29 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
tic_tac_toe.py:27:46 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
tic_tac_toe.py:30:55 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:30:55 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:33:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:33:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:34:24 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:37:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:37:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:38:21 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:39:16 R1714 Consider merging these comparisons with "in" to "marker not in ('X', 'O')"
[pylint]
tic_tac_toe.py:39:49 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:40:71 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:41:34 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:41:9 R1705 Unnecessary "else" after "return", remove the "else" and de-indent the code
inside it [pylint]
tic_tac_toe.py:41:34 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:41:26 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:42:30 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:43:22 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:43:22 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:43:14 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:44:30 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:47:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:47:30 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:47:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:48:96 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:48:96 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:48:48 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:49:79 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:49:79 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:49:32 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:52:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:52:25 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:52:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:53:36 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:56:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:56:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:57:1 C0301 Line too long (101/100) [pylint]
tic_tac_toe.py:58:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:59:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:62:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:62:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:63:40 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:64:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:65:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:66:22 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:69:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:69:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:70:28 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:71:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:72:66 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:72:66 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:72:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:73:1 C0301 Line too long (102/100) [pylint]
tic_tac_toe.py:73:42 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:74:50 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:75:1 C0301 Line too long (102/100) [pylint]
tic_tac_toe.py:75:45 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:76:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:79:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:79:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:80:38 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:83:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:83:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:84:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:85:83 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:85:83 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:85:17 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:86:49 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:87:58 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:88:38 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:89:66 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:89:66 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:89:41 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:90:17 R1722 Consider using sys.exit() [pylint]
tic_tac_toe.py:90:23 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:91:13 W0702 No exception type(s) specified [pylint]
tic_tac_toe.py:91:13 E722 do not use bare 'except' [pycodestyle]
tic_tac_toe.py:91:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:92:53 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:93:18 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:94:44 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:95:22 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:95:62 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:96:86 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:96:86 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:96:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:97:52 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:98:29 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:101:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:101:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:102:35 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:103:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:104:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:105:17 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:106:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:107:83 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:107:14 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:107:83 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:107:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:108:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:109:44 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:112:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:112:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:113:60 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:113:60 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:113:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:114:66 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:114:66 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:114:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:115:1 C0301 Line too long (111/100) [pylint]
tic_tac_toe.py:115:1 C0325 Unnecessary parens after 'if' keyword [pylint]
tic_tac_toe.py:115:44 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:116:33 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:117:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:119:41 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:119:41 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:120:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:120:5 R0912 Too many branches (14/12) [pylint]
tic_tac_toe.py:120:5 R1710 Either all return statements in a function should return an expression, or none
of them should. [pylint]
tic_tac_toe.py:120:5 C901 'TicTacToe.is_winner' is too complex (15) [mccabe]
tic_tac_toe.py:120:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:127:1 C0325 Unnecessary parens after 'if' keyword [pylint]
tic_tac_toe.py:128:1 C0325 Unnecessary parens after 'return' keyword [pylint]
tic_tac_toe.py:134:1 C0325 Unnecessary parens after 'if' keyword [pylint]
tic_tac_toe.py:135:1 C0325 Unnecessary parens after 'return' keyword [pylint]
tic_tac_toe.py:140:1 C0325 Unnecessary parens after 'if' keyword [pylint]
tic_tac_toe.py:141:1 C0325 Unnecessary parens after 'return' keyword [pylint]
tic_tac_toe.py:146:1 C0325 Unnecessary parens after 'if' keyword [pylint]
tic_tac_toe.py:147:1 C0325 Unnecessary parens after 'return' keyword [pylint]
tic_tac_toe.py:149:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:149:5 E301 expected 1 blank line, found 0 [pycodestyle]
tic_tac_toe.py:149:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:150:45 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:150:45 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:151:55 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:151:55 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:152:5 C0116 Missing function or method docstring [pylint]
tic_tac_toe.py:152:5 R1710 Either all return statements in a function should return an expression, or none
of them should. [pylint]
tic_tac_toe.py:152:5 E301 expected 1 blank line, found 0 [pycodestyle]
tic_tac_toe.py:152:1 D102 Missing docstring in public method [pydocstyle]
tic_tac_toe.py:155:9 R1705 Unnecessary "elif" after "return", remove the leading "el" from "elif" [pylint]
tic_tac_toe.py:160:26 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:160:26 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:163:5 C0103 Method name "game_mode_2_PvE" doesn't conform to snake_case naming style
[pylint]
tic_tac_toe.py:164:1 D400 First line should end with a period [pydocstyle]
tic_tac_toe.py:164:1 D415 First line should end with a period, question mark, or exclamation point
[pydocstyle]
tic_tac_toe.py:168:58 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:168:18 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:168:58 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:169:30 W1309 Using an f-string that does not have any interpolated variables [pylint]
tic_tac_toe.py:169:30  f-string is missing placeholders [pyflakes]
tic_tac_toe.py:171:32 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:171:32 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:173:53 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:173:53 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:180:65 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:180:65 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:196:24 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:196:24 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:206:64 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:206:64 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:211:22 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:211:22 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:217:66 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:217:26 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:217:66 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:220:26 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:220:26 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:225:26 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:226:62 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:226:62 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:228:26 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:228:26 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:230:37 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:230:37 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:238:16 C0121 Comparison 'self.replay() == False' should be 'self.replay() is False' if
checking for the singleton value False, or 'not self.replay()' if testing for falsiness [pylint]
tic_tac_toe.py:238:30 E712 comparison to False should be 'if cond is False:' or 'if not cond:'
[pycodestyle]
tic_tac_toe.py:241:1 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:241:1 W293 blank line contains whitespace [pycodestyle]
tic_tac_toe.py:242:5 E303 too many blank lines (2) [pycodestyle]
tic_tac_toe.py:243:5 C0103 Method name "game_mode_1_PvP" doesn't conform to snake_case naming style
[pylint]
tic_tac_toe.py:243:5 E301 expected 1 blank line, found 0 [pycodestyle]
tic_tac_toe.py:244:1 D400 First line should end with a period [pydocstyle]
tic_tac_toe.py:248:58 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:248:18 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:248:58 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:249:30 W1309 Using an f-string that does not have any interpolated variables [pylint]
tic_tac_toe.py:249:30  f-string is missing placeholders [pyflakes]
tic_tac_toe.py:251:32 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:251:32 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:253:53 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:253:53 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:260:65 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:260:65 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:273:49 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:273:49 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:274:64 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:274:64 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:277:37 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:277:37 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:280:24 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:280:24 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:285:66 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:285:26 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:285:66 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:288:26 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:288:26 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:293:26 W1514 Using open without explicitly specifying an encoding [pylint]
tic_tac_toe.py:294:62 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:294:62 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:296:26 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:296:26 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:298:37 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:298:37 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:304:31 C0303 Trailing whitespace [pylint]
tic_tac_toe.py:304:31 W291 trailing whitespace [pycodestyle]
tic_tac_toe.py:306:16 C0121 Comparison 'self.replay() == False' should be 'self.replay() is False' if
checking for the singleton value False, or 'not self.replay()' if testing for falsiness [pylint]
tic_tac_toe.py:306:30 E712 comparison to False should be 'if cond is False:' or 'if not cond:'
[pycodestyle]

ttt_menus.py:1:1 C0114 Missing module docstring [pylint]
ttt_menus.py:1:1 C0115 Missing class docstring [pylint]
ttt_menus.py:1:1 D100 Missing docstring in public module [pydocstyle]
ttt_menus.py:1:1 D101 Missing docstring in public class [pydocstyle]
ttt_menus.py:2:1 D107 Missing docstring in __init__ [pydocstyle]
ttt_menus.py:3:13 C0303 Trailing whitespace [pylint]
ttt_menus.py:3:13 W291 trailing whitespace [pycodestyle]
ttt_menus.py:5:5 C0116 Missing function or method docstring [pylint]
ttt_menus.py:5:1 D102 Missing docstring in public method [pydocstyle]
ttt_menus.py:6:18 E222 multiple spaces after operator [pycodestyle]
ttt_menus.py:13:78 C0303 Trailing whitespace [pylint]
ttt_menus.py:13:78 W291 trailing whitespace [pycodestyle]
ttt_menus.py:15:78 C0303 Trailing whitespace [pylint]
ttt_menus.py:15:78 W291 trailing whitespace [pycodestyle]
ttt_menus.py:23:1 C0325 Unnecessary parens after 'return' keyword [pylint]
ttt_menus.py:25:5 C0116 Missing function or method docstring [pylint]
ttt_menus.py:25:1 D102 Missing docstring in public method [pydocstyle]
ttt_menus.py:28:5 C0116 Missing function or method docstring [pylint]
ttt_menus.py:28:1 D102 Missing docstring in public method [pydocstyle]
ttt_menus.py:31:1 C0325 Unnecessary parens after 'return' keyword [pylint]
ttt_menus.py:33:5 C0116 Missing function or method docstring [pylint]
ttt_menus.py:33:1 D102 Missing docstring in public method [pydocstyle]
ttt_menus.py:34:20 E222 multiple spaces after operator [pycodestyle]
ttt_menus.py:41:80 C0303 Trailing whitespace [pylint]
ttt_menus.py:41:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:43:80 C0303 Trailing whitespace [pylint]
ttt_menus.py:43:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:47:80 C0303 Trailing whitespace [pylint]
ttt_menus.py:47:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:49:1 C0325 Unnecessary parens after 'return' keyword [pylint]
ttt_menus.py:51:5 C0116 Missing function or method docstring [pylint]
ttt_menus.py:51:1 D102 Missing docstring in public method [pydocstyle]
ttt_menus.py:52:1 C0304 Final newline missing [pylint]
ttt_menus.py:52:46 W292 no newline at end of file [pycodestyle]
```

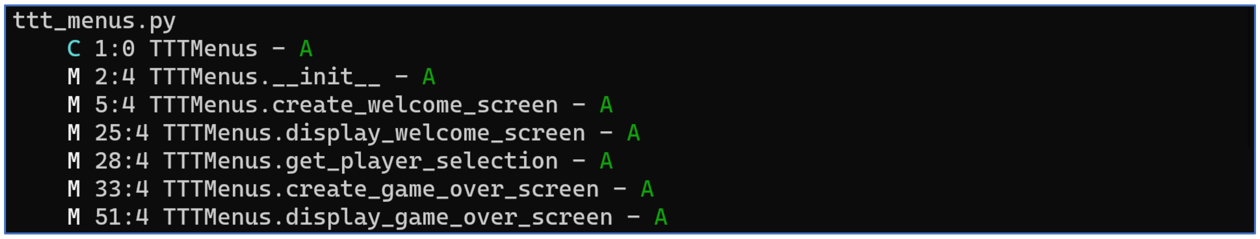
Appendix B

Initial Radon McCabe Cyclomatic Complexity Results



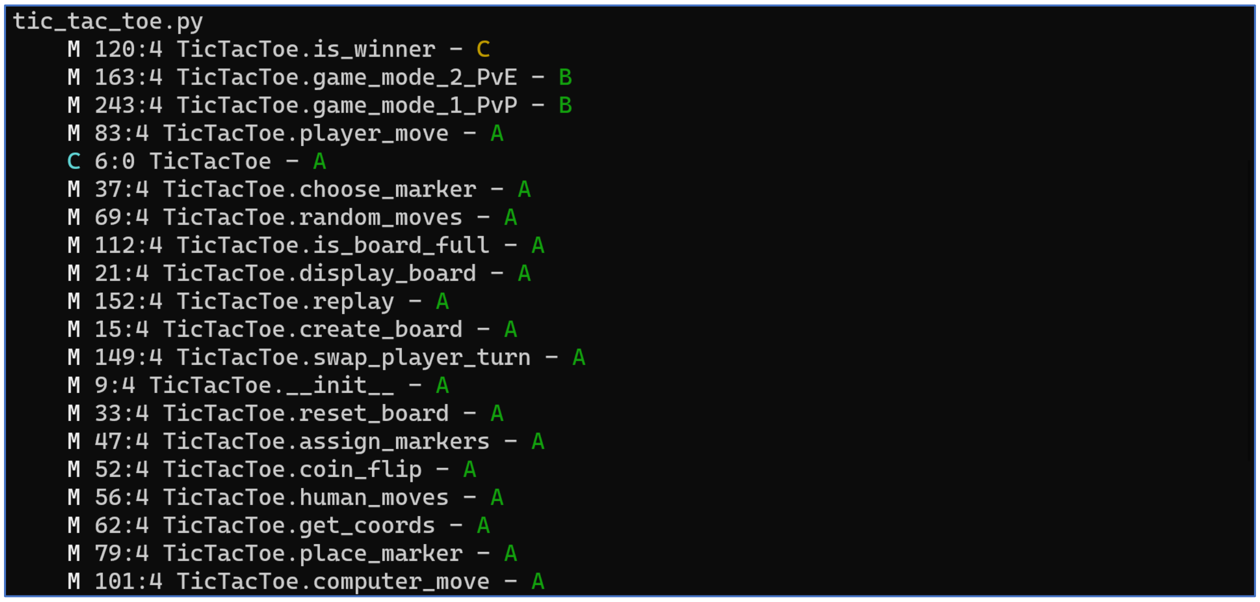
```
main.py
    F 8:0 startup - B
    F 30:0 game_over - B
```

Figure B1. *Radon CC ranks for main module*



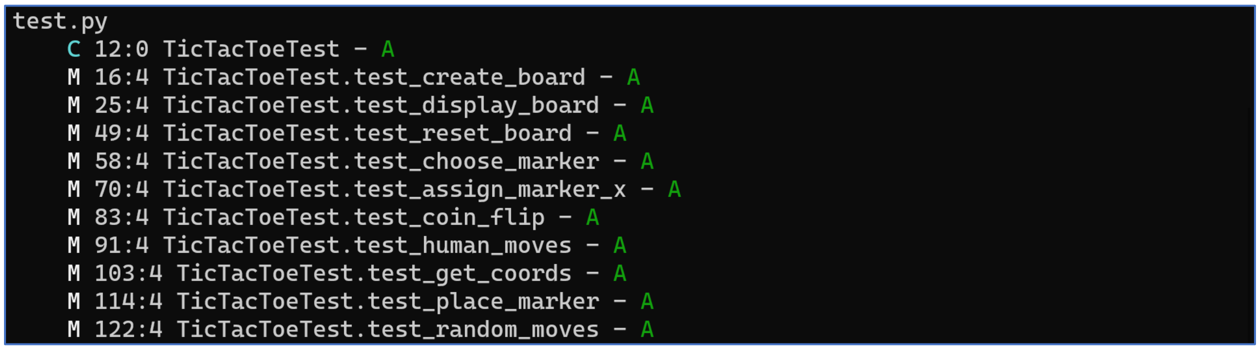
```
ttt_menus.py
    C 1:0 TTTMenus - A
    M 2:4 TTTMenus.__init__ - A
    M 5:4 TTTMenus.create_welcome_screen - A
    M 25:4 TTTMenus.display_welcome_screen - A
    M 28:4 TTTMenus.get_player_selection - A
    M 33:4 TTTMenus.create_game_over_screen - A
    M 51:4 TTTMenus.display_game_over_screen - A
```

Figure B2. *Radon CC ranks for ttt_menus module*



```
tic_tac_toe.py
    M 120:4 TicTacToe.is_winner - C
    M 163:4 TicTacToe.game_mode_2_PvE - B
    M 243:4 TicTacToe.game_mode_1_PvP - B
    M 83:4 TicTacToe.player_move - A
    C 6:0 TicTacToe - A
    M 37:4 TicTacToe.choose_marker - A
    M 69:4 TicTacToe.random_moves - A
    M 112:4 TicTacToe.is_board_full - A
    M 21:4 TicTacToe.display_board - A
    M 152:4 TicTacToe.replay - A
    M 15:4 TicTacToe.create_board - A
    M 149:4 TicTacToe.swap_player_turn - A
    M 9:4 TicTacToe.__init__ - A
    M 33:4 TicTacToe.reset_board - A
    M 47:4 TicTacToe.assign_markers - A
    M 52:4 TicTacToe.coin_flip - A
    M 56:4 TicTacToe.human_moves - A
    M 62:4 TicTacToe.get_coords - A
    M 79:4 TicTacToe.place_marker - A
    M 101:4 TicTacToe.computer_move - A
```

Figure B3. *Radon CC ranks for tic_tac_toe module*



```
test.py
    C 12:0 TicTacToeTest - A
    M 16:4 TicTacToeTest.test_create_board - A
    M 25:4 TicTacToeTest.test_display_board - A
    M 49:4 TicTacToeTest.test_reset_board - A
    M 58:4 TicTacToeTest.test_choose_marker - A
    M 70:4 TicTacToeTest.test_assign_marker_x - A
    M 83:4 TicTacToeTest.test_coin_flip - A
    M 91:4 TicTacToeTest.test_human_moves - A
    M 103:4 TicTacToeTest.test_get_coords - A
    M 114:4 TicTacToeTest.test_place_marker - A
    M 122:4 TicTacToeTest.test_random_moves - A
```

Figure B4. *Radon CC ranks for test module*

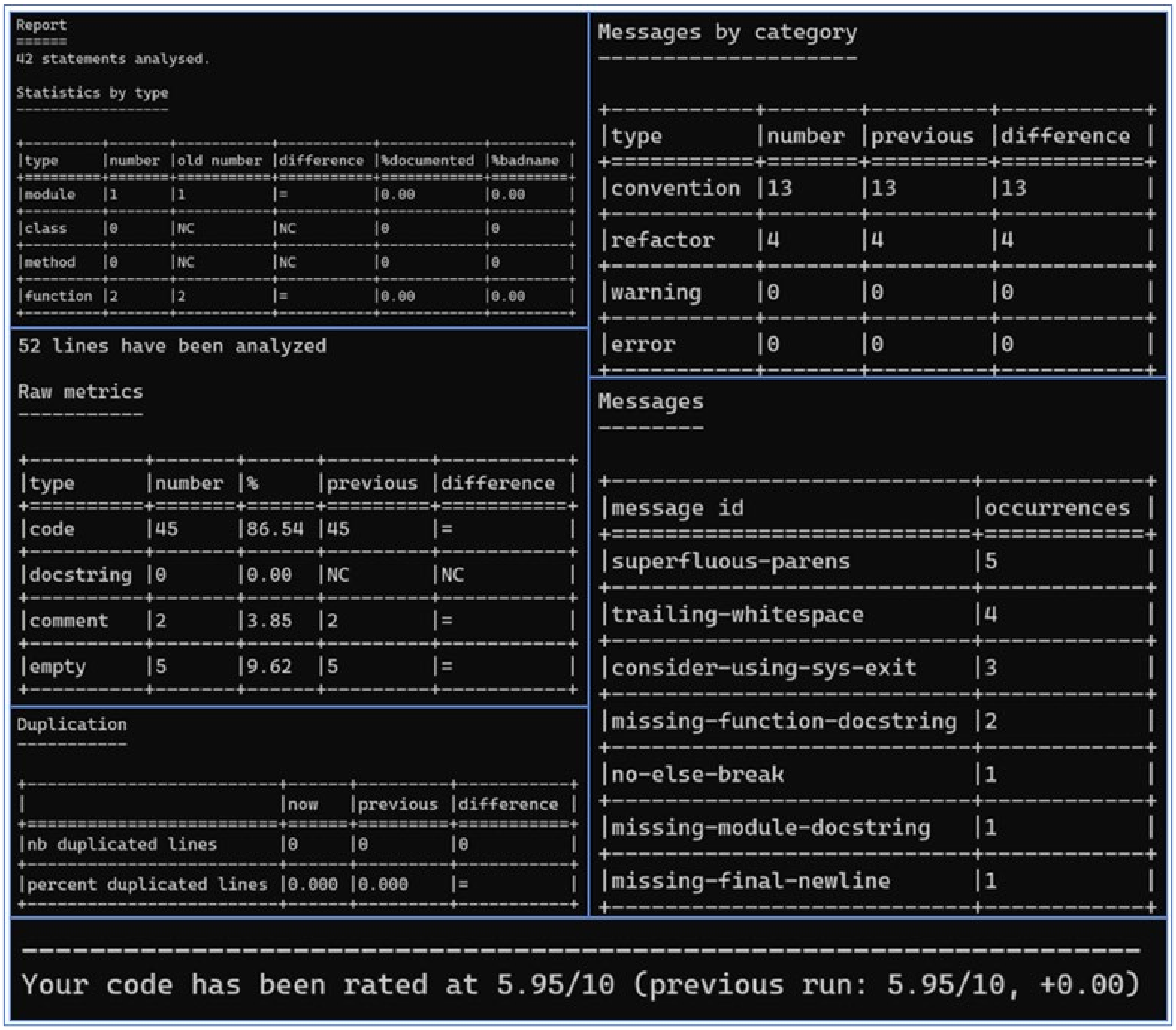
Appendix C

Initial Pylint Message List for the tic_tac_toe Module (sorted by msg id)

```
tic_tac_toe.py:163:4: C0103: Method name "game_mode_2_PvE" doesn't conform to snake_case naming style
(invalid-name)
tic_tac_toe.py:243:4: C0103: Method name "game_mode_1_PvP" doesn't conform to snake_case naming style
(invalid-name)

tic_tac_toe.py:1:0: C0114: Missing module docstring (missing-module-docstring)

tic_tac_toe.py:15:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:21:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:33:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:37:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:47:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:52:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:56:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:62:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:69:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:79:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:83:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:101:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:112:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:120:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:149:4: C0116: Missing function or method docstring (missing-function-docstring)
tic_tac_toe.py:152:4: C0116: Missing function or method docstring (missing-function-docstring)

tic_tac_toe.py:12:0: C0301: Line too long (113/100) (line-too-long)
tic_tac_toe.py:57:0: C0301: Line too long (101/100) (line-too-long)
tic_tac_toe.py:73:0: C0301: Line too long (102/100) (line-too-long)
tic_tac_toe.py:75:0: C0301: Line too long (102/100) (line-too-long)
tic_tac_toe.py:115:0: C0301: Line too long (111/100) (line-too-long)

tic_tac_toe.py:30:54: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:41:33: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:43:21: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:48:95: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:49:78: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:72:65: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:85:82: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:89:65: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:96:85: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:107:82: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:113:59: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:114:65: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:119:40: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:150:44: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:151:54: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:160:25: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:168:57: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:171:31: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:173:52: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:180:64: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:196:23: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:206:63: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:211:21: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:217:65: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:220:25: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:226:61: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:228:25: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:230:36: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:241:0: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:248:57: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:251:31: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:253:52: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:260:64: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:273:48: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:274:63: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:277:36: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:280:23: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:285:65: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:288:25: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:294:61: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:296:25: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:298:36: C0303: Trailing whitespace (trailing-whitespace)
tic_tac_toe.py:304:30: C0303: Trailing whitespace (trailing-whitespace)

tic_tac_toe.py:115:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
tic_tac_toe.py:127:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
tic_tac_toe.py:128:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
tic_tac_toe.py:134:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
tic_tac_toe.py:135:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
tic_tac_toe.py:140:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
tic_tac_toe.py:141:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
tic_tac_toe.py:146:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
tic_tac_toe.py:147:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)

tic_tac_toe.py:2:0: C0411: standard import "import random" should be placed before "import numpy as np"
(wrong-import-order)
tic_tac_toe.py:3:0: C0411: standard import "from time import sleep" should be placed before "import numpy
as np" (wrong-import-order)

tic_tac_toe.py:120:4: R0912: Too many branches (14/12) (too-many-branches)

tic_tac_toe.py:41:8: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code
inside it (no-else-return)
tic_tac_toe.py:155:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-
else-return)

tic_tac_toe.py:120:4: R1710: Either all return statements in a function should return an expression, or
none of them should. (inconsistent-return-statements)
tic_tac_toe.py:152:4: R1710: Either all return statements in a function should return an expression, or
none of them should. (inconsistent-return-statements)

tic_tac_toe.py:39:15: R1714: Consider merging these comparisons with "in" to "marker not in ('X', 'O')"
(consider-using-in)

tic_tac_toe.py:90:16: R1722: Consider using sys.exit() (consider-using-sys-exit)

tic_tac_toe.py:91:12: W0702: No exception type(s) specified (bare-except)

tic_tac_toe.py:169:29: W1309: Using an f-string that does not have any interpolated variables (f-string-
without-interpolation)
tic_tac_toe.py:249:29: W1309: Using an f-string that does not have any interpolated variables (f-string-
without-interpolation)

tic_tac_toe.py:95:21: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:107:13: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:168:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:217:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:225:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:248:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:285:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
tic_tac_toe.py:293:25: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

Appendix D

Additional Pylint Reports & Message Lists



```
Report
======
42 statements analysed.

Statistics by type
------------------

+---------+-------+-----------+-----------+------------+---------+
|type     |number |old number |difference |%documented |%badname |
+=========+=======+===========+===========+============+=========+
|module   |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|class    |0      |NC         |NC         |0           |0        |
+---------+-------+-----------+-----------+------------+---------+
|method   |0      |NC         |NC         |0           |0        |
+---------+-------+-----------+-----------+------------+---------+
|function |2      |2          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
```

```
52 lines have been analyzed

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |45     |86.54 |45       |=          |
+----------+-------+------+---------+-----------+
|docstring |0      |0.00  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |2      |3.85  |2        |=          |
+----------+-------+------+---------+-----------+
|empty     |5      |9.62  |5        |=          |
+----------+-------+------+---------+-----------+
```

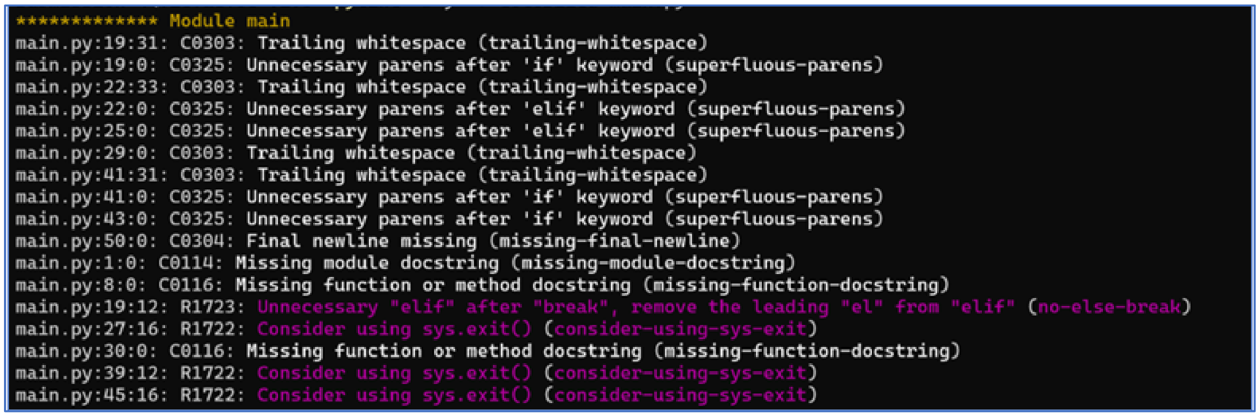
```
Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |0        |0          |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |0.000    |=          |
+-------------------------+------+---------+-----------+
```

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |13     |13       |13         |
+-----------+-------+---------+-----------+
|refactor   |4      |4        |4          |
+-----------+-------+---------+-----------+
|warning    |0      |0        |0          |
+-----------+-------+---------+-----------+
|error      |0      |0        |0          |
+-----------+-------+---------+-----------+
```

```
Messages
--------

+-------------------------+------------+
|message id               |occurrences |
+=========================+============+
|superfluous-parens       |5           |
+-------------------------+------------+
|trailing-whitespace      |4           |
+-------------------------+------------+
|consider-using-sys-exit  |3           |
+-------------------------+------------+
|missing-function-docstring |2         |
+-------------------------+------------+
|no-else-break            |1           |
+-------------------------+------------+
|missing-module-docstring |1           |
+-------------------------+------------+
|missing-final-newline    |1           |
+-------------------------+------------+
```

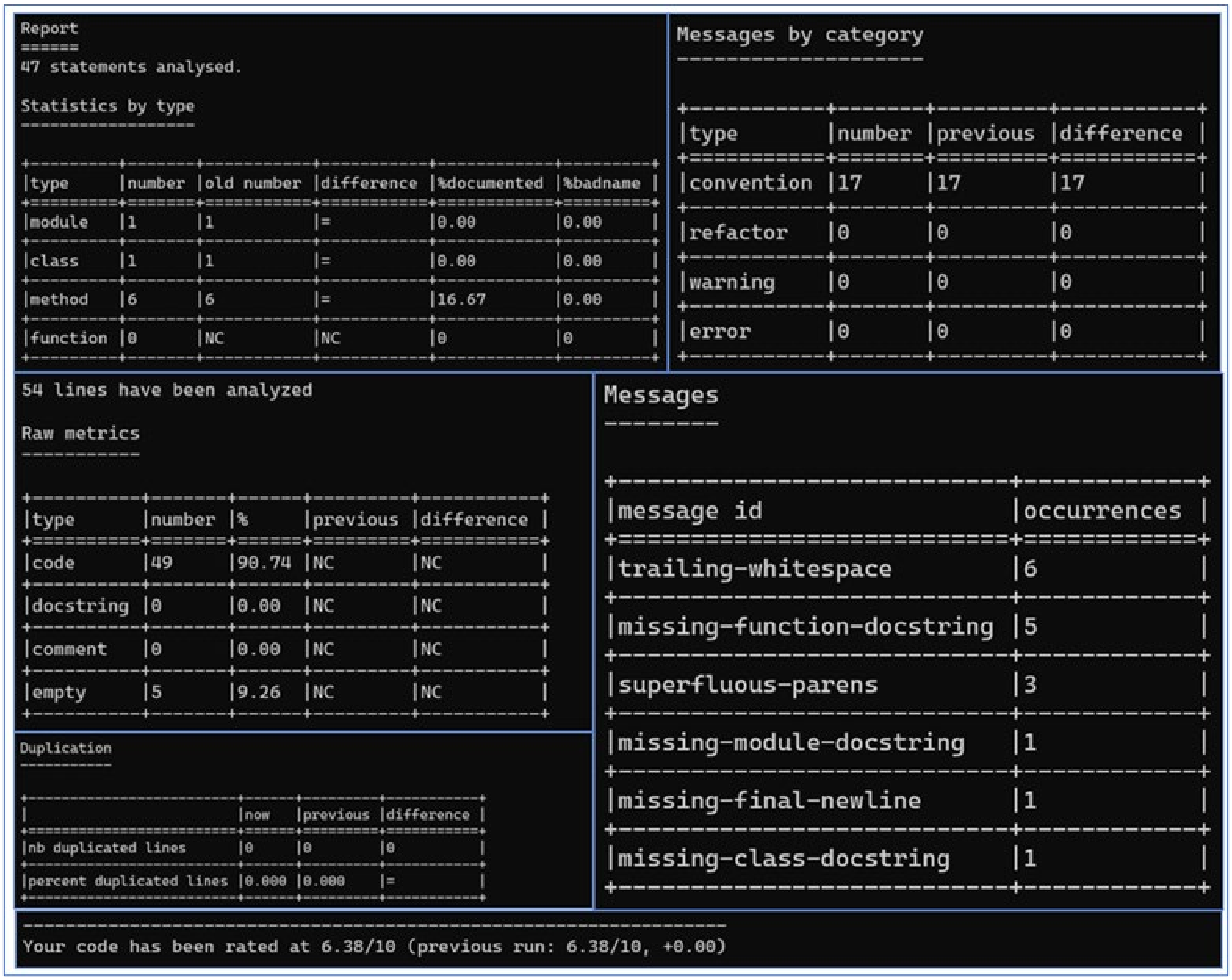
```
------------------------------------------------------------------
Your code has been rated at 5.95/10 (previous run: 5.95/10, +0.00)
```

Figure D1. *Pylint report for main module*



```
************* Module main
main.py:19:31: C0303: Trailing whitespace (trailing-whitespace)
main.py:19:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
main.py:22:33: C0303: Trailing whitespace (trailing-whitespace)
main.py:22:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
main.py:25:0: C0325: Unnecessary parens after 'elif' keyword (superfluous-parens)
main.py:29:0: C0303: Trailing whitespace (trailing-whitespace)
main.py:41:31: C0303: Trailing whitespace (trailing-whitespace)
main.py:41:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
main.py:43:0: C0325: Unnecessary parens after 'if' keyword (superfluous-parens)
main.py:50:0: C0304: Final newline missing (missing-final-newline)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:8:0: C0116: Missing function or method docstring (missing-function-docstring)
main.py:19:12: R1723: Unnecessary "elif" after "break", remove the leading "el" from "elif" (no-else-break)
main.py:27:16: R1722: Consider using sys.exit() (consider-using-sys-exit)
main.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
main.py:39:12: R1722: Consider using sys.exit() (consider-using-sys-exit)
main.py:45:16: R1722: Consider using sys.exit() (consider-using-sys-exit)
```

Figure D2. *Pylint message list for main module*



```
Report
======
47 statements analysed.

Statistics by type
------------------

+---------+-------+-----------+-----------+------------+---------+
|type     |number |old number |difference |%documented |%badname |
+=========+=======+===========+===========+============+=========+
|module   |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|class    |1      |1          |=          |0.00        |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|method   |6      |6          |=          |16.67       |0.00     |
+---------+-------+-----------+-----------+------------+---------+
|function |0      |NC         |NC         |0           |0        |
+---------+-------+-----------+-----------+------------+---------+
```

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |17     |17       |17         |
+-----------+-------+---------+-----------+
|refactor   |0      |0        |0          |
+-----------+-------+---------+-----------+
|warning    |0      |0        |0          |
+-----------+-------+---------+-----------+
|error      |0      |0        |0          |
+-----------+-------+---------+-----------+
```

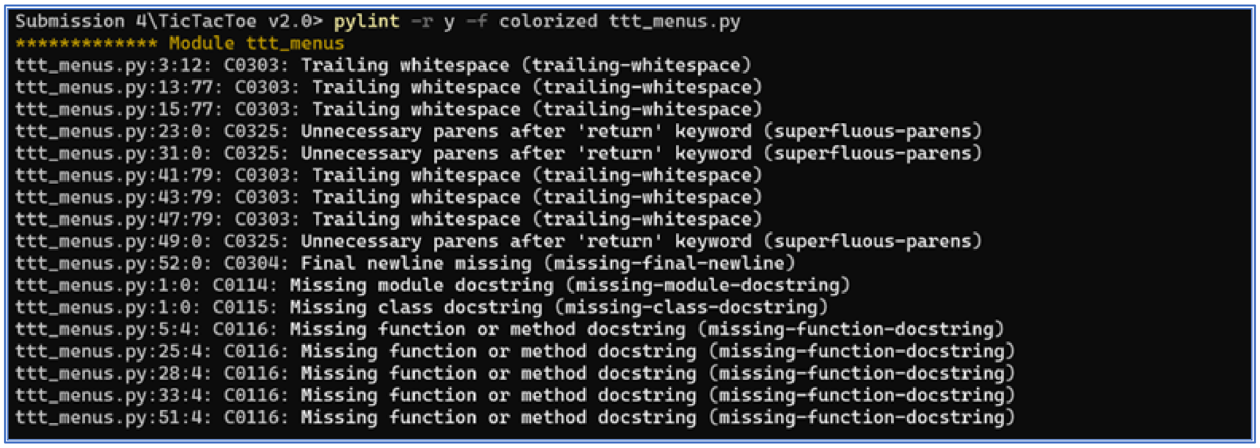
```
54 lines have been analyzed

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |49     |90.74 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |0      |0.00  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |0      |0.00  |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |5      |9.26  |NC       |NC         |
+----------+-------+------+---------+-----------+
```

```
Messages
--------

+--------------------------+------------+
|message id                |occurrences |
+==========================+============+
|trailing-whitespace       |6           |
+--------------------------+------------+
|missing-function-docstring|5           |
+--------------------------+------------+
|superfluous-parens        |3           |
+--------------------------+------------+
|missing-module-docstring  |1           |
+--------------------------+------------+
|missing-final-newline     |1           |
+--------------------------+------------+
|missing-class-docstring   |1           |
+--------------------------+------------+
```

```
Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |0        |0          |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |0.000    |=          |
+-------------------------+------+---------+-----------+
```

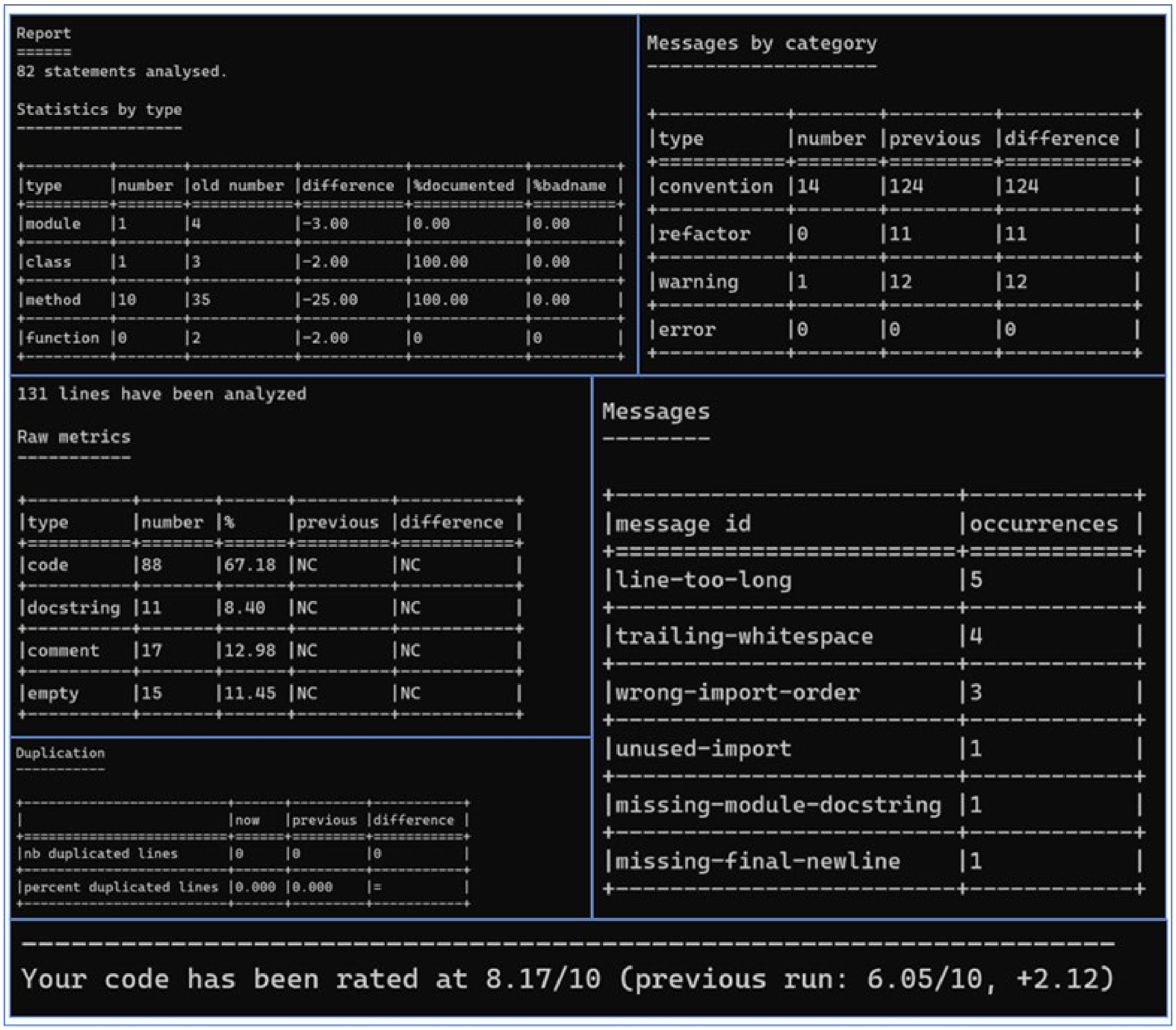
```
-------------------------------------------------------------------
Your code has been rated at 6.38/10 (previous run: 6.38/10, +0.00)
```

Figure D3. *Pylint report for ttt_menus module*



```
Submission 4\TicTacToe v2.0> pylint -r y -f colorized ttt_menus.py
************* Module ttt_menus
ttt_menus.py:3:12: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:13:77: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:15:77: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:23:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
ttt_menus.py:31:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
ttt_menus.py:41:79: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:43:79: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:47:79: C0303: Trailing whitespace (trailing-whitespace)
ttt_menus.py:49:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
ttt_menus.py:52:0: C0304: Final newline missing (missing-final-newline)
ttt_menus.py:1:0: C0114: Missing module docstring (missing-module-docstring)
ttt_menus.py:1:0: C0115: Missing class docstring (missing-class-docstring)
ttt_menus.py:5:4: C0116: Missing function or method docstring (missing-function-docstring)
ttt_menus.py:25:4: C0116: Missing function or method docstring (missing-function-docstring)
ttt_menus.py:28:4: C0116: Missing function or method docstring (missing-function-docstring)
ttt_menus.py:33:4: C0116: Missing function or method docstring (missing-function-docstring)
ttt_menus.py:51:4: C0116: Missing function or method docstring (missing-function-docstring)
```

Figure D4. *Pylint message list for ttt_menus module*



```
Report
======
82 statements analysed.

Statistics by type
------------------

+---------+-------+-----------+----------+-----------+--------+
|type     |number |old number |difference|%documented|%badname|
+=========+=======+===========+==========+===========+========+
|module   |1      |4          |-3.00     |0.00       |0.00    |
+---------+-------+-----------+----------+-----------+--------+
|class    |1      |3          |-2.00     |100.00     |0.00    |
+---------+-------+-----------+----------+-----------+--------+
|method   |10     |35         |-25.00    |100.00     |0.00    |
+---------+-------+-----------+----------+-----------+--------+
|function |0      |2          |-2.00     |0          |0       |
+---------+-------+-----------+----------+-----------+--------+
```

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |14     |124      |124        |
+-----------+-------+---------+-----------+
|refactor   |0      |11       |11         |
+-----------+-------+---------+-----------+
|warning    |1      |12       |12         |
+-----------+-------+---------+-----------+
|error      |0      |0        |0          |
+-----------+-------+---------+-----------+
```

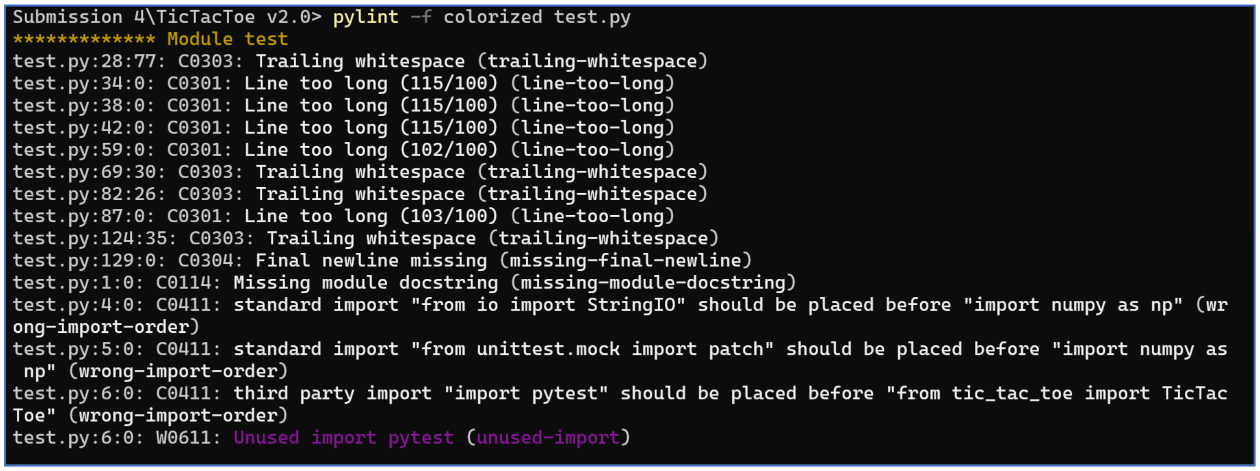
```
131 lines have been analyzed

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |88     |67.18 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |11     |8.40  |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |17     |12.98 |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |15     |11.45 |NC       |NC         |
+----------+-------+------+---------+-----------+
```

```
Messages
--------

+------------------------+------------+
|message id              |occurrences |
+========================+============+
|line-too-long           |5           |
+------------------------+------------+
|trailing-whitespace     |4           |
+------------------------+------------+
|wrong-import-order      |3           |
+------------------------+------------+
|unused-import           |1           |
+------------------------+------------+
|missing-module-docstring|1           |
+------------------------+------------+
|missing-final-newline   |1           |
+------------------------+------------+
```

```
Duplication
-----------

+-------------------------+------+---------+-----------+
|                         |now   |previous |difference |
+=========================+======+=========+===========+
|nb duplicated lines      |0     |0        |0          |
+-------------------------+------+---------+-----------+
|percent duplicated lines |0.000 |0.000    |=          |
+-------------------------+------+---------+-----------+
```

```
------------------------------------------------------------------
Your code has been rated at 8.17/10 (previous run: 6.05/10, +2.12)
```

Figure D5. *Pylint report for test module*



```
Submission 4\TicTacToe v2.0> pylint -f colorized test.py
************* Module test
test.py:28:77: C0303: Trailing whitespace (trailing-whitespace)
test.py:34:0: C0301: Line too long (115/100) (line-too-long)
test.py:38:0: C0301: Line too long (115/100) (line-too-long)
test.py:42:0: C0301: Line too long (115/100) (line-too-long)
test.py:59:0: C0301: Line too long (102/100) (line-too-long)
test.py:69:30: C0303: Trailing whitespace (trailing-whitespace)
test.py:82:26: C0303: Trailing whitespace (trailing-whitespace)
test.py:87:0: C0301: Line too long (103/100) (line-too-long)
test.py:124:35: C0303: Trailing whitespace (trailing-whitespace)
test.py:129:0: C0304: Final newline missing (missing-final-newline)
test.py:1:0: C0114: Missing module docstring (missing-module-docstring)
test.py:4:0: C0411: standard import "from io import StringIO" should be placed before "import numpy as np" (wr
ong-import-order)
test.py:5:0: C0411: standard import "from unittest.mock import patch" should be placed before "import numpy as
 np" (wrong-import-order)
test.py:6:0: C0411: third party import "import pytest" should be placed before "from tic_tac_toe import TicTac
Toe" (wrong-import-order)
test.py:6:0: W0611: Unused import pytest (unused-import)
```

Figure D6. *Pylint message list for test module*

Appendix E

Pylama Results After Fixing Pylint Messages

```
main.py:8:1 E302 expected 2 blank lines, found 1 [pycodestyle]
main.py:19:32 W291 trailing whitespace [pycodestyle]
main.py:22:34 W291 trailing whitespace [pycodestyle]
main.py:29:1 W293 blank line contains whitespace [pycodestyle]
main.py:30:1 E302 expected 2 blank lines, found 1 [pycodestyle]
main.py:41:32 W291 trailing whitespace [pycodestyle]
main.py:48:1 E305 expected 2 blank lines after class or function definition, found 1 [pycodestyle]
main.py:50:12 W292 no newline at end of file [pycodestyle]

test.py:6:1 W0611 'pytest' imported but unused [pyflakes]
test.py:12:1 E302 expected 2 blank lines, found 1 [pycodestyle]
test.py:18:27 E261 at least two spaces before inline comment [pycodestyle]
test.py:20:48 E261 at least two spaces before inline comment [pycodestyle]
test.py:28:78 W291 trailing whitespace [pycodestyle]
test.py:28:32 E261 at least two spaces before inline comment [pycodestyle]
test.py:32:23 E222 multiple spaces after operator [pycodestyle]
test.py:33:24 E222 multiple spaces after operator [pycodestyle]
test.py:34:101 E501 line too long (115 > 100 characters) [pycodestyle]
test.py:35:24 E222 multiple spaces after operator [pycodestyle]
test.py:36:24 E222 multiple spaces after operator [pycodestyle]
test.py:37:24 E222 multiple spaces after operator [pycodestyle]
test.py:38:101 E501 line too long (115 > 100 characters) [pycodestyle]
test.py:39:24 E222 multiple spaces after operator [pycodestyle]
test.py:40:24 E222 multiple spaces after operator [pycodestyle]
test.py:41:24 E222 multiple spaces after operator [pycodestyle]
test.py:42:101 E501 line too long (115 > 100 characters) [pycodestyle]
test.py:43:24 E222 multiple spaces after operator [pycodestyle]
test.py:44:24 E222 multiple spaces after operator [pycodestyle]
test.py:51:26 E261 at least two spaces before inline comment [pycodestyle]
test.py:52:24 E261 at least two spaces before inline comment [pycodestyle]
test.py:56:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:59:101 E501 line too long (102 > 100 characters) [pycodestyle]
test.py:60:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:61:37 E261 at least two spaces before inline comment [pycodestyle]
test.py:62:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:64:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:65:37 E261 at least two spaces before inline comment [pycodestyle]
test.py:66:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:68:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:69:31 W291 trailing whitespace [pycodestyle]
test.py:72:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:73:68 E261 at least two spaces before inline comment [pycodestyle]
test.py:74:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:75:46 E261 at least two spaces before inline comment [pycodestyle]
test.py:77:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:78:68 E261 at least two spaces before inline comment [pycodestyle]
test.py:79:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:80:46 E261 at least two spaces before inline comment [pycodestyle]
test.py:82:27 W291 trailing whitespace [pycodestyle]
test.py:85:41 E261 at least two spaces before inline comment [pycodestyle]
test.py:86:35 E261 at least two spaces before inline comment [pycodestyle]
test.py:87:101 E501 line too long (103 > 100 characters) [pycodestyle]
test.py:87:57 E261 at least two spaces before inline comment [pycodestyle]
test.py:89:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:94:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:95:25 E261 at least two spaces before inline comment [pycodestyle]
test.py:96:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:97:52 E261 at least two spaces before inline comment [pycodestyle]
test.py:98:38 E261 at least two spaces before inline comment [pycodestyle]
test.py:99:65 E261 at least two spaces before inline comment [pycodestyle]
test.py:101:53 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
test.py:106:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:107:25 E261 at least two spaces before inline comment [pycodestyle]
test.py:108:45 E261 at least two spaces before inline comment [pycodestyle]
test.py:109:44 E261 at least two spaces before inline comment [pycodestyle]
test.py:110:49 E261 at least two spaces before inline comment [pycodestyle]
test.py:111:65 E261 at least two spaces before inline comment [pycodestyle]
test.py:121:5 E265 block comment should start with '# ' [pycodestyle]
test.py:124:36 W291 trailing whitespace [pycodestyle]
test.py:128:1 E305 expected 2 blank lines after class or function definition, found 1 [pycodestyle]
test.py:129:21 W292 no newline at end of file [pycodestyle]
test.py:129:20 W291 trailing whitespace [pycodestyle]

tic_tac_toe.py:8:1 E302 expected 2 blank lines, found 1 [pycodestyle]
tic_tac_toe.py:12:24 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:13:68 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:14:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:18:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:19:33 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:20:52 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:27:29 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
tic_tac_toe.py:29:46 E251 unexpected spaces around keyword / parameter equals [pycodestyle]
tic_tac_toe.py:36:24 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:40:21 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:41:40 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:42:71 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:43:26 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:44:30 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:45:26 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:47:30 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:49:48 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:50:32 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:52:25 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:54:36 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:57:5 E303 too many blank lines (2) [pycodestyle]
tic_tac_toe.py:59:101 E501 line too long (101 > 100 characters) [pycodestyle]
tic_tac_toe.py:60:47 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:61:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:65:40 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:66:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:67:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:68:22 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:72:28 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:73:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:74:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:75:42 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:76:50 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:77:45 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:78:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:82:38 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:86:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:87:17 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:88:49 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:89:58 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:90:38 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:91:41 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:92:23 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:93:13 E722 do not use bare 'except' [pycodestyle]
tic_tac_toe.py:93:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:94:53 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:95:18 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:96:44 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:97:62 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:98:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:99:52 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:100:29 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:104:35 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:105:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:106:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:107:17 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:108:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:109:54 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:110:46 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:111:44 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:115:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:116:31 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:117:42 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:118:33 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:119:20 E261 at least two spaces before inline comment [pycodestyle]
tic_tac_toe.py:121:5 C901 'TicTacToe.is_winner' is too complex (15) [mccabe]
tic_tac_toe.py:240:30 E712 comparison to False should be 'if cond is False:' or 'if not cond:'
[pycodestyle]
tic_tac_toe.py:307:30 E712 comparison to False should be 'if cond is False:' or 'if not cond:'
[pycodestyle]

ttt_menus.py:3:13 W291 trailing whitespace [pycodestyle]
ttt_menus.py:6:18 E222 multiple spaces after operator [pycodestyle]
ttt_menus.py:13:78 W291 trailing whitespace [pycodestyle]
ttt_menus.py:15:78 W291 trailing whitespace [pycodestyle]
ttt_menus.py:34:20 E222 multiple spaces after operator [pycodestyle]
ttt_menus.py:41:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:43:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:47:80 W291 trailing whitespace [pycodestyle]
ttt_menus.py:52:46 W292 no newline at end of file [pycodestyle]
```