



IT314

Software Engineering

(Lab - 5)

Name : Deep Kanani

ID : 202001454

Date : 24/03/23

Static Analysis

Static Analysis Tool: Pylint

S.No	Message Object	Expansion	Explanation
1.	C	Convention	It is displayed when the program is not following the standard rules.
2.	R	Refactor	It is displayed for bad code smell
3.	W	Warning	It is displayed for python specific problems
4.	E	Error	It is displayed when that particular line execution results some error
5.	F	Fatal	It is displayed when pylint has no access to further process that line.

Reference : [pylint](#)

1. average_mode

Code :

```
1  from typing import Any
2
3
4  def mode(input_list: list) -> list[Any]:
5      """This function returns the mode(Mode as in the measures of
6      central tendency) of the input data.
7
8      The input list may contain any Datastructure or any Datatype.
9
10     >>> mode([2, 3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 2, 2, 2])
11     [2]
12     >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 2, 2, 2])
13     [2]
14     >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 4, 2, 2, 4, 2])
15     [2, 4]
16     >>> mode(["x", "y", "y", "z"])
17     ['y']
18     >>> mode(["x", "x", "y", "y", "z"])
19     ['x', 'y']
20     """
21     if not input_list:
22         return []
23     result = [input_list.count(value) for value in input_list]
24     y = max(result) # Gets the maximum count in the input list.
25     # Gets values of modes
26     return sorted({input_list[i] for i, value in enumerate(result) if value == y})
27
28
29 if __name__ == "__main__":
30     import doctest
31
32     doctest.testmod()
```

Output :

```
PS C:\Users\student\Pictures\l5\Salmon_Exclusive> py -m pylint average_mode.py
***** Module average_mode
average_mode.py:32:0: C0304: Final newline missing (missing-final-newline)
average_mode.py:1:0: C0114: Missing module docstring (missing-module-docstring)
average_mode.py:24:4: C0103: Variable name "y" doesn't conform to snake_case naming style (invalid-name)

-----
Your code has been rated at 7.00/10 (previous run: 7.00/10, +0.00)
```

Error Analysis :

- **Missing final newline.**
 - We have to add a new line when our code is complete.
- **Missing module docstring**
 - At the start of the code we add a string(comment) which indicates the use of our programme.
- **Snake_case naming style**
 - It shows that we have to name our variable in proper format.

Improved Code :

```
1  "Program to Calculate Mode"
2
3  from typing import Any
4
5
6  def mode(input_list: list) -> list[Any]:
7      """This function returns the mode(Mode as in the measures of
8      central tendency) of the input data.
9
10     The input list may contain any Datastructure or any Datatype.
11
12     >>> mode([2, 3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 2, 2, 2])
13     [2]
14     >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 2, 2, 2])
15     [2]
16     >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 4, 2, 2, 4, 2])
17     [2, 4]
18     >>> mode(["x", "y", "y", "z"])
19     ['y']
20     >>> mode(["x", "x", "y", "y", "z"])
21     ['x', 'y']
22     """
23     if not input_list:
24         return []
25     result = [input_list.count(value) for value in input_list]
26     answer = max(result) # Gets the maximum count in the input list.
27     # Gets values of modes
28     return sorted({input_list[i] for i, value in enumerate(result) if value == answer})
29
30
31 if __name__ == "__main__":
32     import doctest
33
34     doctest.testmod()
35
```

Output :

```
PS C:\Users\student\Pictures\l5\Salmon_Exclusive> py -m pylint average_mode.py
-----
Your code has been rated at 10.00/10 (previous run: 7.00/10, +3.00)
```

2. arc_length

Code :

```
1  from math import pi
2
3
4  def arc_length(angle: int, radius: int) -> float:
5      """
6          >>> arc_length(45, 5)
7          3.9269908169872414
8          >>> arc_length(120, 15)
9          31.415926535897928
10         """
11         return 2 * pi * radius * (angle / 360)
12
13
14 if __name__ == "__main__":
15     print(arc_length(90, 10))
16
17
18
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint arc.py
***** Module arc
arc.py:16:0: C0303: Trailing whitespace (trailing-whitespace)
arc.py:17:0: C0305: Trailing newlines (trailing-newlines)
arc.py:1:0: C0114: Missing module docstring (missing-module-docstring)

-----
Your code has been rated at 4.00/10 (previous run: 4.00/10, +0.00)
```

Error Analysis :

- **Trailing newline.**
 - We have to remove new lines.
- **Missing module docstring**
 - At the start of the code we add a string(comment) which indicates the use of our programme.
- **Trailing Whitespace**
 - Remove the redundant spaces.

Improved Code :

```
arc.py > ...
1  "Calculating length of arc"
2
3  from math import pi
4
5
6  def arc_length(angle: int, radius: int) -> float:
7      """
8          >>> arc_length(45, 5)
9              3.9269908169872414
10         >>> arc_length(120, 15)
11             31.415926535897928
12         """
13         return 2 * pi * radius * (angle / 360)
14  if __name__ == "__main__":
15      print(arc_length(90, 10))
16
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint arc.py
-----
Your code has been rated at 10.00/10 (previous run: 4.00/10, +6.00)
PS C:\Users\student\Pictures\IT314> █
```

3. ceil

Code :

```
ceil.py > ceil
1  def ceil(x: float) -> int:
2      """
3          Return the ceiling of x as an Integral.
4          :param x: the number
5          :return: the smallest integer >= x.
6          >>> import math
7          >>> all(ceil(n) == math.ceil(n) for n
8              ...     in (1, -1, 0, -0, 1.1, -1.1, 1.0, -1.0, 1_000_000_000))
9          True
10         """
11         return int(x) if x - int(x) <= 0 else int(x) + 1
12
13
14  if __name__ == "__main__":
15      import doctest
16
17      doctest.testmod()
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint ceil.py
***** Module ceil
ceil.py:17:0: C0304: Final newline missing (missing-final-newline)
ceil.py:1:0: C0114: Missing module docstring (missing-module-docstring)
ceil.py:1:9: C0103: Argument name "x" doesn't conform to snake_case naming style (invalid-name)

-----
Your code has been rated at 4.00/10 (previous run: 4.00/10, +0.00)

PS C:\Users\student\Pictures\IT314> 
```


Error Analysis :

- **Missing final newline.**
 - We have to add a new line when our code is complete.
- **Missing module docstring**
 - At the start of the code we add a string(comment) which indicates the use of our programme.
- **Snake_case naming style**
 - It shows that we have to name our variable in proper format.

Improved Code :

```
ceil.py > ...
1  "Calculating ceil of a number"
2
3  def ceil(num: float) -> int:
4      """
5      Return the ceiling of x as an Integral.
6      :param x: the number
7      :return: the smallest integer >= x.
8      >>> import math
9      >>> all(ceil(n) == math.ceil(n) for n
10         ...     in (1, -1, 0, -0, 1.1, -1.1, 1.0, -1.0, 1_000_000_000))
11         True
12         """
13         return int(num) if num - int(num) <= 0 else int(num) + 1
14
15
16 if __name__ == "__main__":
17     import doctest
18
19     doctest.testmod()
20
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint ceil.py

-----
Your code has been rated at 10.00/10 (previous run: 4.00/10, +6.00)

PS C:\Users\student\Pictures\IT314> █
```

4. average_mean

Code :

```
1  from __future__ import annotations
2
3
4  def mean(nums: list) -> float:
5      """
6      Find mean of a list of numbers.
7      Wiki: https://en.wikipedia.org/wiki/Mean
8
9      >>> mean([3, 6, 9, 12, 15, 18, 21])
10     12.0
11     >>> mean([5, 10, 15, 20, 25, 30, 35])
12     20.0
13     >>> mean([1, 2, 3, 4, 5, 6, 7, 8])
14     4.5
15     >>> mean([])
16     Traceback (most recent call last):
17     |     ...
18     ValueError: List is empty
19     """
20     if not nums:
21         raise ValueError("List is empty")
22     return sum(nums) / len(nums)
23
24
25 if __name__ == "__main__":
26     import doctest
27
28     doctest.testmod()
```

Output :

```
PS C:\Users\student\Pictures\15\Salmon_Exclusive> py -m pylint average_mean.py
***** Module average_mean
average_mean.py:28:0: C0304: Final newline missing (missing-final-newline)
average_mean.py:1:0: C0114: Missing module docstring (missing-module-docstring)

-----
Your code has been rated at 7.50/10
```

Error Analysis :

- **Missing final newline.**
 - We have to add a new line when our code is complete.
- **Missing module docstring**
 - At the start of the code we add a string(comment) which indicates the use of our programme.

Improved Code :

```
1  "Program to Calculate Average"
2
3  from __future__ import annotations
4
5
6  def mean(nums: list) -> float:
7      """
8      Find mean of a list of numbers.
9      Wiki: https://en.wikipedia.org/wiki/Mean
10
11      >>> mean([3, 6, 9, 12, 15, 18, 21])
12      12.0
13      >>> mean([5, 10, 15, 20, 25, 30, 35])
14      20.0
15      >>> mean([1, 2, 3, 4, 5, 6, 7, 8])
16      4.5
17      >>> mean([])
18      Traceback (most recent call last):
19      | ...
20      ValueError: List is empty
21      """
22      if not nums:
23          raise ValueError("List is empty")
24      return sum(nums) / len(nums)
25
26
27  if __name__ == "__main__":
28      import doctest
29
30      doctest.testmod()
31
```

Output :

```
PS C:\Users\student\Pictures\15\Salmon_Exclusive> py -m pylint average_mean.py
-----
Your code has been rated at 10.00/10 (previous run: 7.50/10, +2.50)
```

5. find_min

Code :

```
1  from __future__ import annotations
2
3
4  def find_min(nums: list[int | float]) -> int | float:
5      """
6      Find Minimum Number in a List
7      :param nums: contains elements
8      :return: min number in list
9      >>> for nums in ([3, 2, 1], [-3, -2, -1], [3, -3, 0], [3.0, 3.1, 2.9]):
10         ...     find_min(nums) == min(nums)
11         True
12         True
13         True
14         True
15         >>> find_min([0, 1, 2, 3, 4, 5, -3, 24, -56])
16         -56
17         >>> find_min([])
18         Traceback (most recent call last):
19         |     ...
20         | ValueError: find_min() arg is an empty sequence
21         """
22         if len(nums) == 0:
23             raise ValueError("find_min() arg is an empty sequence")
24         min_num = nums[0]
25         for num in nums:
26             min_num = min(min_num, num)
27         return min_num
28
29
30 if __name__ == "__main__":
31     import doctest
32
33     doctest.testmod(verbose=True)
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint find_min.py
***** Module find_min
find_min.py:33:0: C0304: Final newline missing (missing-final-newline)
find_min.py:1:0: C0114: Missing module docstring (missing-module-docstring)

-----
Your code has been rated at 8.18/10 (previous run: 8.18/10, +0.00)

PS C:\Users\student\Pictures\IT314> █
```

Error Analysis :

- **Missing final newline.**
 - We have to add a new line when our code is complete.
- **Missing module docstring**
 - At the start of the code we add a string(comment) which indicates the use of our programme.

Improved Code :

```
find_min.py > find_min
1  "Finding minimum"
2
3  from __future__ import annotations
4
5
6  def find_min(nums: list[int | float]) -> int | float:
7      """
8      Find Minimum Number in a List
9      :param nums: contains elements
10     :return: min number in list
11     >>> for nums in ([3, 2, 1], [-3, -2, -1], [3, -3, 0], [3.0, 3.1, 2.9]):
12         ...     find_min(nums) == min(nums)
13         True
14         True
15         True
16         True
17     >>> find_min([0, 1, 2, 3, 4, 5, -3, 24, -56])
18     -56
19     >>> find_min([])
20     Traceback (most recent call last):
21     |     ...
22     ValueError: find_min() arg is an empty sequence
23     """
24     if len(nums) == 0:
25         raise ValueError["find_min() arg is an empty sequence"]
26     min_num = nums[0]
27     for num in nums:
28         min_num = min(min_num, num)
29     return min_num
30
31
32 if __name__ == "__main__":
33     import doctest
34
35     doctest.testmod(verbose=True)
36
```

Output :

```
PS C:\Users\student\Pictures\IT314> py -m pylint find_min.py
```

```
-----  
Your code has been rated at 10.00/10 (previous run: 8.18/10, +1.82)
```

```
PS C:\Users\student\Pictures\IT314> █
```