IT314

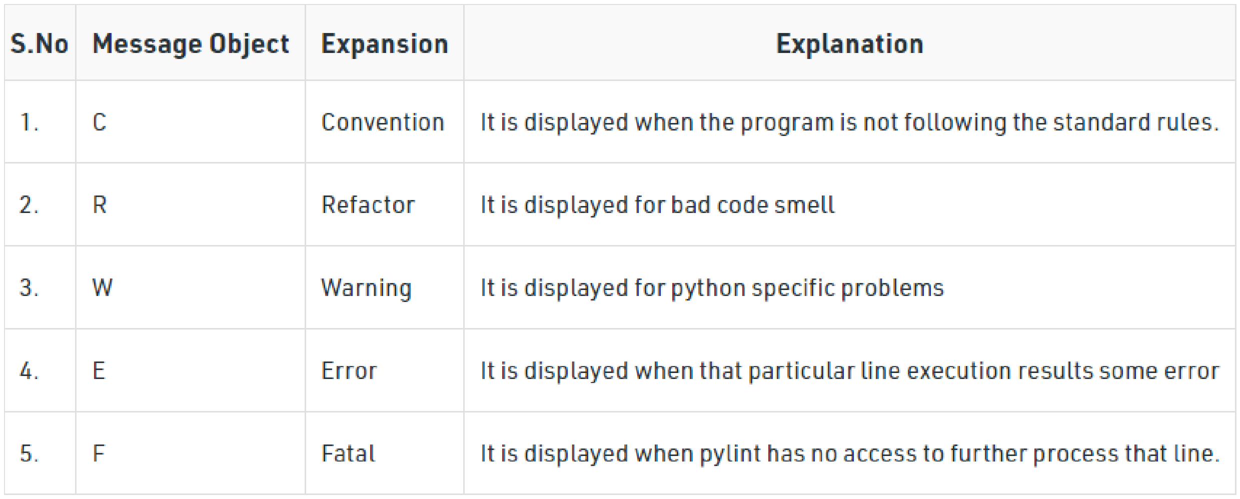
Software Engineering

(Lab - 5)

Name : Deep Kanani
ID : 202001454
Date : 24/03/23

Static Analysis

Static Analysis Tool : **Pylint**

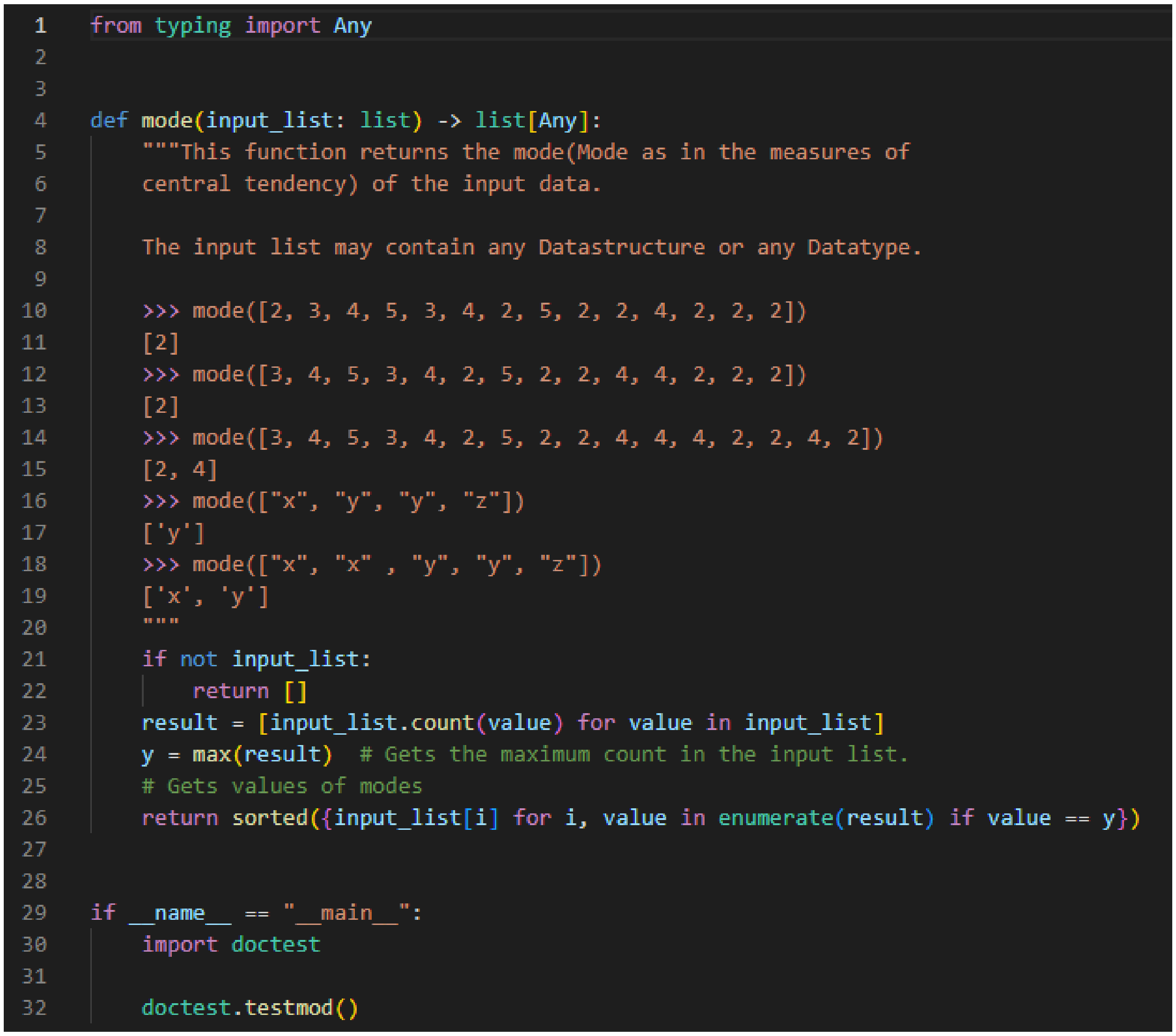


S.No	Message Object	Expansion	Explanation
1.	C	Convention	It is displayed when the program is not following the standard rules.
2.	R	Refactor	It is displayed for bad code smell
3.	W	Warning	It is displayed for python specific problems
4.	E	Error	It is displayed when that particular line execution results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

Reference : pylint

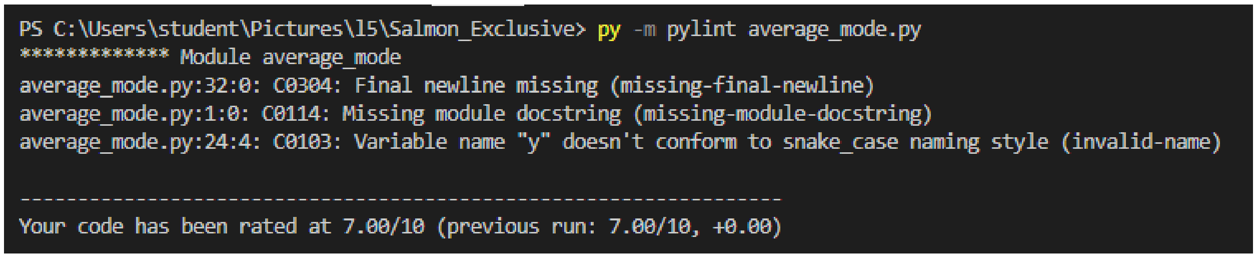
1. average_mode

Code :



```python
from typing import Any


def mode(input_list: list) -> list[Any]:
    """This function returns the mode(Mode as in the measures of
    central tendency) of the input data.

    The input list may contain any Datastructure or any Datatype.

    >>> mode([2, 3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 2, 2, 2])
    [2]
    >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 2, 2, 2])
    [2]
    >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 4, 2, 2, 4, 2])
    [2, 4]
    >>> mode(["x", "y", "y", "z"])
    ['y']
    >>> mode(["x", "x" , "y", "y", "z"])
    ['x', 'y']
    """
    if not input_list:
        return []
    result = [input_list.count(value) for value in input_list]
    y = max(result)  # Gets the maximum count in the input list.
    # Gets values of modes
    return sorted({input_list[i] for i, value in enumerate(result) if value == y})


if __name__ == "__main__":
    import doctest

    doctest.testmod()
```

Output :

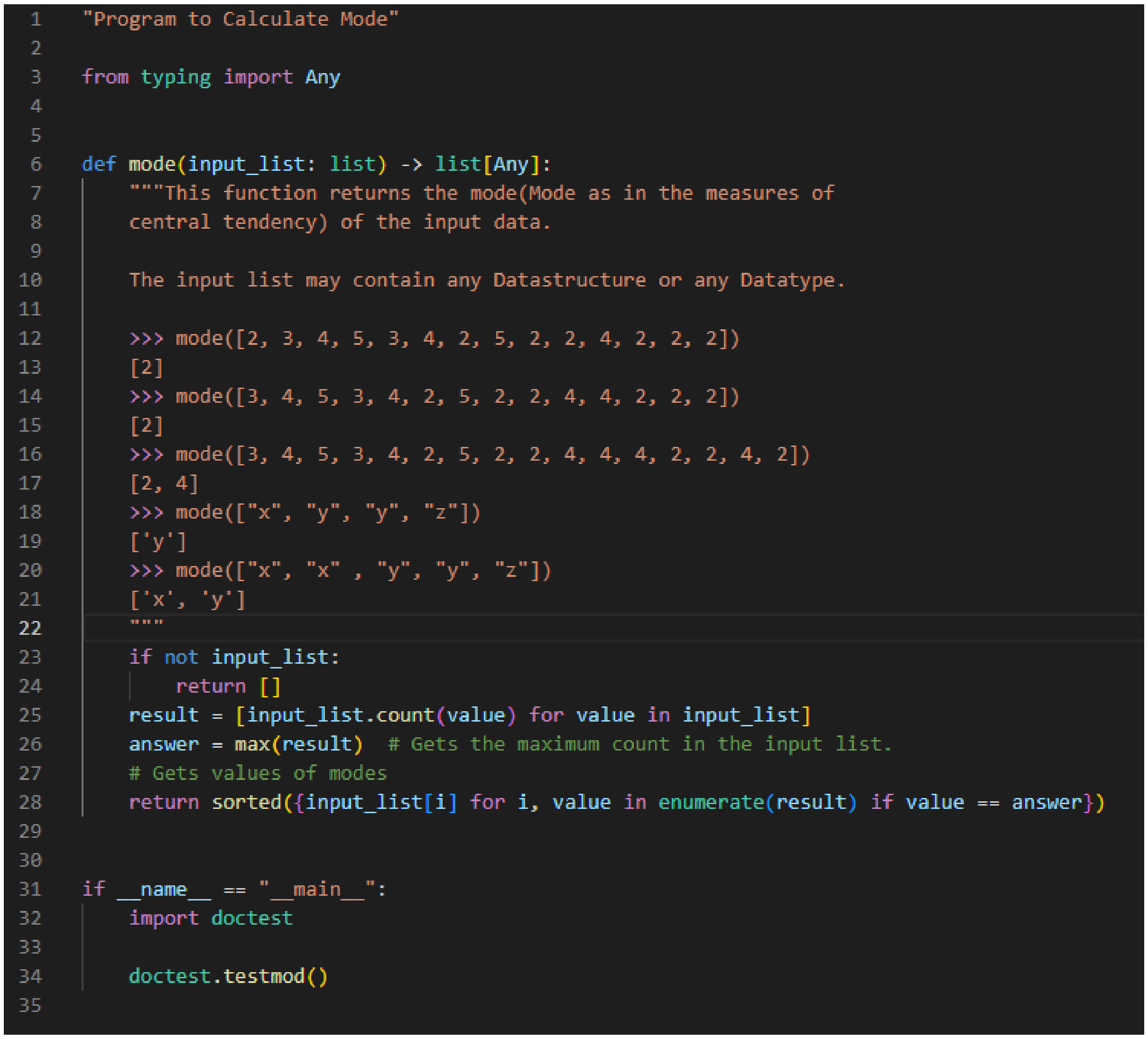


```
PS C:\Users\student\Pictures\15\Salmon_Exclusive> py -m pylint average_mode.py
************* Module average_mode
average_mode.py:32:0: C0304: Final newline missing (missing-final-newline)
average_mode.py:1:0: C0114: Missing module docstring (missing-module-docstring)
average_mode.py:24:4: C0103: 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 :

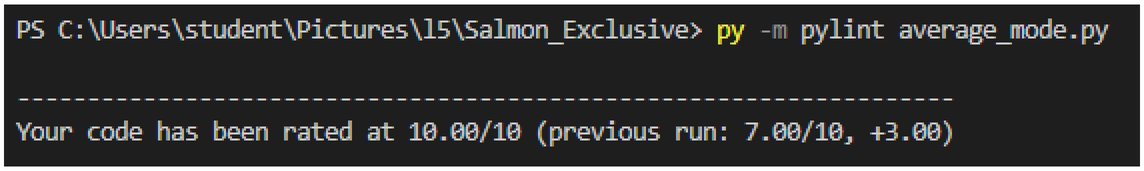
- **Missing final newline.**
  - We have to add a new line when our code is complete.
- **Missing module docstring**
  - At the start of the code we add a string(comment) which indicates the use of our programme.
- **Snake_case naming style**
  - It shows that we have to name our variable in proper format.

Improved Code :



```python
"Program to Calculate Mode"

from typing import Any


def mode(input_list: list) -> list[Any]:
    """This function returns the mode(Mode as in the measures of
    central tendency) of the input data.

    The input list may contain any Datastructure or any Datatype.

    >>> mode([2, 3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 2, 2, 2])
    [2]
    >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 2, 2, 2])
    [2]
    >>> mode([3, 4, 5, 3, 4, 2, 5, 2, 2, 4, 4, 4, 2, 2, 4, 2])
    [2, 4]
    >>> mode(["x", "y", "y", "z"])
    ['y']
    >>> mode(["x", "x" , "y", "y", "z"])
    ['x', 'y']
    """
    if not input_list:
        return []
    result = [input_list.count(value) for value in input_list]
    answer = max(result)  # Gets the maximum count in the input list.
    # Gets values of modes
    return sorted({input_list[i] for i, value in enumerate(result) if value == answer})


if __name__ == "__main__":
    import doctest

    doctest.testmod()

```

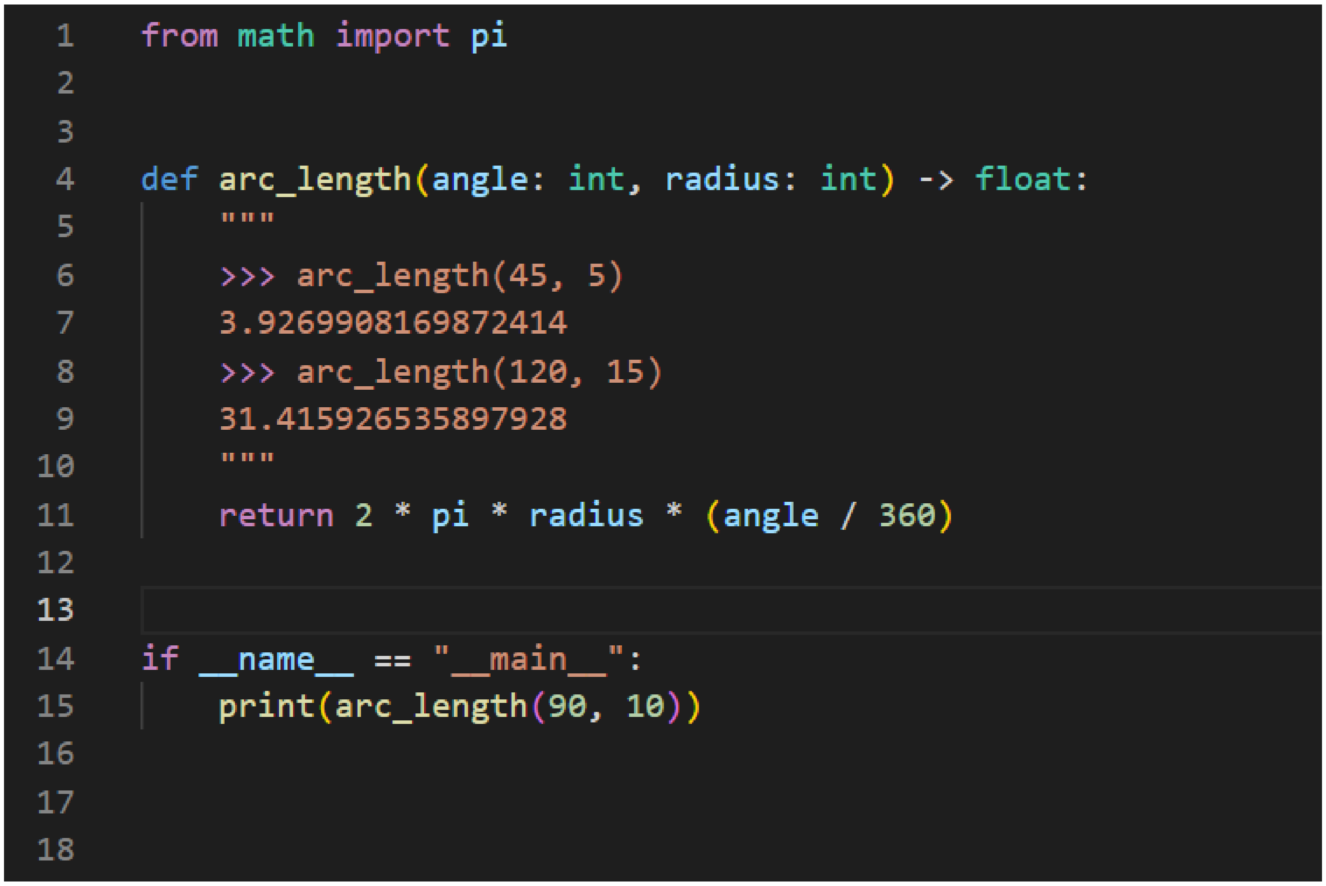
Output :



```
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)
```

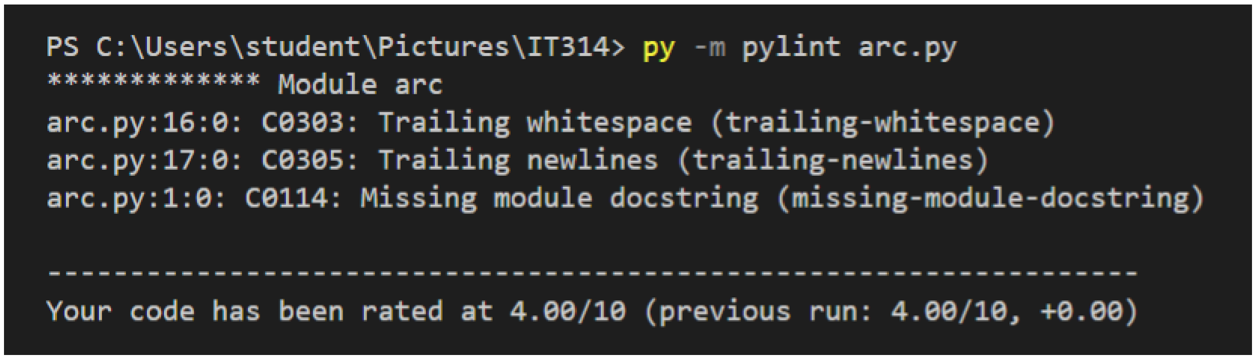
2. arc_length

Code :



```python
from math import pi


def arc_length(angle: int, radius: int) -> float:
    """
    >>> arc_length(45, 5)
    3.9269908169872414
    >>> arc_length(120, 15)
    31.415926535897928
    """
    return 2 * pi * radius * (angle / 360)


if __name__ == "__main__":
    print(arc_length(90, 10))



```

Output :

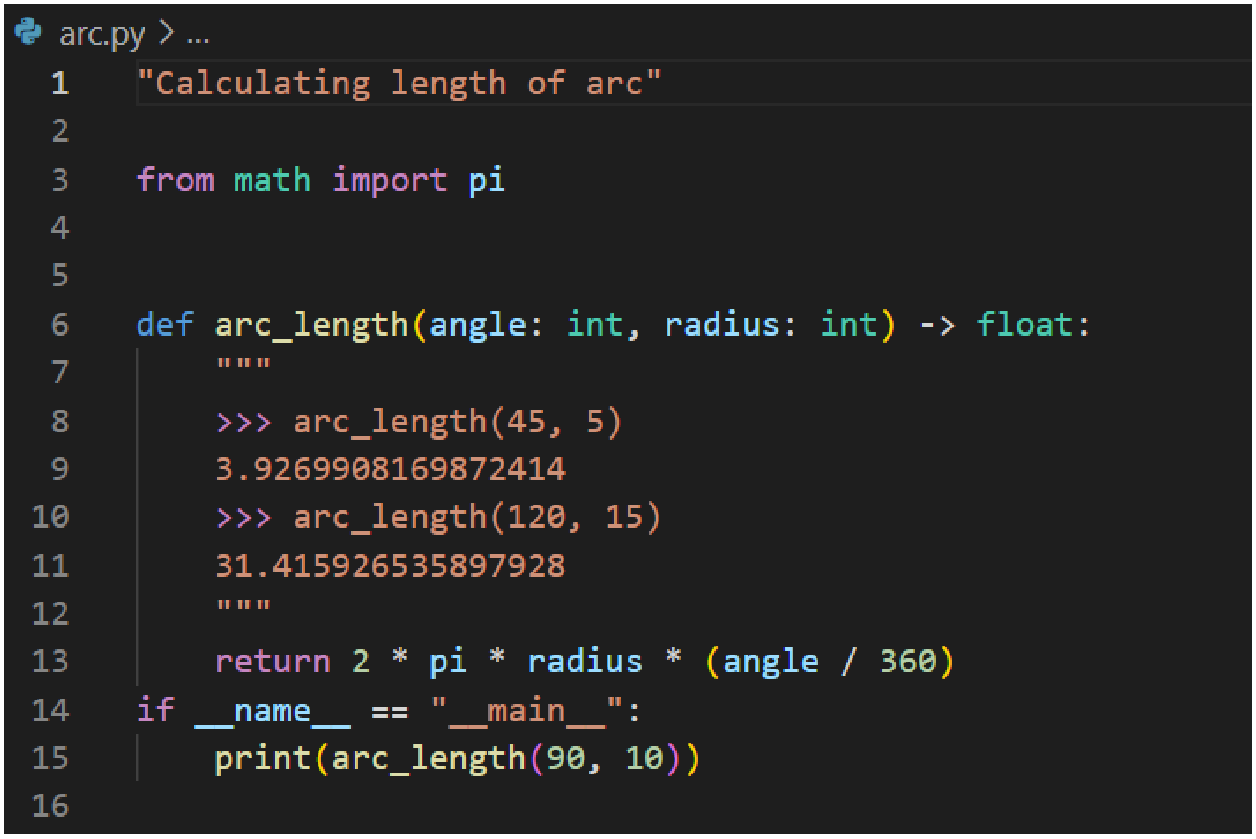


```
PS C:\Users\student\Pictures\IT314> py -m pylint arc.py
************* Module arc
arc.py:16:0: C0303: Trailing whitespace (trailing-whitespace)
arc.py:17:0: C0305: Trailing newlines (trailing-newlines)
arc.py:1:0: C0114: Missing module docstring (missing-module-docstring)

------------------------------------------------------------------
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.

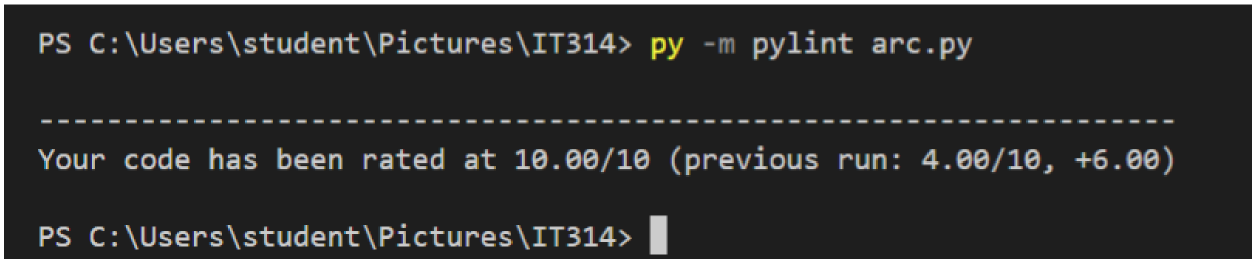
Improved Code :



arc.py > ...

```python
"Calculating length of arc"

from math import pi


def arc_length(angle: int, radius: int) -> float:
    """
    >>> arc_length(45, 5)
    3.9269908169872414
    >>> arc_length(120, 15)
    31.415926535897928
    """
    return 2 * pi * radius * (angle / 360)
if __name__ == "__main__":
    print(arc_length(90, 10))

```

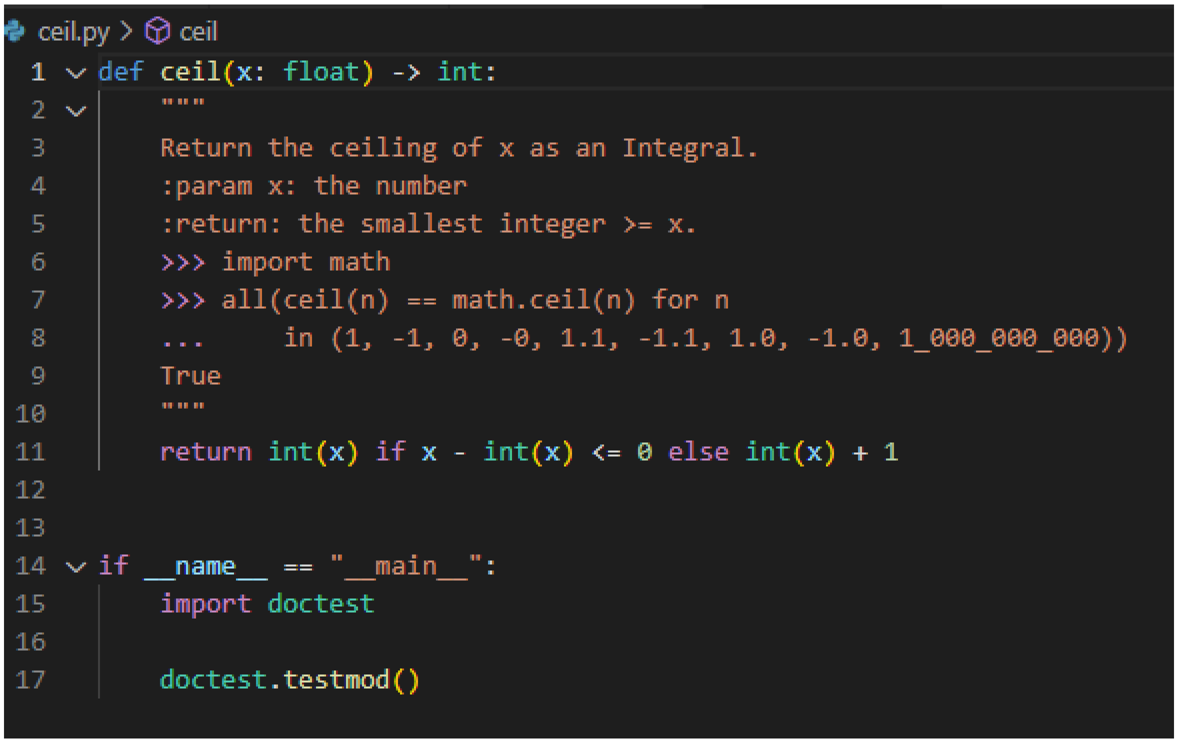
Output :



```
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> 
```

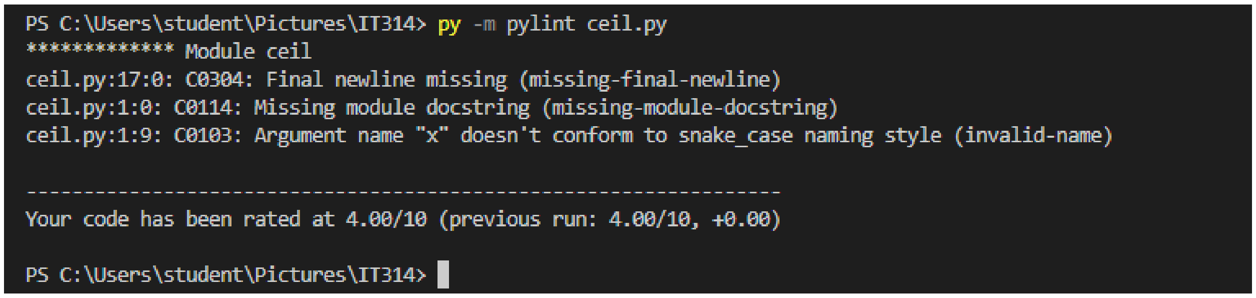
3. ceil

Code :



```
ceil.py > ceil
def ceil(x: float) -> int:
    """
    Return the ceiling of x as an Integral.
    :param x: the number
    :return: the smallest integer >= x.
    >>> import math
    >>> all(ceil(n) == math.ceil(n) for n
    ...     in (1, -1, 0, -0, 1.1, -1.1, 1.0, -1.0, 1_000_000_000))
    True
    """
    return int(x) if x - int(x) <= 0 else int(x) + 1


if __name__ == "__main__":
    import doctest

    doctest.testmod()
```

Output :

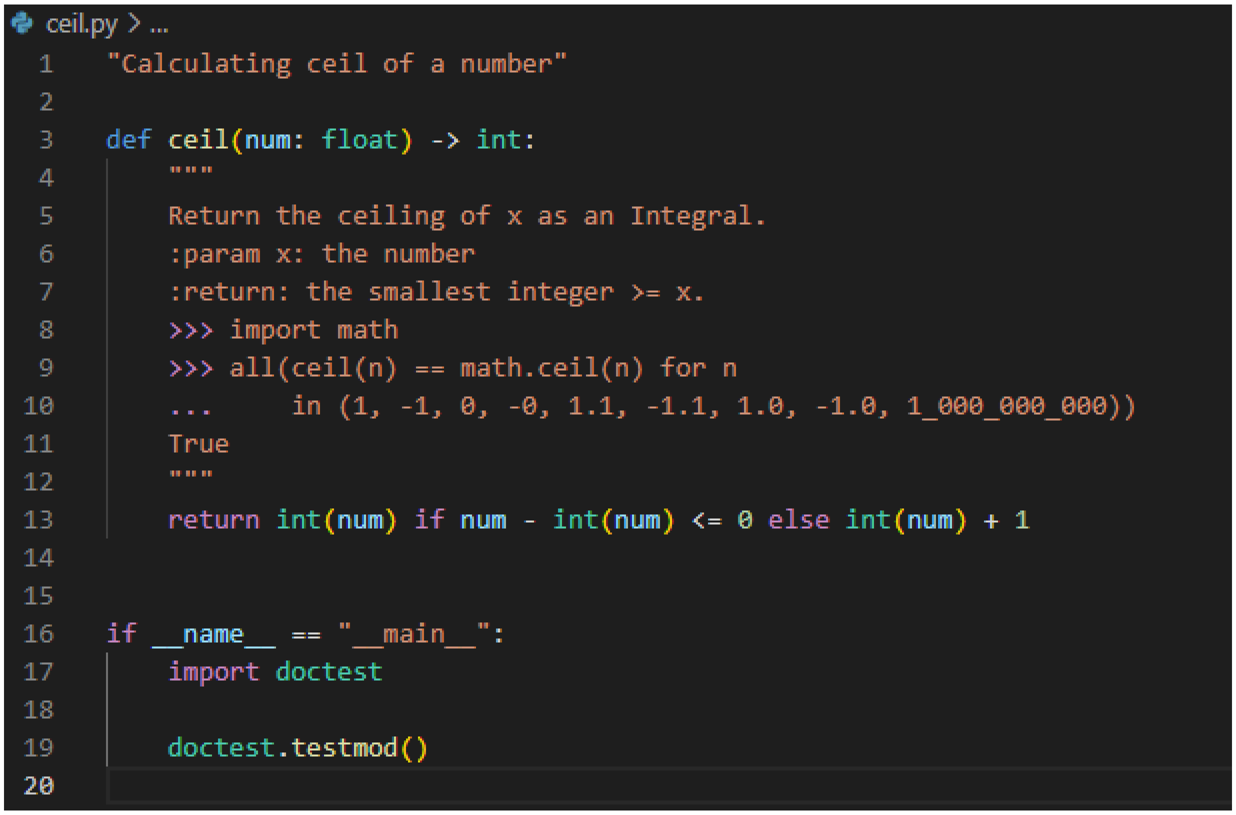


```
PS C:\Users\student\Pictures\IT314> py -m pylint ceil.py
************* Module ceil
ceil.py:17:0: C0304: Final newline missing (missing-final-newline)
ceil.py:1:0: C0114: Missing module docstring (missing-module-docstring)
ceil.py:1:9: C0103: Argument name "x" doesn't conform to snake_case naming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

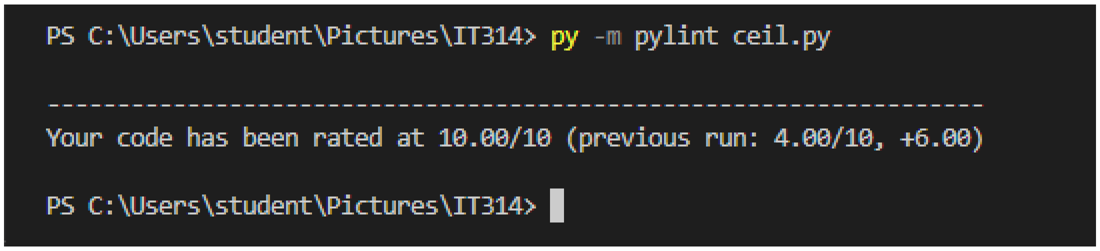
Improved Code :



ceil.py > ...

```python
"Calculating ceil of a number"

def ceil(num: float) -> int:
    """
    Return the ceiling of x as an Integral.
    :param x: the number
    :return: the smallest integer >= x.
    >>> import math
    >>> all(ceil(n) == math.ceil(n) for n
    ...     in (1, -1, 0, -0, 1.1, -1.1, 1.0, -1.0, 1_000_000_000))
    True
    """
    return int(num) if num - int(num) <= 0 else int(num) + 1


if __name__ == "__main__":
    import doctest

    doctest.testmod()

```

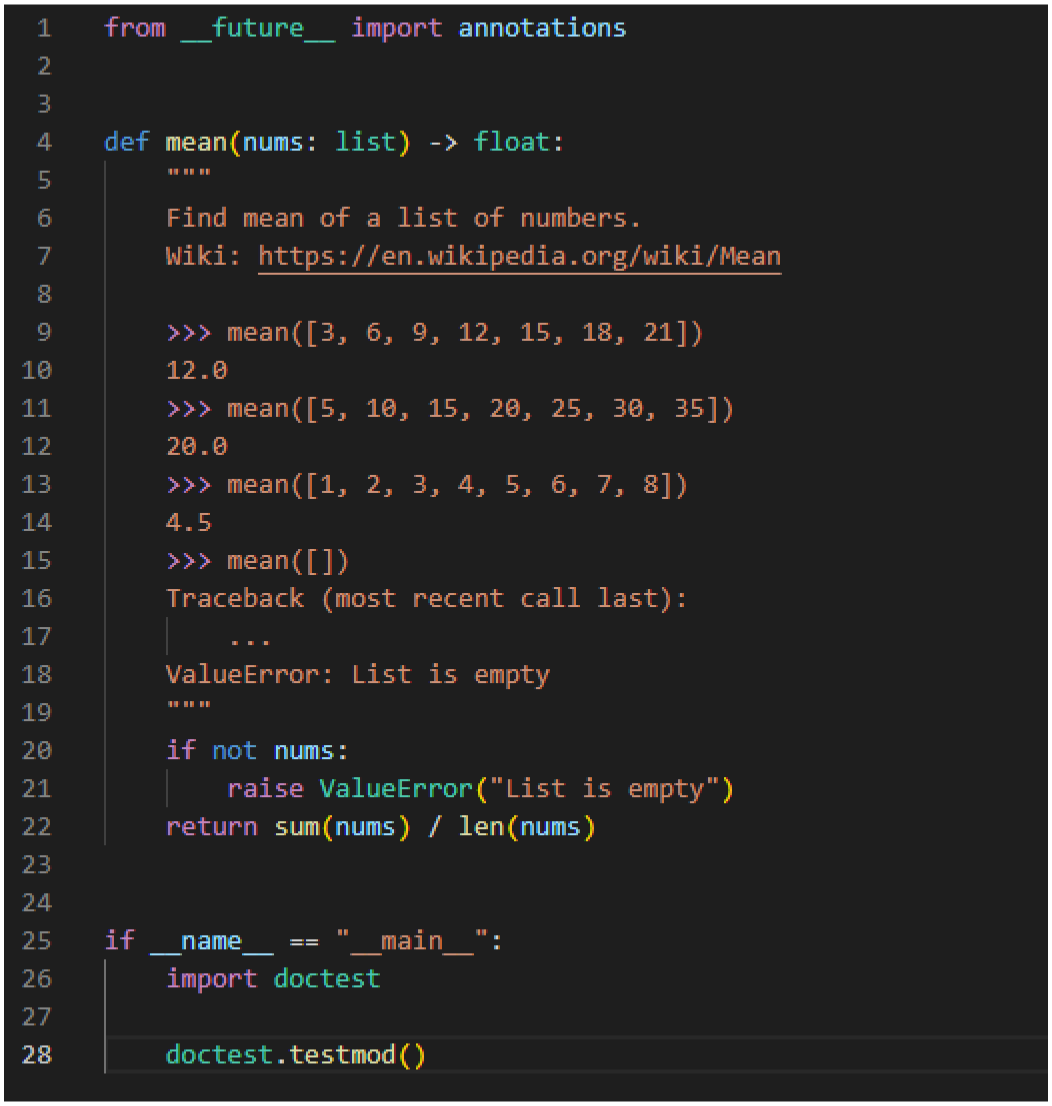
Output :



```
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>
```

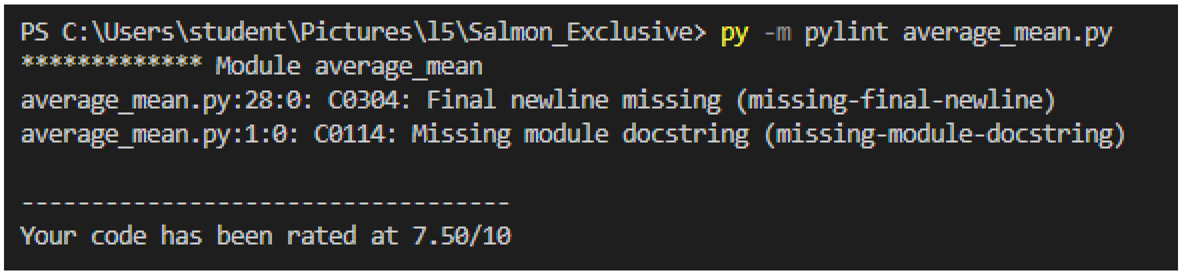
4. average_mean

Code :



```python
from __future__ import annotations


def mean(nums: list) -> float:
    """
    Find mean of a list of numbers.
    Wiki: https://en.wikipedia.org/wiki/Mean

    >>> mean([3, 6, 9, 12, 15, 18, 21])
    12.0
    >>> mean([5, 10, 15, 20, 25, 30, 35])
    20.0
    >>> mean([1, 2, 3, 4, 5, 6, 7, 8])
    4.5
    >>> mean([])
    Traceback (most recent call last):
        ...
    ValueError: List is empty
    """
    if not nums:
        raise ValueError("List is empty")
    return sum(nums) / len(nums)


if __name__ == "__main__":
    import doctest

    doctest.testmod()
```

Output :

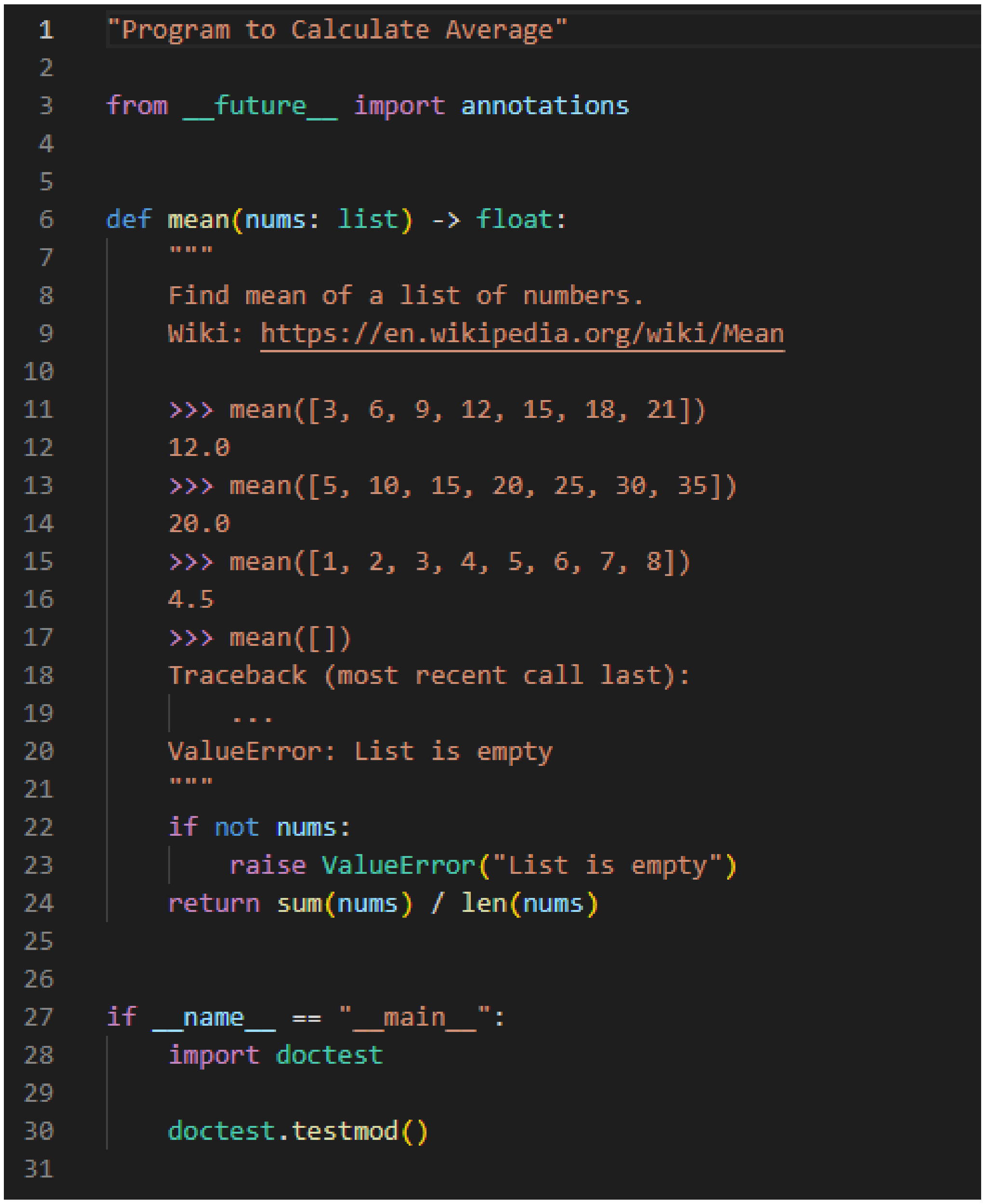


```
PS C:\Users\student\Pictures\15\Salmon_Exclusive> py -m pylint average_mean.py
************* Module average_mean
average_mean.py:28:0: C0304: Final newline missing (missing-final-newline)
average_mean.py:1:0: C0114: Missing module docstring (missing-module-docstring)

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

Error Analysis :

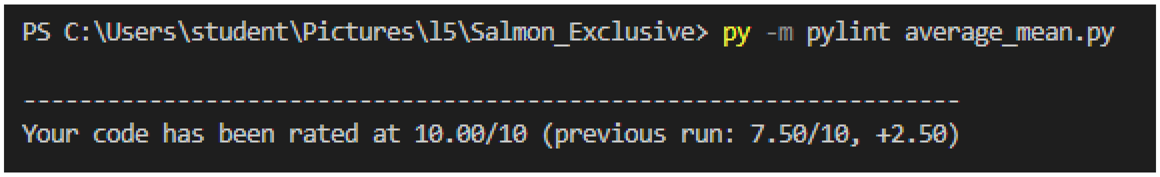
- **Missing final newline.**
    - We have to add a new line when our code is complete.
- **Missing module docstring**
    - At the start of the code we add a string(comment) which indicates the use of our programme.

Improved Code :



```python
"Program to Calculate Average"

from __future__ import annotations


def mean(nums: list) -> float:
    """
    Find mean of a list of numbers.
    Wiki: https://en.wikipedia.org/wiki/Mean

    >>> mean([3, 6, 9, 12, 15, 18, 21])
    12.0
    >>> mean([5, 10, 15, 20, 25, 30, 35])
    20.0
    >>> mean([1, 2, 3, 4, 5, 6, 7, 8])
    4.5
    >>> mean([])
    Traceback (most recent call last):
        ...
    ValueError: List is empty
    """
    if not nums:
        raise ValueError("List is empty")
    return sum(nums) / len(nums)


if __name__ == "__main__":
    import doctest

    doctest.testmod()

```

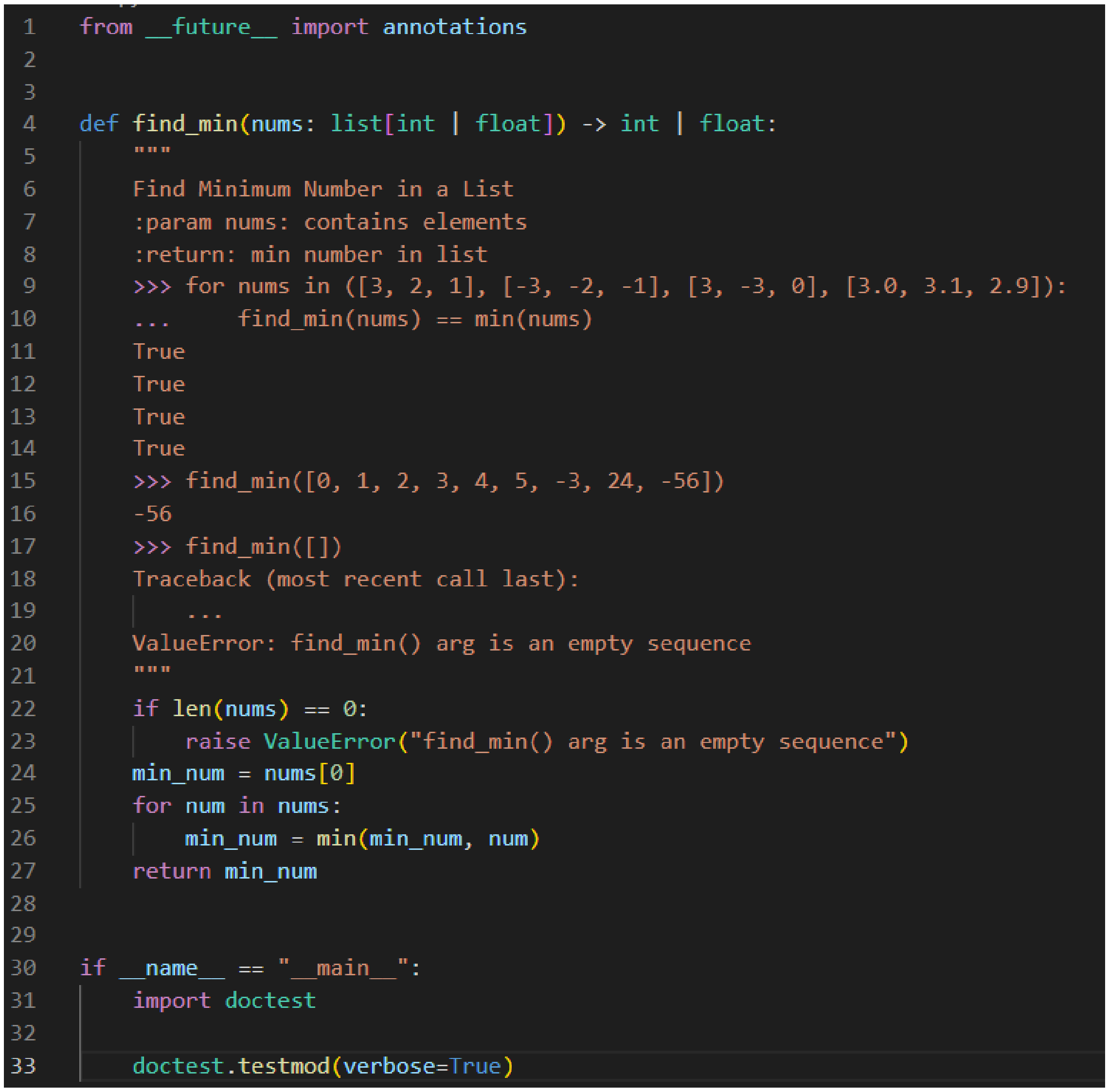
Output :



```
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)
```

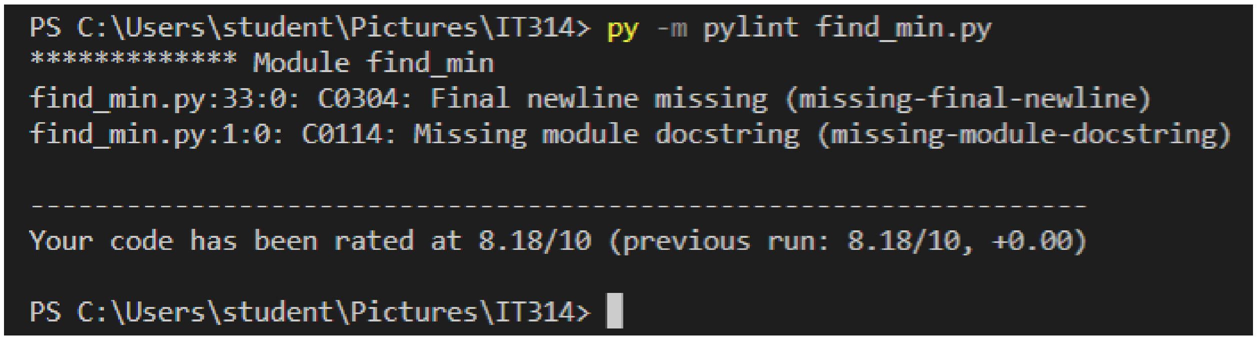
5. find_min

Code :



```python
from __future__ import annotations


def find_min(nums: list[int | float]) -> int | float:
    """
    Find Minimum Number in a List
    :param nums: contains elements
    :return: min number in list
    >>> for nums in ([3, 2, 1], [-3, -2, -1], [3, -3, 0], [3.0, 3.1, 2.9]):
    ...     find_min(nums) == min(nums)
    True
    True
    True
    True
    >>> find_min([0, 1, 2, 3, 4, 5, -3, 24, -56])
    -56
    >>> find_min([])
    Traceback (most recent call last):
        ...
    ValueError: find_min() arg is an empty sequence
    """
    if len(nums) == 0:
        raise ValueError("find_min() arg is an empty sequence")
    min_num = nums[0]
    for num in nums:
        min_num = min(min_num, num)
    return min_num


if __name__ == "__main__":
    import doctest

    doctest.testmod(verbose=True)
```

Output :

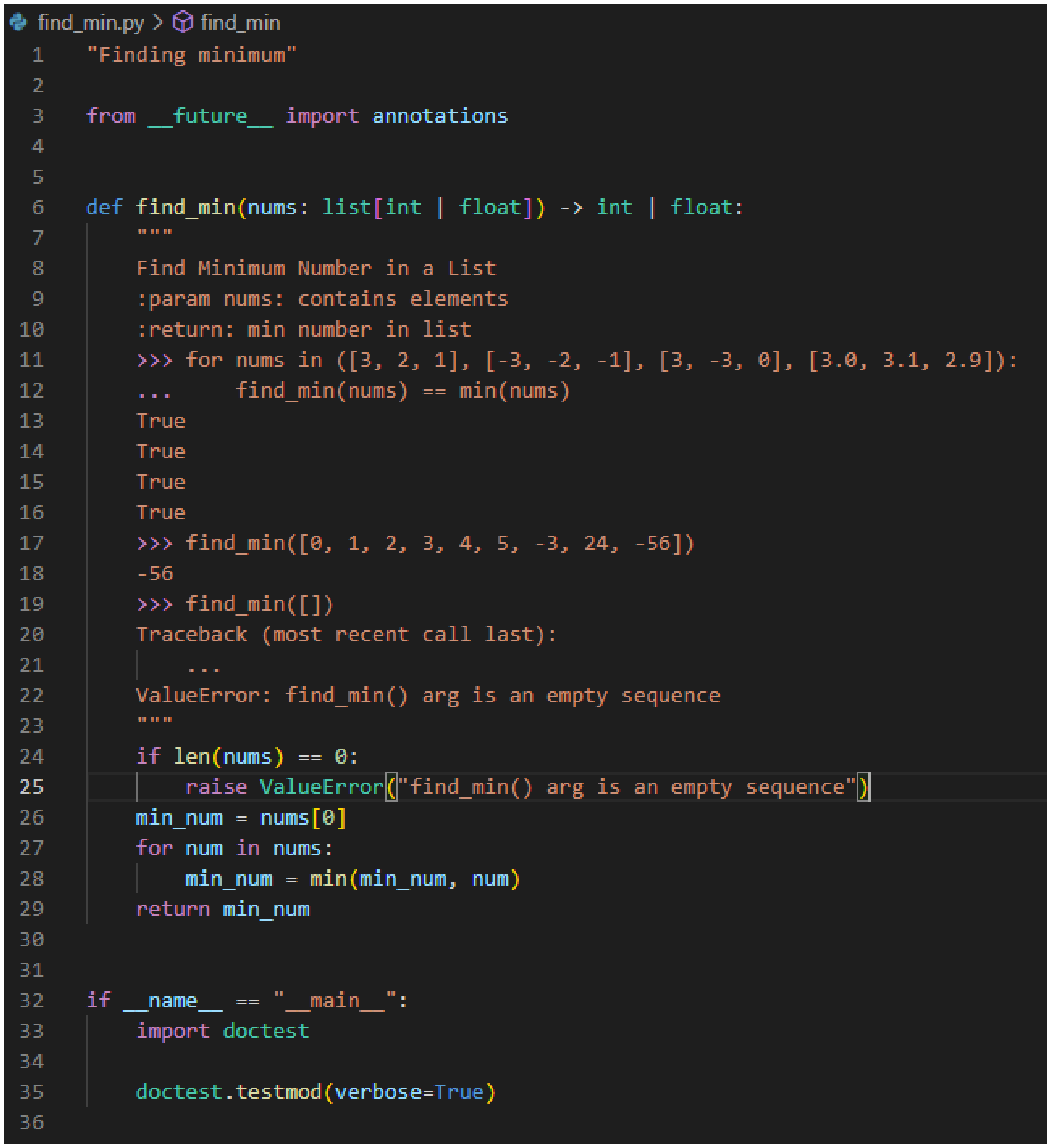


```
PS C:\Users\student\Pictures\IT314> py -m pylint find_min.py
************* Module find_min
find_min.py:33:0: C0304: Final newline missing (missing-final-newline)
find_min.py:1:0: C0114: Missing module docstring (missing-module-docstring)

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

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

Error Analysis :

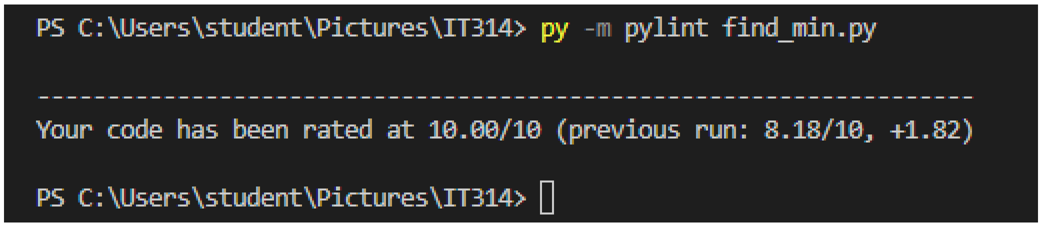
- **Missing final newline.**
    - We have to add a new line when our code is complete.
- **Missing module docstring**
    - At the start of the code we add a string(comment) which indicates the use of our programme.

Improved Code :



```python
"Finding minimum"

from __future__ import annotations


def find_min(nums: list[int | float]) -> int | float:
    """
    Find Minimum Number in a List
    :param nums: contains elements
    :return: min number in list
    >>> for nums in ([3, 2, 1], [-3, -2, -1], [3, -3, 0], [3.0, 3.1, 2.9]):
    ...     find_min(nums) == min(nums)
    True
    True
    True
    True
    >>> find_min([0, 1, 2, 3, 4, 5, -3, 24, -56])
    -56
    >>> find_min([])
    Traceback (most recent call last):
        ...
    ValueError: find_min() arg is an empty sequence
    """
    if len(nums) == 0:
        raise ValueError("find_min() arg is an empty sequence")
    min_num = nums[0]
    for num in nums:
        min_num = min(min_num, num)
    return min_num


if __name__ == "__main__":
    import doctest

    doctest.testmod(verbose=True)
```

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> 
```