Software Engineering (IT314)
Lab : 5

ID : 202001042
Name : Jay Kuvadiya

Date : 24 / 03 / 2023

- 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.

Let’s discuss some techniques to improve your score.

- ID **C0326** suggests a bad-white space error means we need to give a whitespace between a and = symbol. This rule is applicable to all declarations where an operator is used immediately after an identifier.
- ID **C0304** comes under missing-new-line suggestion which means we have to add a blank line when we complete our code.
- ID **C0114** comes under missing-module-docstring suggestion which means we need to add a docstring at the top which refers to the use of the program written below that.
- ID **C0103** comes under invalid-name suggestion which can be avoided by writing the identifiers starting with a capital letter. But, we usually believe that class names use CamelCasing i.e class names start with an upper-case letter. To avoid this suggestion we will add a regular expression to pylint that actually accepts all the variables in the lowercase letters. We will discuss this more in the further examples.

1. Repo link

- **Code**:

```
from doctest import testmod
from math import sqrt


def factors_of_a_number(num: int) -> list:
    """
    >>> factors_of_a_number(1)
    [1]
    >>> factors_of_a_number(5)
    [1, 5]
    >>> factors_of_a_number(24)
    [1, 2, 3, 4, 6, 8, 12, 24]
    >>> factors_of_a_number(-24)
    []
    """
    facs: list[int] = []
    if num < 1:
        return facs
    facs.append(1)
    if num == 1:
        return facs
    facs.append(num)
    for i in range(2, int(sqrt(num)) + 1):
        if num % i == 0:  # If i is a factor of num
            facs.append(i)
            d = num // i  # num//i is the other factor of num
            if d != i:  # If d and i are distinct
                facs.append(d)  # we have found another factor
    facs.sort()
    return facs


if __name__ == "__main__":
    testmod(name="factors_of_a_number", verbose=True)
```

- **Output:**

```
PS C:\Users\student\Downloads\202001042> py -m pylint 1.py
************* Module 1
1.py:34:0: C0304: Final newline missing (missing-final-newline)
1.py:1:0: C0114: Missing module docstring (missing-module-docstring)
1.py:1:0: C0103: Module name "1" doesn't conform to snake_case naming style (invalid-name)
1.py:26:12: C0103: Variable name "d" doesn't conform to snake_case naming style (invalid-name)

-----------------------------------
Your code has been rated at 8.00/10
```

- **Analysis**
  - All errors are ture.

- **Error types:**
  - C0304 - Missing line
  - C0114- Missing Module docstring
  - C0103- Invalid name.

- **After Improvement:**

- **Code**

```
"""Find all factors of a number"""
from doctest import testmod
from math import sqrt
def factors_of_a_number(num: int) -> list:
    """
       >>> factors_of_a_number(1)
    [1]
    >>> factors_of_a_number(5)
```

```
    [1, 5]
    >>> factors_of_a_number(24)
    [1, 2, 3, 4, 6, 8, 12, 24]

  >>> factors_of_a_number(-24)
    []
    """
    facs: list[int] = []
    if num < 1:
        return facs
    facs.append(1)
    if num == 1:
        return facs
    facs.append(num)
    for i in range(2, int(sqrt(num)) + 1):
        if num % i == 0:  # If i is a factor of num
            facs.append(i)
            temp = num // i  # num//i is the other factor of num
            if temp != i:  # If temp and i are distinct
                facs.append(temp)  # we have found another factor
    facs.sort()
    return facs
if __name__ == "__main__":
    testmod(name="factors_of_a_number", verbose=True)
```

- **Output**

```
PS C:\Users\student\Downloads\202001042> py -m pylint factor.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.00/10, +1.00)
```

2. [Repo link]

- **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\Downloads\202001042> py -m pylint 2.py
************* Module 2
2.py:28:0: C0304: Final newline missing (missing-final-newline)
2.py:1:0: C0114: Missing module docstring (missing-module-docstring)
2.py:1:0: C0103: Module name "2" doesn't conform to snake_case naming style (invalid-name)

-----------------------------------
Your code has been rated at 6.25/10
```

- **Analysis**
  - All errors are ture.

- **Error types:**
  - C0304 - Missing line
  - C0114- Missing Module docstring
  - C0103- Invalid name.

- **After Improvement**

- **Code:**

```python
"""Find mean of a list of numbers"""
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\Downloads\202001042> py -m pylint mean.py

-----------------------------------
Your code has been rated at 10.00/10
```

3. [Repo link]

- **Code**

```
"""
Illustrate how to add the integer without arithmetic operation
Author: suraj Kumar
Time Complexity: 1
https://en.wikipedia.org/wiki/Bitwise_operation
"""


def add(first: int, second: int) -> int:
    """
    Implementation of addition of integer

    Examples:
    >>> add(3, 5)
    8
    >>> add(13, 5)
    18
    >>> add(-7, 2)
    -5
    >>> add(0, -7)
    -7
    >>> add(-321, 0)
    -321
    """
    while second != 0:
        c = first & second
        first ^= second
        second = c << 1
    return first


if __name__ == "__main__":
    import doctest

    doctest.testmod()

    first = int(input("Enter the first number: ").strip())
    second = int(input("Enter the second number: ").strip())
```

```
    print(f"{add(first, second) = }")
```

- **Output**

```
PS C:\Users\student\Downloads\202001042> py -m pylint 3.py
************* Module 3
3.py:39:0: C0304: Final newline missing (missing-final-newline)
3.py:1:0: C0103: Module name "3" doesn't conform to snake_case naming style (invalid-name)
3.py:9:8: W0621: Redefining name 'first' from outer scope (line 37) (redefined-outer-name)
3.py:9:20: W0621: Redefining name 'second' from outer scope (line 38) (redefined-outer-name)
3.py:26:8: C0103: Variable name "c" doesn't conform to snake_case naming style (invalid-name)

-----------------------------------
Your code has been rated at 5.83/10
```

- **Analysis**
  - All errors are ture.

- **Error types:**
  - C0304 - Missing line
  - W0621- Redefined outer name.
  - C0103- Invalid name.

- **After Improvement**

- **Code:**

```
"""
Illustrate how to add the integer without arithmetic operation
Author: suraj Kumar
Time Complexity: 1
https://en.wikipedia.org/wiki/Bitwise_operation
```

```
"""


def add(first: int, second: int) -> int:
    """
    Implementation of addition of integer

    Examples:
    >>> add(3, 5)
    8
    >>> add(13, 5)
    18
    >>> add(-7, 2)
    -5
    >>> add(0, -7)
    -7
    >>> add(-321, 0)
    -321
    """
    while second != 0:
        temp = first & second
        first ^= second
        second = temp << 1
    return first


if __name__ == "__main__":
    import doctest

    doctest.testmod()

    num1 = int(input("Enter the first number: ").strip())
    num2 = int(input("Enter the second number: ").strip())
    print(f"{add(num1, num2) = }")
```

- **Output:**

```
PS C:\Users\student\Downloads\202001042> py -m pylint add.py

--------------------------------------
Your code has been rated at 10.00/10
```

4. [Repo link]

- **Code:**

```python
"""
Write a function that takes an angle and a radius as input and returns
the arc length of the angle.
"""
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\Downloads\202001042> py -m pylint arclength.py
************* Module arclength
arclength.py:18:0: C0304: Final newline missing (missing-final-newline)

-----------------------------------
Your code has been rated at 8.00/10
```

- **Analysis**
  - All errors are ture.

- **Error types:**
  - C0304 - Missing line

- **After Improvement**

- **Code:**

```
"""

Write a function that takes an angle and a radius as input and returns
the arc length of the angle.

"""

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\Downloads\202001042> py -m pylint arclength.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 8.00/10, +2.00)
```

5. [Repo link]

- **Code:**

```
"""
Find median of a list of numbers.
"""
from __future__ import annotations
import doctest


#function to find median
def median(nums: list) -> int | float:
    """
    Find median of a list of numbers.
    Wiki: https://en.wikipedia.org/wiki/Median
```

```
    >>> median([0])
    0
    >>> median([4, 1, 3, 2])
    2.5
    >>> median([2, 70, 6, 50, 20, 8, 4])
    8

    Args:
        nums: List of nums

    Returns:
        Median.
    """
    sorted_list = sorted(nums)
    length = len(sorted_list)
    mid_index = length >> 1
    return (
        (sorted_list[mid_index] + sorted_list[mid_index - 1]) / 2
        if length % 2 == 0
        else sorted_list[mid_index]
    )


def main():
    """Main function"""
    doctest.testmod()


if __name__ == "__main__":
    main()
```

- **Output**

```
PS C:\Users\student\Downloads\202001042> py -m pylint .\average_median.py

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.09/10, +0.91)
```