Pylint Error Code Documentation

— Jian Zhe, Jason

C0103: invalid-name

Stegman: names/format

Corresponding Hyperstyle Code(s): E741, N816

Description:

Used when the name doesn't conform to naming rules associated to its type (constant, variable, class...). The source of the naming rules is PEP8.

Examples:

— Does not conform to snake_case: n / d / s / e / li / v (variables inside of a function)

— Does not conform to upper_case: size / table / collusion / ans (variables outside of a function)

Disable: Yes

C0115: missing-class-docstring

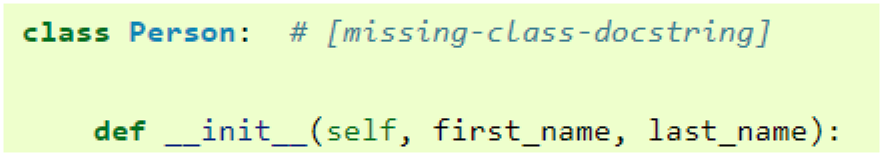
Stegman: documentation (comments)

Description:

Used when a class has no docstring right after definition. Even an empty class must have a docstring to help readers understand the class.

Examples:

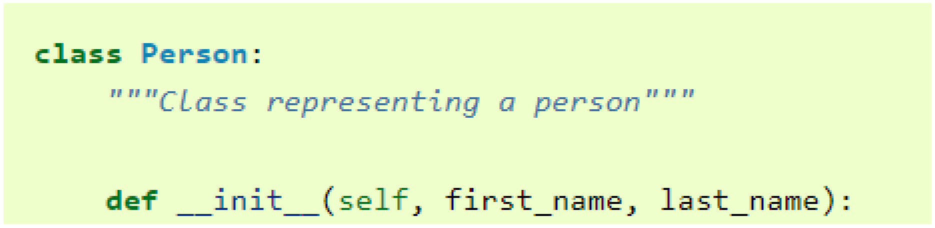
- Class without docstring



```
class Person:  # [missing-class-docstring]

    def __init__(self, first_name, last_name):
```

- Class with docstring



```
class Person:
    """Class representing a person"""

    def __init__(self, first_name, last_name):
```

Disable: Yes

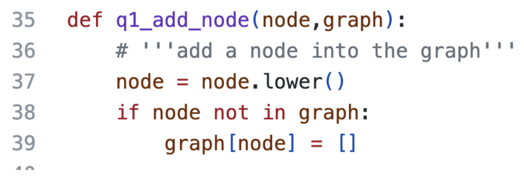
C0116: missing-function-docstring

Stegman: documentation (comments)

Description:

Used when a function or method has no docstring after definition.

Examples:



```
def q1_add_node(node,graph):
    # '''add a node into the graph'''
    node = node.lower()
    if node not in graph:
        graph[node] = []
```

Disable: Yes

C0121: singleton-comparison

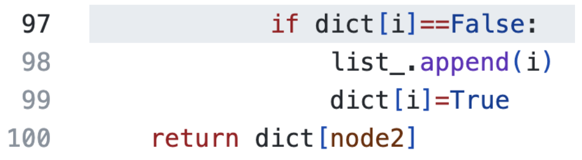
Corresponding Hyperstyle Code(s): E721

Stegman: expressions

Description:

Used when an expression is compared to singleton values like True, False or None. **(Against Refactoring Rule # 1)**

Examples:



```
            if dict[i]==False:
                list_.append(i)
                dict[i]=True
    return dict[node2]
```

Disable: No

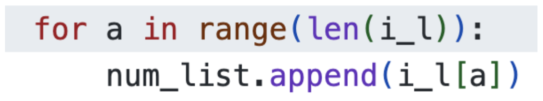
C0200: consider-using-enumerate

Stegman: presentation (formatting)

Description:

Emitted when code that iterates with range and len is encountered. Such code can be simplified by using the enumerate builtin.

Example:



```
for a in range(len(i_l)):
    num_list.append(i_l[a])
```

Disable: No

C0301: line-too-long

Stegman: presentation (formatting)

Corresponding Hyperstyle Code(s): C001, C002, C901, E501

Description:

Used when a line is longer than a given number of characters.

Disable: Yes

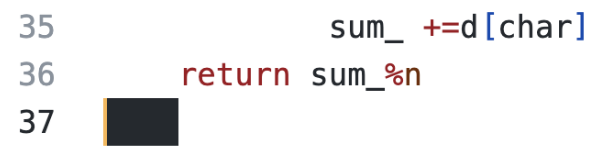
C0303: trailing-whitespace

Stegman: formatting

Corresponding Hyperstyle Code(s): E201, E202, E203, E211, E221, E222, E241, E251, E271

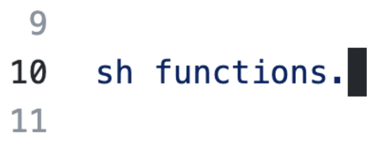
Description: Used when there is whitespace between the end of a line and the newline.

Examples:



```
            sum_ +=d[char]
    return sum_%n

```



```

sh functions.

```

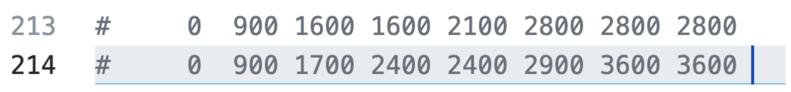
Disable: Yes

C0304: missing-final-newline

Stegman: formatting

Description: Used when the last line in a file is missing a newline.

Examples:



```
#     0   900 1600 1600 2100 2800 2800 2800
#     0   900 1700 2400 2400 2900 3600 3600
```

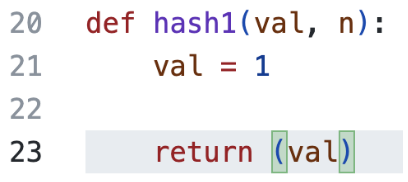
Disable: Yes

C0325: superfluous-parens (TBD)

Stegman: expression

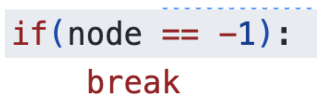
Description: Used when a single item in parentheses follows an if, for, or other keywords.

Examples:



```python
def hash1(val, n):
    val = 1

    return (val)
```



```python
if(node == -1):
    break
```

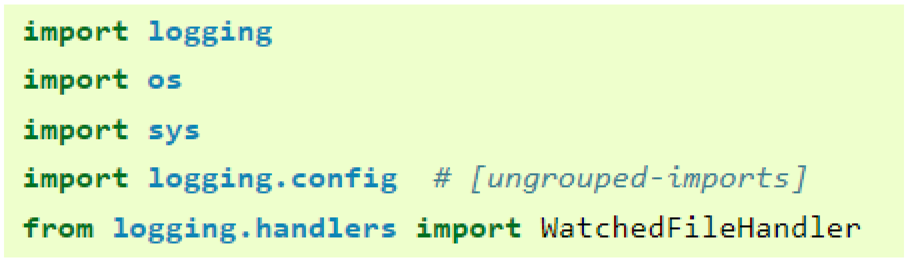
Disable: Yes / No

C0412: ungrouped-imports

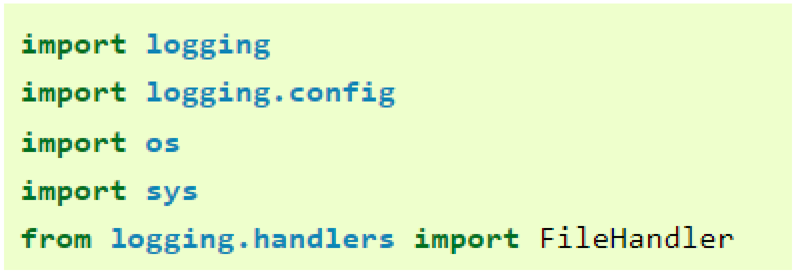
Stegman: presentation (layout)

Description: Used when imports are not grouped by packages.

Examples:



```python
import logging
import os
import sys
import logging.config  # [ungrouped-imports]
from logging.handlers import WatchedFileHandler
```



```python
import logging
import logging.config
import os
import sys
from logging.handlers import FileHandler
```

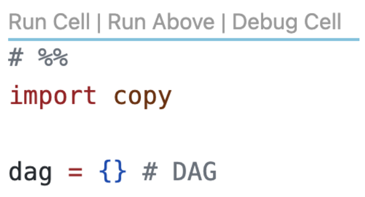
Disable: Yes

C0413: wrong-import-position

Stegman: presentation (layout)

Description: Used when code and imports are mixed.

Examples:



```python
# %%
import copy

dag = {} # DAG
```

Disable: Yes

E0001: syntax-error

Stegman: None (Error)

Description: Used when a syntax error is raised for a module.

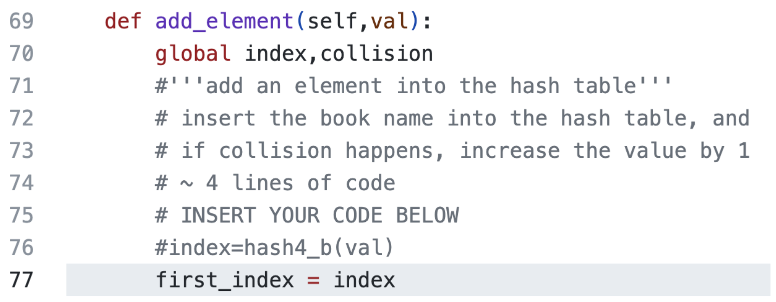
Disable: No

E0601: used-before-assignment

Stegman: None (Error)

Description: Emitted when a local variable is accessed before its assignment took place

Examples:



```python
    def add_element(self,val):
        global index,collision
        #'''add an element into the hash table'''
        # insert the book name into the hash table, and
        # if collision happens, increase the value by 1
        # ~ 4 lines of code
        # INSERT YOUR CODE BELOW
        #index=hash4_b(val)
        first_index = index
```

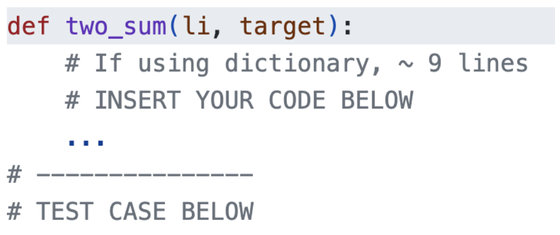
Disable: No

E1111: assignment-from-no-return

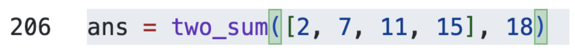
Stegman: None (Error)

Description: Assigning result of a function call, where the function has no return

Examples:



```python
def two_sum(li, target):
    # If using dictionary, ~ 9 lines
    # INSERT YOUR CODE BELOW
    ...
# ----------------
# TEST CASE BELOW
```



```
ans = two_sum([2, 7, 11, 15], 18)
```

Disable: No

E1121: too-many-function-args (TBD)

Stegman: Presentation (Formatting)

Description: Used when a function call passes too many positional arguments.

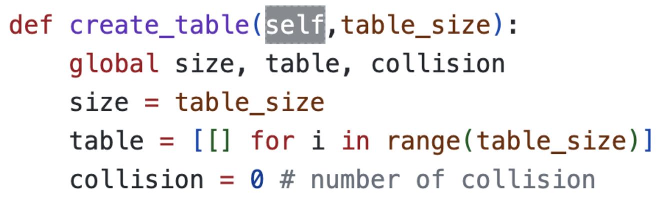
Disable: Yes

R0201: no-self-use (TBD)

Stegman: None (Best practice)

Description: Used when a method in a class doesn't interact with its class variables using the keyword 'self'

Examples:



```
def create_table(self,table_size):
    global size, table, collision
    size = table_size
    table = [[] for i in range(table_size)]
    collision = 0 # number of collision
```

Disable: Yes

R0801: Similar lines in 11 files

Stegman: None

Description: Used when a file contains similar functionalities compared to other files in the same directory

Disable: Yes

R0914: too-many-locals

Stegman: Presentation (Formatting)

Description: Used when a function or method has too many local variables.

Disable: Yes

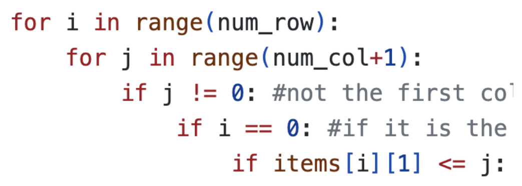
R1702: too-many-nested-blocks

Stegman: Flow

Corresponding Hyperstyle Code(s): WPS220

Description: Used when a function or a method has too many nested blocks.

Examples:



```
for i in range(num_row):
    for j in range(num_col+1):
        if j != 0: #not the first co
            if i == 0: #if it is the
                if items[i][1] <= j:
```

Disable: No

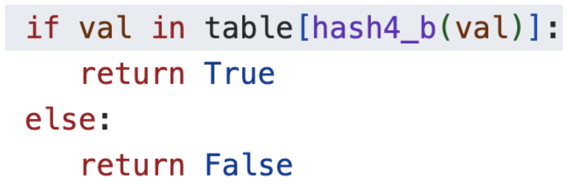
R1703: simplifiable-if-statement

Stegman: Expression

Corresponding Hyperstyle Code(s): WPS503, WPS531

Description: Used when an if statement can be replaced with 'bool(test)'.

Examples:



```
if val in table[hash4_b(val)]:
    return True
else:
    return False
```

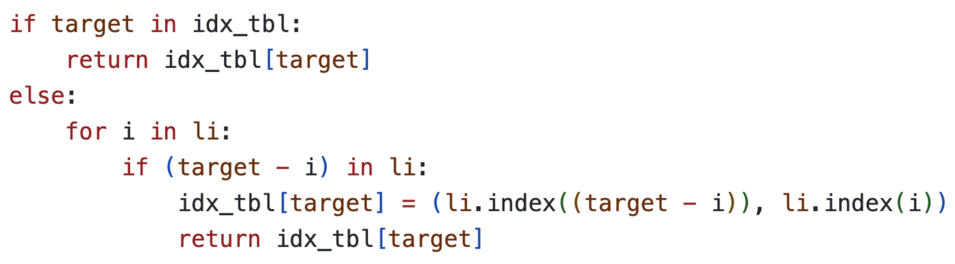
Disable: No

R1705: no-else-return

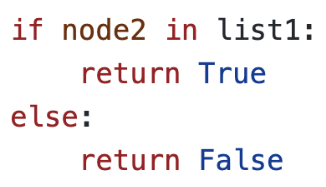
Stegman: Expression

Description: Used in order to highlight an unnecessary block of code following an if containing a return statement.

Examples:



```
if target in idx_tbl:
    return idx_tbl[target]
else:
    for i in li:
        if (target - i) in li:
            idx_tbl[target] = (li.index((target - i)), li.index(i))
            return idx_tbl[target]
```



```python
if node2 in list1:
    return True
else:
    return False
```

Disable: No

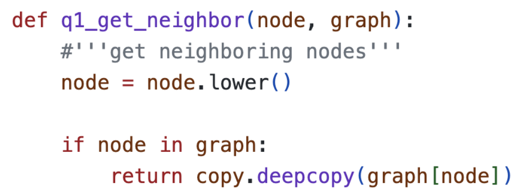
R1710: inconsistent-return-statements

Stegman: Flow

Corresponding Hyperstyle Code(s): R503

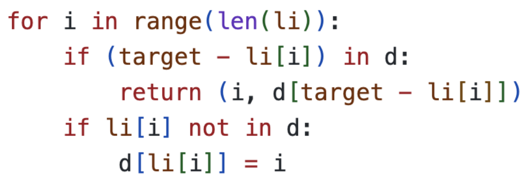
Description: if any return statement in a method/function returns an expression, any other return statements where no value is returned should explicitly state this as return **None**.

Examples:



```python
def q1_get_neighbor(node, graph):
    #'''get neighboring nodes'''
    node = node.lower()

    if node in graph:
        return copy.deepcopy(graph[node])
```



```python
for i in range(len(li)):
    if (target - li[i]) in d:
        return (i, d[target - li[i]])
    if li[i] not in d:
        d[li[i]] = i
```

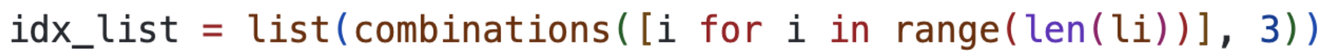
Disable: No

R1721: unnecessary-comprehension (TBD)

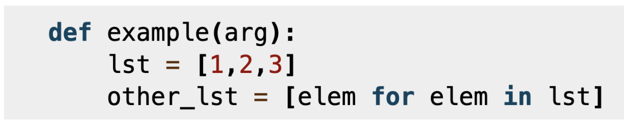
Stegman: Presentation (Formatting)

Description: Instead of using an identity comprehension, consider using the list, dict or set constructor.

Examples:

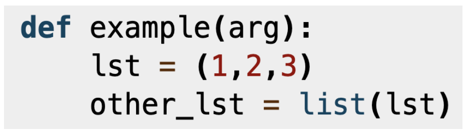


```python
idx_list = list(combinations([i for i in range(len(li))], 3))
```



```python
def example(arg):
    lst = [1,2,3]
    other_lst = [elem for elem in lst]
```

(unnecessary-comprehension)



```python
def example(arg):
    lst = (1,2,3)
    other_lst = list(lst)
```

(No issue found)

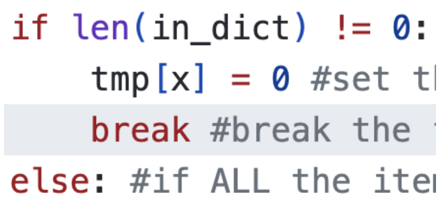
Disable: Yes

R1723: no-else-break

Stegman: Expression

Description: When an else statement is found after a chain of ifs, all containing break statements

Examples:



```
if len(in_dict) != 0:
    tmp[x] = 0 #set t
    break #break the
else: #if ALL the ite
```

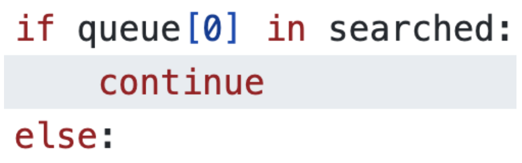
Disable: No

R1724: no-else-continue

Stegman: Expression

Description: When an else statement is found after a chain of ifs, all containing continue statements

Examples:



```
if queue[0] in searched:
    continue
else:
```

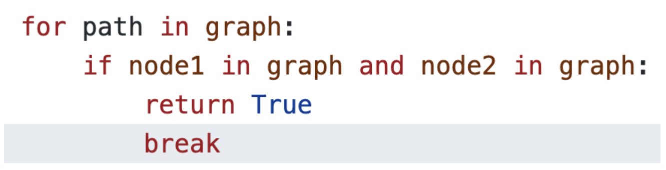
Disable: No

W0101: unreachable

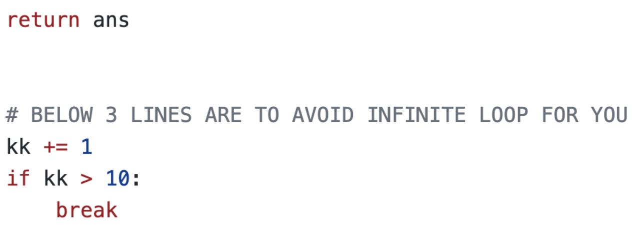
Stegman: Expression

Description: Used when there is some code behind a "return" or "raise" statement, which will never be accessed.

Examples:



```
for path in graph:
    if node1 in graph and node2 in graph:
        return True
        break
```



```
return ans

# BELOW 3 LINES ARE TO AVOID INFINITE LOOP FOR YOU
kk += 1
if kk > 10:
    break
```

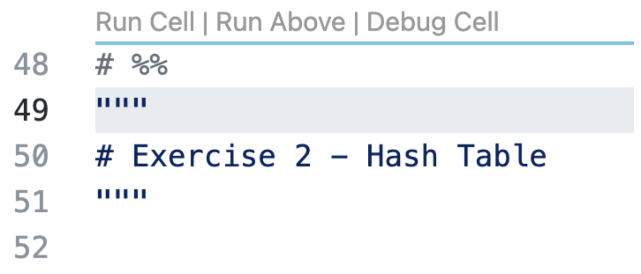
Disable: No

W0105: pointless-string-statement

Stegman: Comments

Description: Used when a string is used as a statement instead of a docstring

Examples:



Run Cell | Run Above | Debug Cell

```
# %%
"""
# Exercise 2 - Hash Table
"""

```

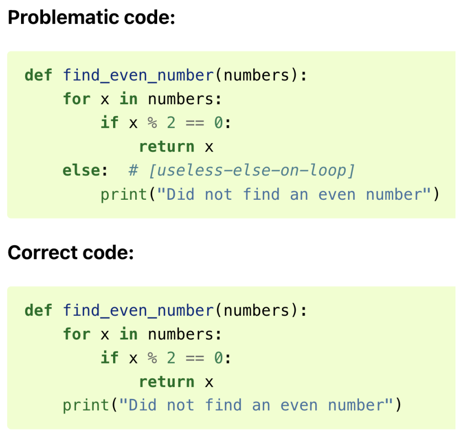
Disable: Yes

W0120: useless-else-on-loop (TBD)

Stegman: Expression

Description: Loops should only have an else clause if they can exit early with a break statement, otherwise the statements under else should be on the same scope as the loop itself.

Examples:

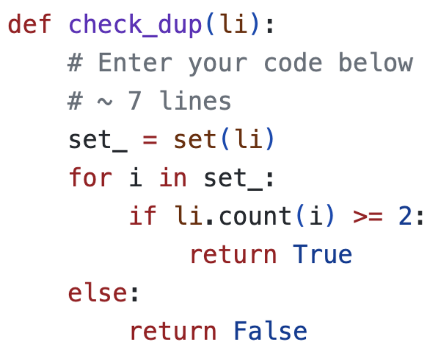


Problematic code:

```
def find_even_number(numbers):
    for x in numbers:
        if x % 2 == 0:
            return x
    else:  # [useless-else-on-loop]
        print("Did not find an even number")
```

Correct code:

```
def find_even_number(numbers):
    for x in numbers:
        if x % 2 == 0:
            return x
    print("Did not find an even number")
```



```
def check_dup(li):
    # Enter your code below
    # ~ 7 lines
    set_ = set(li)
    for i in set_:
        if li.count(i) >= 2:
            return True
    else:
        return False
```

Disable:

W0301: unnecessary-semicolon

Stegman: Presentation (Layout)

Description: Used when a statement is ended by a semi-colon (";")

Disable: No

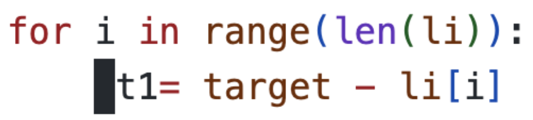
W0311: bad-indentation (TBD)

Stegman: Presentation (Layout)

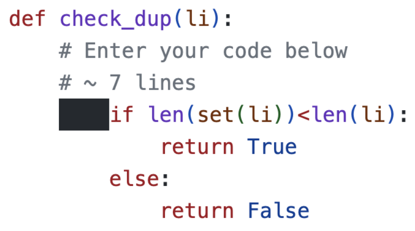
Corresponding Hyperstyle Code(s): E111, E112, E113, E114, E115, E116, E117, E999

Description: Used when an unexpected number of indentation's tabulations or spaces has been found.

Examples:



```
for i in range(len(li)):
     t1= target - li[i]
```



```
def check_dup(li):
    # Enter your code below
    # ~ 7 lines
        if len(set(li))<len(li):
            return True
        else:
            return False
```

Disable: Not sure about this. It seems like the codes can still run?

W0404: reimported

Stegman: Presentation (Layout)

Corresponding Hyperstyle Code(s): WPS458

Description: Used when a module is imported more than once.

Disable: No

W0601: global-variable-undefined

Stegman: None (Best Practice)

Description: Used when a variable is defined through the "global" statement but the variable is not defined in the module scope.

Examples:

```
class HashTable:
   size = 0
   table = []
   collision = 0

   def create_table(self,table_size):
       global size, table, collision
```

Disable: No

W0603: global-statement

Stegman: None (Best Practice)

Description: Used to discourage the usage of the "global" statement to update a global variable

Examples:

```
size = None
table = None
collision = None

def create_table(table_size):
    global table, collision
    size = table_size
    table = [[] for i in range(table_size)]# allocate an empty 2D list
    collision = 0 # number of collision
```

Disable: No

W0611: unused-import

Stegman: None (Best Practice)

Corresponding Hyperstyle Code(s): F401

Description: Used when an imported module or variable is not used.

Disable: No

W0612: unused-variable

Stegman: None (Best Practice)

Corresponding Hyperstyle Code(s): F841, B007

Description: Used when a variable is defined but not used.

Disable: No

W0613: unused-argument

Stegman: None (Best Practice)

Description: Used when a function or method argument is not used.

Disable: No

W0621: redefined-outer-name

Stegman: None (Best Practice)

Description: Used when a variable's name hides a name defined in an outer scope or except handler.

Examples:

```
count = 10


def count_it(count):  # [redefined-outer-name]
    for i in range(count):
        print(i)
```

Disable: Yes

W0622: redefined-builtin

Stegman: Name

Description: Used when a variable or function override a built-in.

Examples:

```
def hash2(val, n):
    hash = len(val)
    return hash
```

Disable: No

W0702: bare-except

Stegman: None (Best Practice)

Description: Used when an ‘except’ clause doesn't specify exceptions type to catch.

Examples:

```
try:
    s = s + d[e]
except:
    s = s + 0
```

Disable: No

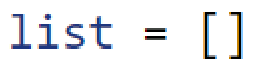
Corresponding Hyperstyle Code(s): E722

Hyperstyle Error Code Documentation

A001

Description: Used when a variable name is the same as a python builtin

Examples:



```
list = []
```

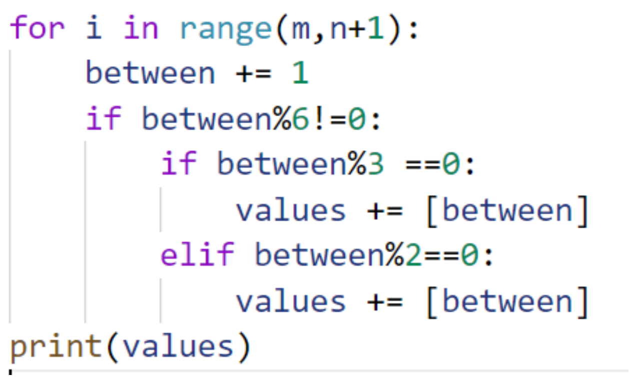
Disable: No

Corresponding Pylint Error Code: W0622

B007

Description: Used when a loop control variable is not actually used in the loop

Examples:



```
for i in range(m,n+1):
    between += 1
    if between%6!=0:
        if between%3 ==0:
            values += [between]
        elif between%2==0:
            values += [between]
print(values)
```

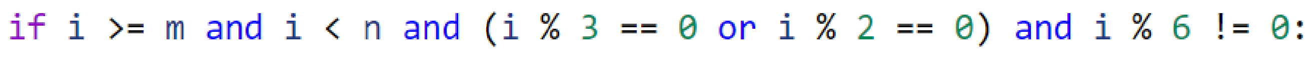
Disable: No

Corresponding Pylint Error Code: W0612

C001

Description: Used when a boolean expression is too long

Examples:



```
if i >= m and i < n and (i % 3 == 0 or i % 2 == 0) and i % 6 != 0:
```

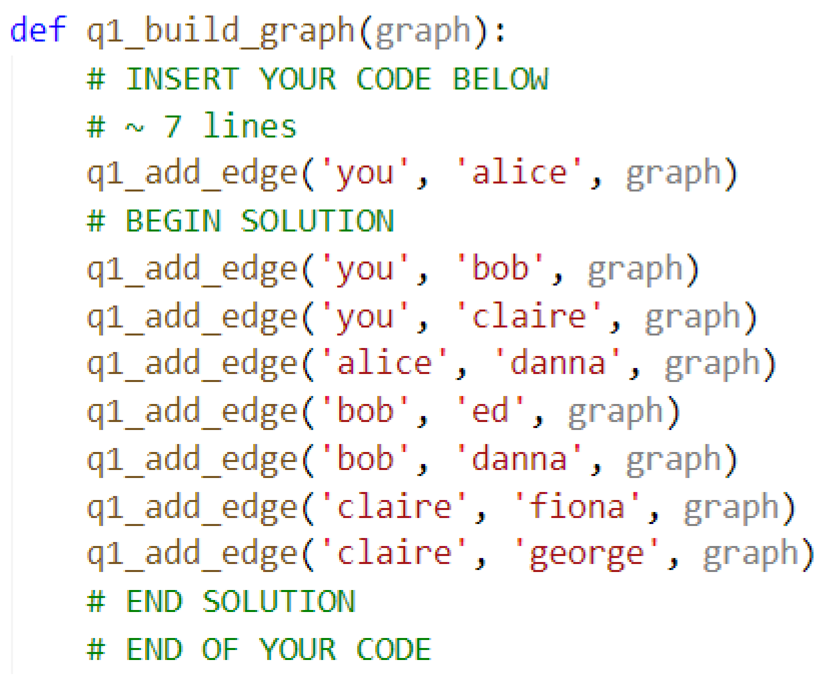
Disable: Yes

Corresponding Pylint Error Code: C0301

C002

Description: Used when a function has too many lines

Examples:



```python
def q1_build_graph(graph):
    # INSERT YOUR CODE BELOW
    # ~ 7 lines
    q1_add_edge('you', 'alice', graph)
    # BEGIN SOLUTION
    q1_add_edge('you', 'bob', graph)
    q1_add_edge('you', 'claire', graph)
    q1_add_edge('alice', 'danna', graph)
    q1_add_edge('bob', 'ed', graph)
    q1_add_edge('bob', 'danna', graph)
    q1_add_edge('claire', 'fiona', graph)
    q1_add_edge('claire', 'george', graph)
    # END SOLUTION
    # END OF YOUR CODE
```

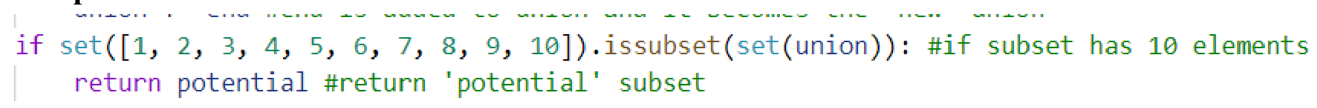
Disable: No

Corresponding Pylint Error Code: C0301

C405

Description: Unnecessary list literal - rewrite as a set literal.

Examples:



```
if set([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).issubset(set(union)): #if subset has 10 elements
    return potential #return 'potential' subset
```

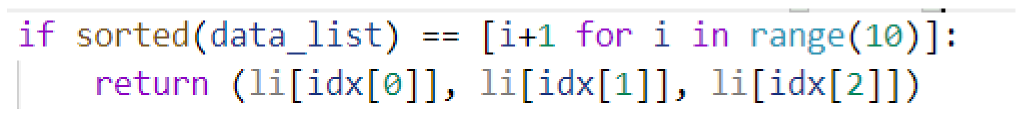
Disable: Yes

Corresponding Pylint Error Code: None

C414

Description: Unnecessary list call within sorted().

Examples:



```
if sorted(data_list) == [i+1 for i in range(10)]:
    return (li[idx[0]], li[idx[1]], li[idx[2]])
```

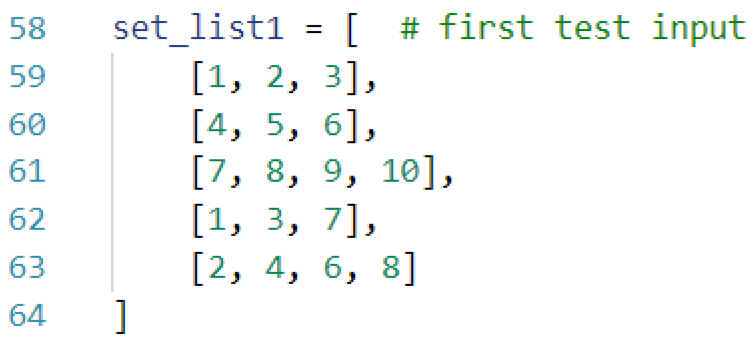
Disable: Yes

Corresponding Pylint Error Code: None

C812 (TBD)

Description: missing trailing comma

Examples:



```
set_list1 = [  # first test input
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9, 10],
    [1, 3, 7],
    [2, 4, 6, 8]
]
```

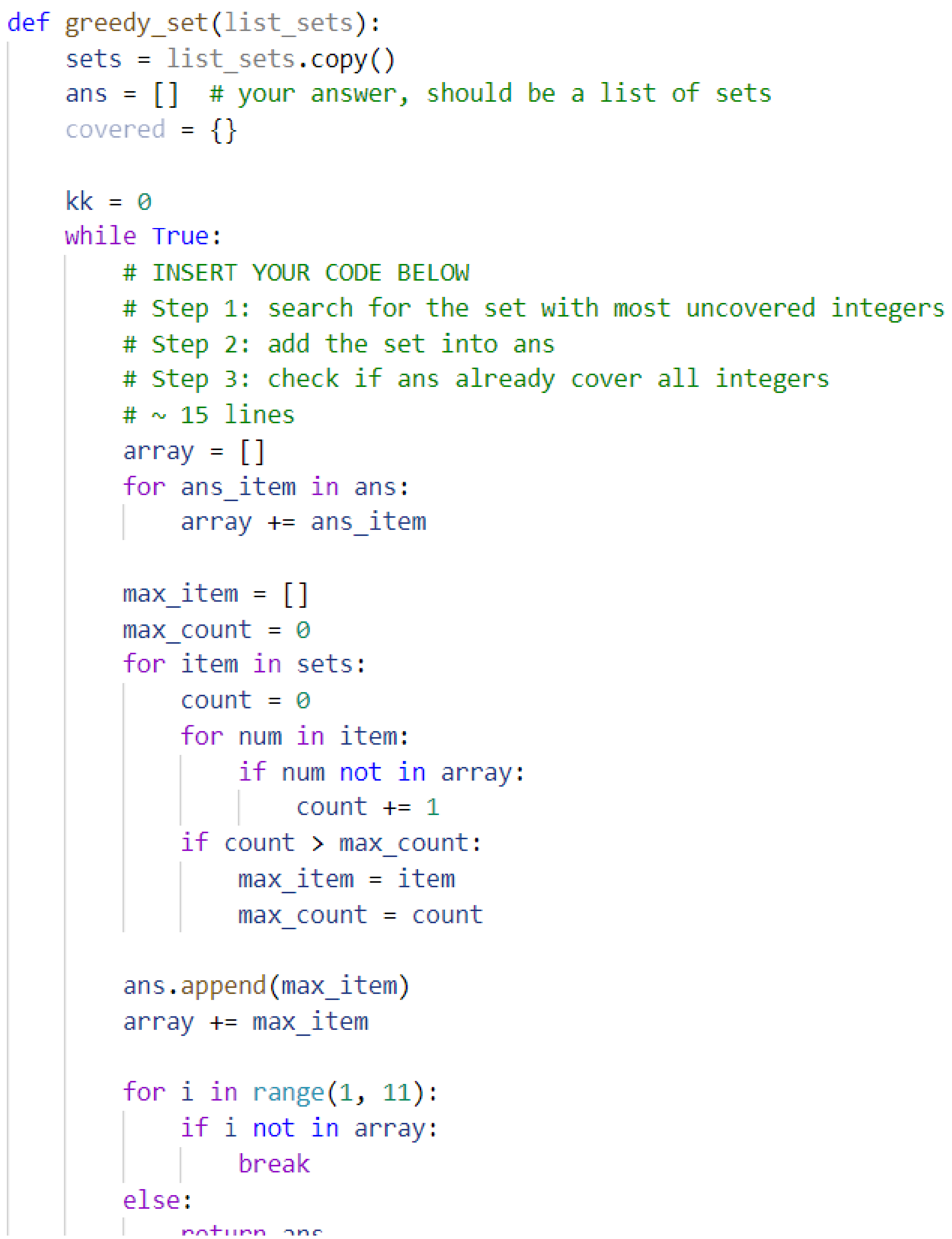
Disable: Yes

Corresponding Pylint Error Code: None

C901

Description: Used when cyclomatic complexity is too high. Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program

Examples:



```
def greedy_set(list_sets):
    sets = list_sets.copy()
    ans = []  # your answer, should be a list of sets
    covered = {}

    kk = 0
    while True:
        # INSERT YOUR CODE BELOW
        # Step 1: search for the set with most uncovered integers
        # Step 2: add the set into ans
        # Step 3: check if ans already cover all integers
        # ~ 15 lines
        array = []
        for ans_item in ans:
            array += ans_item

        max_item = []
        max_count = 0
        for item in sets:
            count = 0
            for num in item:
                if num not in array:
                    count += 1
            if count > max_count:
                max_item = item
                max_count = count

        ans.append(max_item)
        array += max_item

        for i in range(1, 11):
            if i not in array:
                break
        else:
            return ans
```

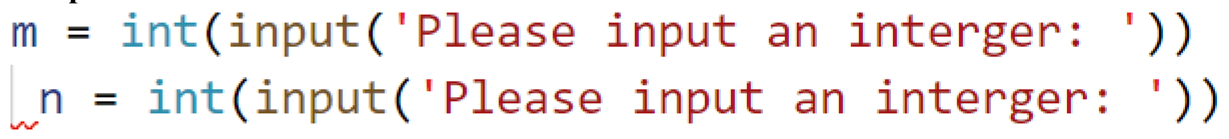
Disable: No

Corresponding Pylint Error Code: C0301

E111

Description: Used when indentation is not a multiple of a certain number

Examples:



```
m = int(input('Please input an interger: '))
 n = int(input('Please input an interger: '))
```

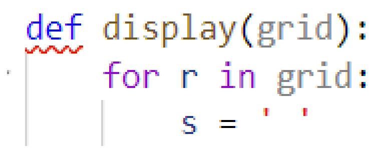
Disable: No

Corresponding Pylint Error Code: W0311

E112

Description: Used when an indented block is expected but not found

Examples:



```
def display(grid):
    for r in grid:
        s = ' '
```

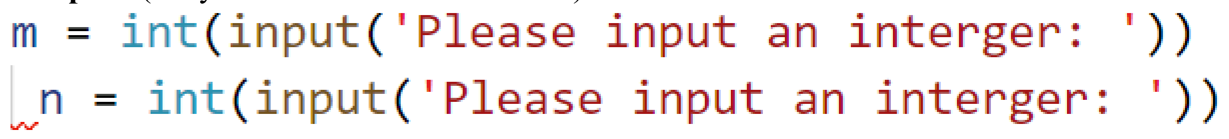
Disable: No

Corresponding Pylint Error Code: W0311

E113

Description: Used when there is unexpected indentation

Examples: (Only student with this error code)



```
m = int(input('Please input an interger: '))
 n = int(input('Please input an interger: '))
```

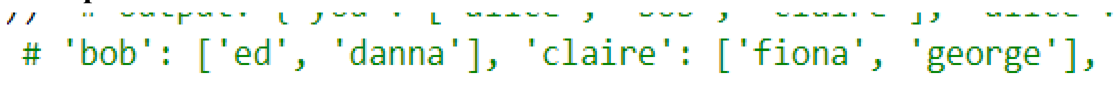
Disable: No

Corresponding Pylint Error Code: W0311

E114

Description: indentation is not a multiple of   (comment)

Examples:



```
[illegible]
# 'bob': ['ed', 'danna'], 'claire': ['fiona', 'george'],
```

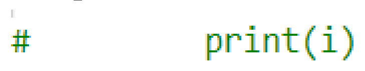
Disable: Yes

Corresponding Pylint Error Code: W0311

E115

Description: expected an indented block (comment)

Examples:



```
#         print(i)
```

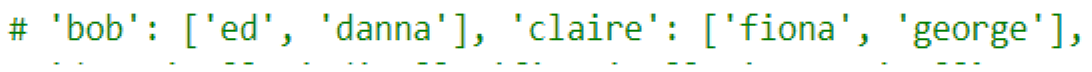
Disable: Yes

Corresponding Pylint Error Code: W0311

E116

Description: Used when there is an unexpected indentation (comment)

Examples:



```
 # 'bob': ['ed', 'danna'], 'claire': ['fiona', 'george'],
```

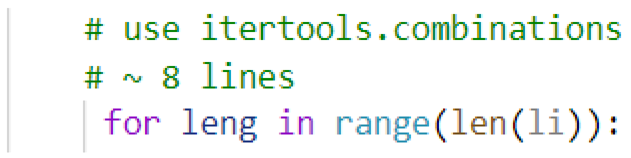
Disable: Yes

Corresponding Pylint Error Code: W0311

E117

Description: Used when a line is over-indented

Examples:



```
    # use itertools.combinations
    # ~ 8 lines
     for leng in range(len(li)):
```

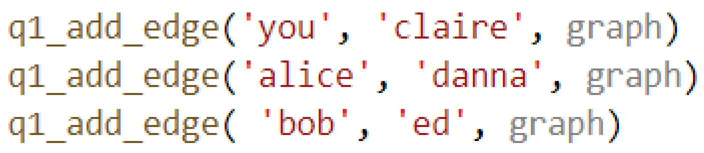
Disable: No

Corresponding Pylint Error Code: W0311

E201

Description: Used when there's a whitespace after '('

Examples:



```
q1_add_edge('you', 'claire', graph)
q1_add_edge('alice', 'danna', graph)
q1_add_edge( 'bob', 'ed', graph)
```

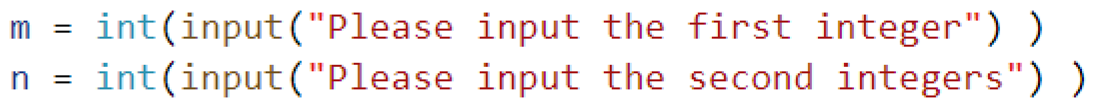
Disable: Yes

Corresponding Pylint Error Code: C0303

E202

Description: Used when there’s a whitespace before the character ‘)’

Examples:



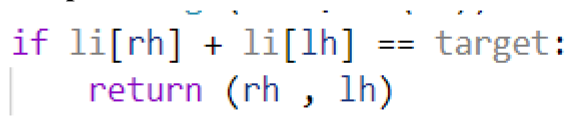
```
m = int(input("Please input the first integer") )
n = int(input("Please input the second integers") )
```

Disable: Yes
Corresponding Pylint Error Code: C0303

E203

Description: Used when there’s a whitespace before the character ‘,’ (comma)

Examples:



```
if li[rh] + li[lh] == target:
    return (rh , lh)
```

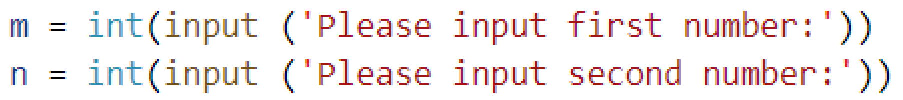
Disable: Yes

Corresponding Pylint Error Code: C0303

E211

Description: Used when there’s a whitespace before the character ‘(’

Examples:



```
m = int(input ('Please input first number:'))
n = int(input ('Please input second number:'))
```

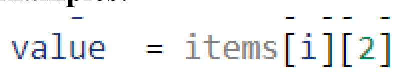
Disable: Yes

Corresponding Pylint Error Code: C0303

E221

Description: Used when there are multiple spaces before operator

Examples:



```
value   = items[i][2]
```

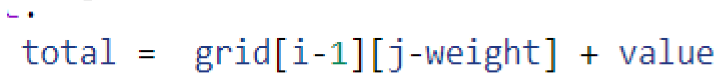
Disable: Yes

Corresponding Pylint Error Code: C0303

E222

Description: Used when there are multiple spaces before operator

Examples:



```
total =  grid[i-1][j-weight] + value
```

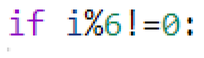
Disable: Yes

Corresponding Pylint Error Code: C0303

E225

Description: Used when there's a missing whitespace before the character around an operator

Examples:



```
if i%6!=0:
```

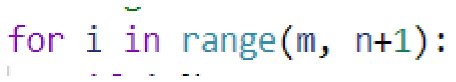
Disable: Yes

Corresponding Pylint Error Code: None

E226

Description: Used when there’s a missing whitespace before the character around an arithmetic operator

Examples:



```
for i in range(m, n+1):
```

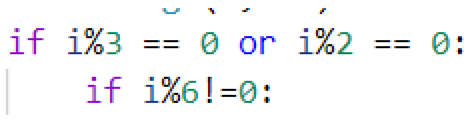
Disable: Yes

Corresponding Pylint Error Code: None

E228

Description: Used when there’s a missing whitespace before the character around a modulo operator

Examples:



```
if i%3 == 0 or i%2 == 0:
    if i%6!=0:
```

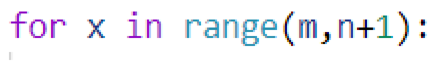
Disable: Yes

Corresponding Pylint Error Code: None

E231

Description: Used when there’s a missing whitespace after the character ‘,’

Examples:



```
for x in range(m,n+1):
```

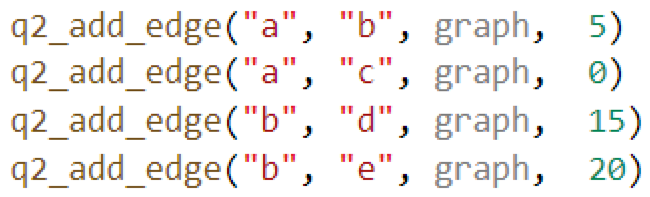
Disable: Yes

Corresponding Pylint Error Code: None

E241

Description: Used when there are multiple spaces after ','

Examples:



```
q2_add_edge("a", "b", graph,  5)
q2_add_edge("a", "c", graph,  0)
q2_add_edge("b", "d", graph,  15)
q2_add_edge("b", "e", graph,  20)
```

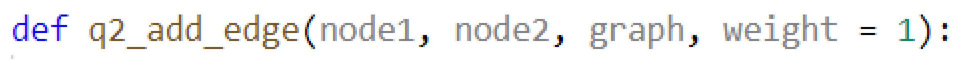
Disable: Yes

Corresponding Pylint Error Code: C0303

E251

Description: Used when there are unexpected spaces around keyword / parameter equals

Examples:



```
def q2_add_edge(node1, node2, graph, weight = 1):
```

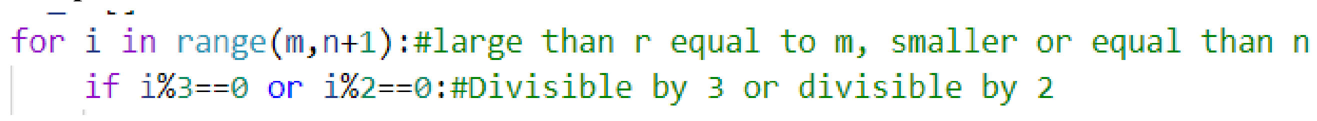
Disable: Yes

Corresponding Pylint Error Code: C0303

E261

Description: Used when there isn't at least two spaces before inline comment

Examples:



```
for i in range(m,n+1):#large than r equal to m, smaller or equal than n
    if i%3==0 or i%2==0:#Divisible by 3 or divisible by 2
```

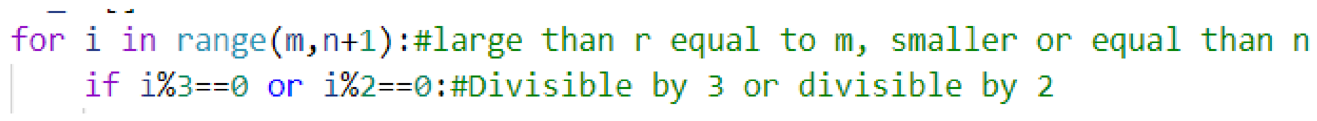
Disable: Yes

Corresponding Pylint Error Code: None

E262

Description: Used when an inline comment doesn't start with a '# (space)'

Examples:



```
for i in range(m,n+1):#large than r equal to m, smaller or equal than n
    if i%3==0 or i%2==0:#Divisible by 3 or divisible by 2
```

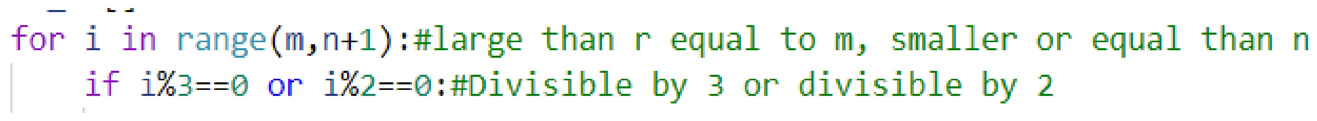
Disable: Yes

Corresponding Pylint Error Code: None

E265

Description: Used when a block comment doesn't start with a '# (space)'

Examples:



```
for i in range(m,n+1):#large than r equal to m, smaller or equal than n
    if i%3==0 or i%2==0:#Divisible by 3 or divisible by 2
```

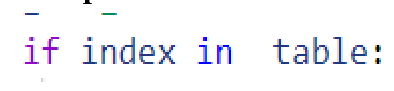
Disable: Yes

Corresponding Pylint Error Code: None

E271

Description: Used when there are multiple spaces after keyword

Examples:



```
if index in  table:
```

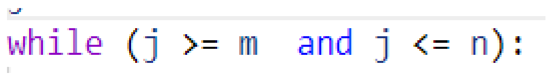
Disable: Yes

Corresponding Pylint Error Code: C0303

E272

Description: Used when there are multiple spaces before a keyword

Examples:



```
while (j >= m  and j <= n):
```

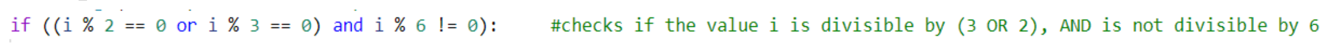
Disable: Yes

Corresponding Pylint Error Code: None

E501

Description: Used when line is too long

Examples:



```
if ((i % 2 == 0 or i % 3 == 0) and i % 6 != 0):     #checks if the value i is divisible by (3 OR 2), AND is not divisible by 6
```

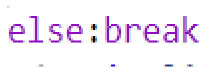
Disable: Yes

Corresponding Pylint Error Code: C0301

E701

Description: Used when there are multiple statements on one line (colon)

Examples:



```
else:break
```

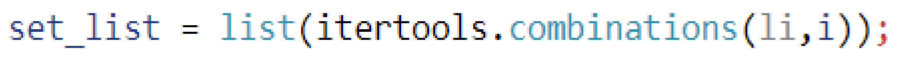
Disable: No

Corresponding Pylint Error Code: None

E703

Description: Used when a statement ends with a semicolon

Examples:



```
set_list = list(itertools.combinations(li,i));
```

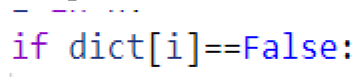
Disable: No

Corresponding Pylint Error Code: W0301

E712

Description: Used when comparison to False should be 'if cond is False:' or 'if not cond:'

Examples:



```
if dict[i]==False:
```

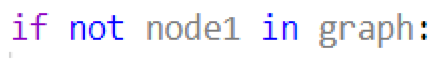
Disable: No

Corresponding Pylint Error Code: C0121

E713

Description: Used when test for membership should be 'not in'

Examples:



```
if not node1 in graph:
```

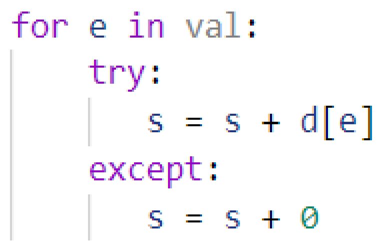
Disable: No

Corresponding Pylint Error Code: None

E722

Description: Used when a bare except is used

Examples:



```
for e in val:
    try:
        s = s + d[e]
    except:
        s = s + 0
```

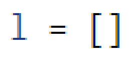
Disable: No

Corresponding Pylint Error Code: W0702

E741

Description: Used when a variable’s name is ambiguous

Examples:



```
l = []
```

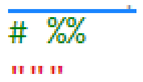
Disable: yes

Corresponding Pylint Error Code: C0103

E902

Description: Used when there is an EOF in multi-line statement

Examples:



```
# %%
""""
```

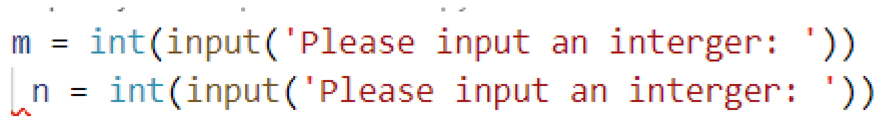
Disable: Yes

Corresponding Pylint Error Code: C0103

E999

Description: Used when a there’s an indentation error

Examples:



```
m = int(input('Please input an interger: '))
 n = int(input('Please input an interger: '))
```

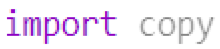
Disable: No

Corresponding Pylint Error Code: W0311

F401

Description: Used when an import is unused

Examples:



```
import copy
```

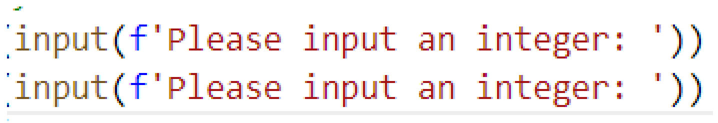
Disable: No

Corresponding Pylint Error Code: W0611

F541

Description: Used when an f string is missing placeholders

Examples:



```
input(f'Please input an integer: '))
input(f'Please input an integer: '))
```

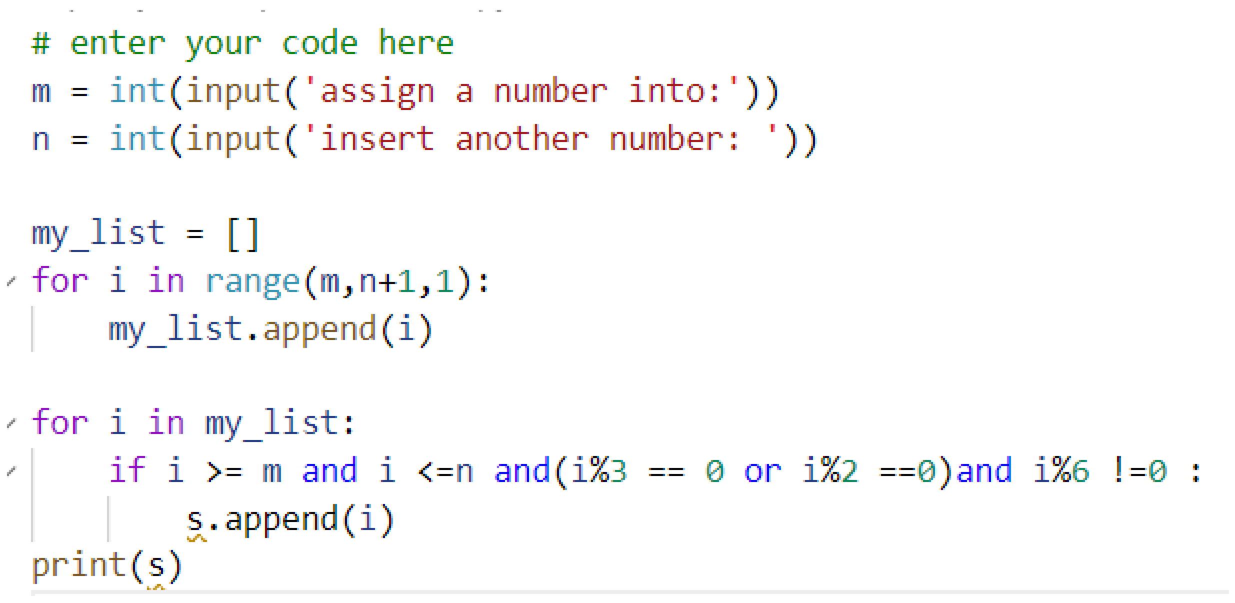
Disable: No

Corresponding Pylint Error Code: None

F821

Description: Undefined name

Examples:



```
# enter your code here
m = int(input('assign a number into:'))
n = int(input('insert another number: '))

my_list = []
for i in range(m,n+1,1):
    my_list.append(i)

for i in my_list:
    if i >= m and i <=n and(i%3 == 0 or i%2 ==0)and i%6 !=0 :
        s.append(i)
print(s)
```

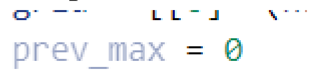
Disable: No

Corresponding Pylint Error Code: E0601

F841

Description: Used when a local variable is assigned to but never used

Examples:



```
prev_max = 0
```

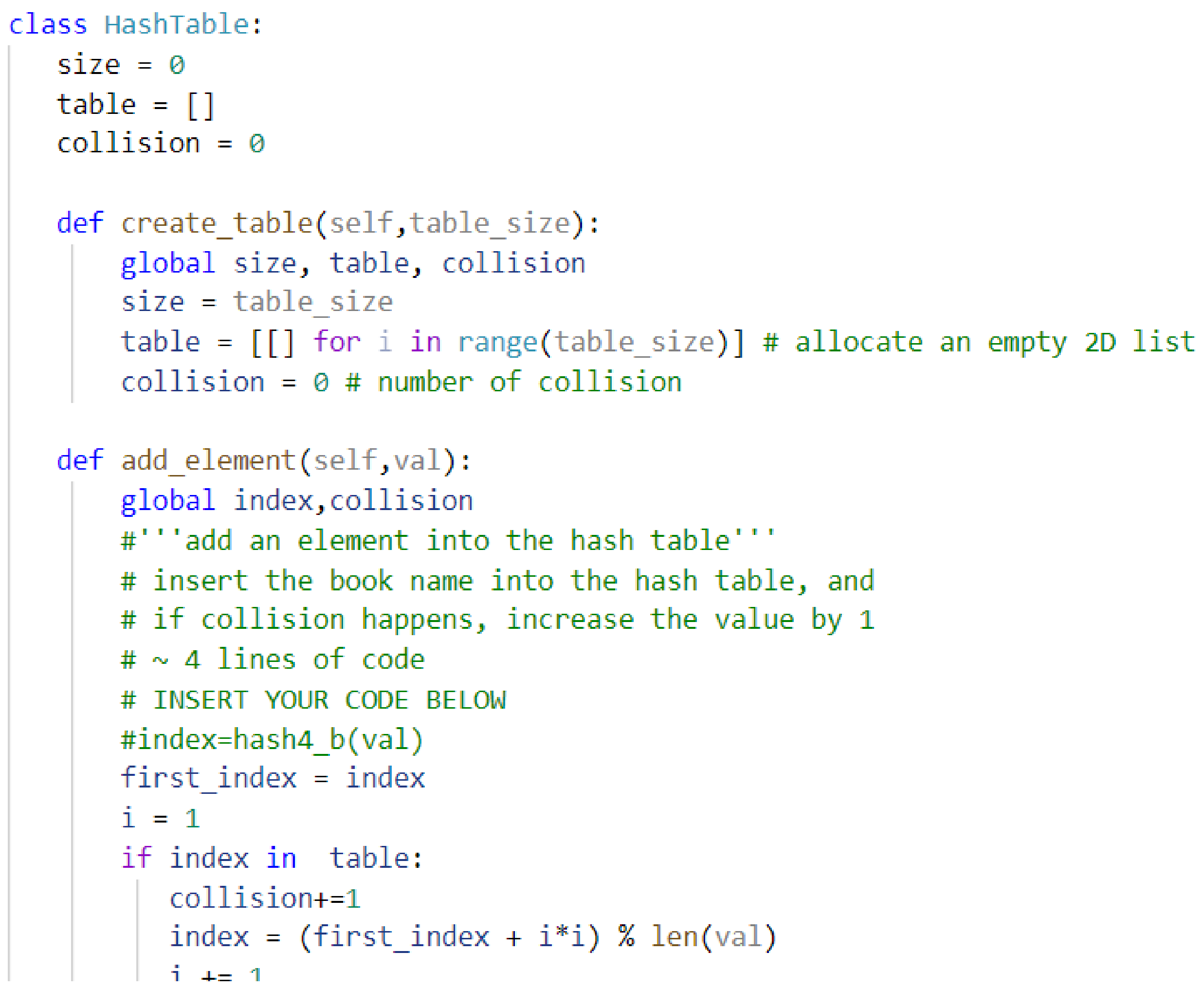
Disable: No

Corresponding Pylint Error Code: W0612

H601

Description: Used when lack of cohesion is too high (100%). Cohesion measures the strength of relationship between pieces of functionality within a given module. When lack of cohesion is low, the methods and variables of the class are co-dependent and hang together as a logical whole. However, if the task requires implementing classes without methods, the lack of cohesion always will be high since all variables will be in-dependent.

Examples:



```
class HashTable:
    size = 0
    table = []
    collision = 0

    def create_table(self,table_size):
        global size, table, collision
        size = table_size
        table = [[] for i in range(table_size)] # allocate an empty 2D list
        collision = 0 # number of collision

    def add_element(self,val):
        global index,collision
        #'''add an element into the hash table'''
        # insert the book name into the hash table, and
        # if collision happens, increase the value by 1
        # ~ 4 lines of code
        # INSERT YOUR CODE BELOW
        #index=hash4_b(val)
        first_index = index
        i = 1
        if index in  table:
            collision+=1
            index = (first_index + i*i) % len(val)
            i += 1
```

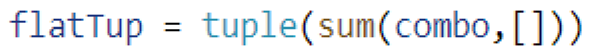
Disable: Yes

Corresponding Pylint Error Code: None

N806

Description: Used when variable name in function is not in lowercase

Examples:



```
flatTup = tuple(sum(combo,[]))
```

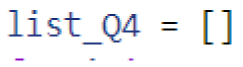
Disable: Yes

Corresponding Pylint Error Code: None

N816

Description: Used when a global variable uses mixedCase

Examples:



```
list_Q4 = []
```

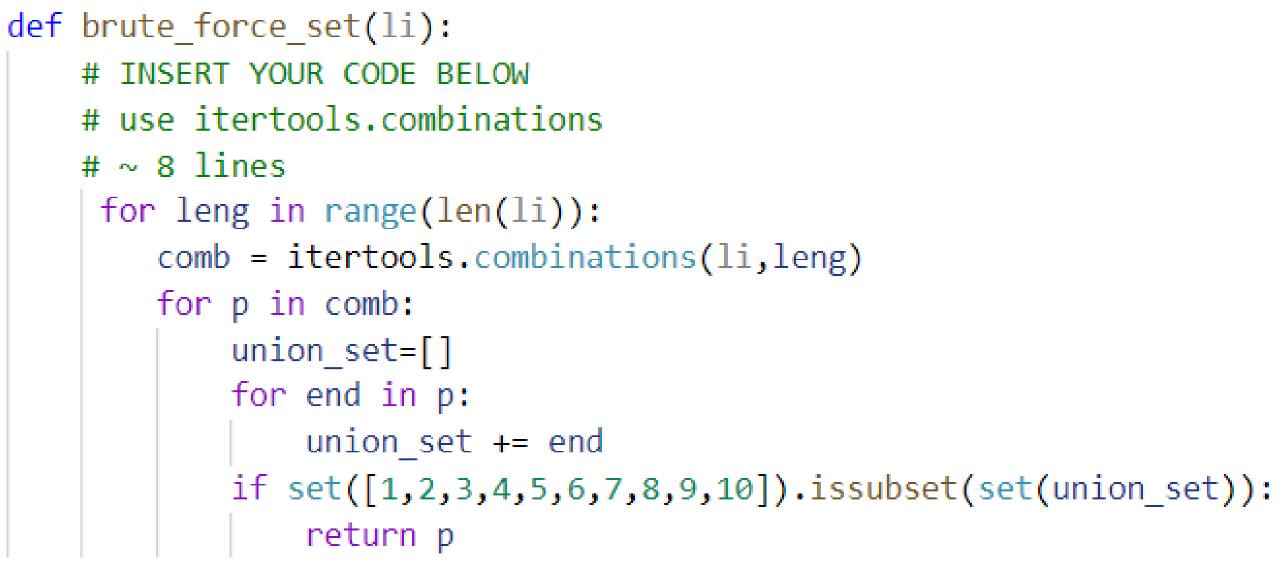
Disable: No

Corresponding Pylint Error Code: C0103

R503

Description: Used when a function with return value does not have a return statement

Examples:


```
def brute_force_set(li):
    # INSERT YOUR CODE BELOW
    # use itertools.combinations
    # ~ 8 lines
     for leng in range(len(li)):
        comb = itertools.combinations(li,leng)
        for p in comb:
            union_set=[]
            for end in p:
                union_set += end
            if set([1,2,3,4,5,6,7,8,9,10]).issubset(set(union_set)):
                return p
```

```
# ================
```

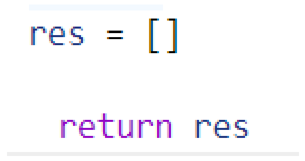
Disable: No

Corresponding Pylint Error Code: R1710

R504

Description: Used when value is assigned to a variable if it will be used only as return value (same as WPS331)

Examples:


```
res = []

  return res
```

Disable: Yes

Corresponding Pylint Error Code: None

SC100

Description: Used when there’s a misspelled word in comments

Examples:

```
# range - larger than or euqal to m and smaller than n
```

Disable: Yes

Corresponding Pylint Error Code: None

SC200

Description: Used when there's a misspelled variable name

Examples:

```
list_n3 = []
```

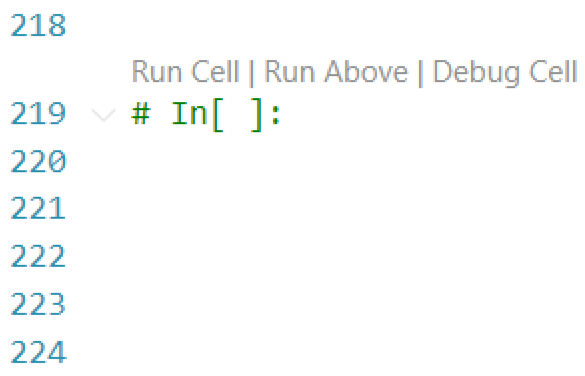
Disable: Yes

Corresponding Pylint Error Code: None

W391

Description: Used when there is a blank line at end of file

Examples:



```
218
219  # In[ ]:
220
221
222
223
224
```

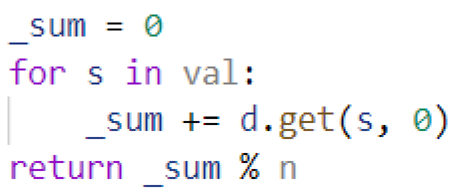
Disable: Yes

Corresponding Pylint Error Code: None

WPS121

Description: Used when usage is found for a variable marked as unused. Variables are marked as unused when they begin with an underscore '_'.

Examples:



```python
_sum = 0
for s in val:
    _sum += d.get(s, 0)
return _sum % n
```

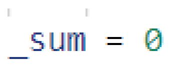
Disable: Yes

Corresponding Pylint Error Code: None

WPS122

Description: Use when definition/initialization for variables marked as unused is found

Examples:



```python
_sum = 0
```

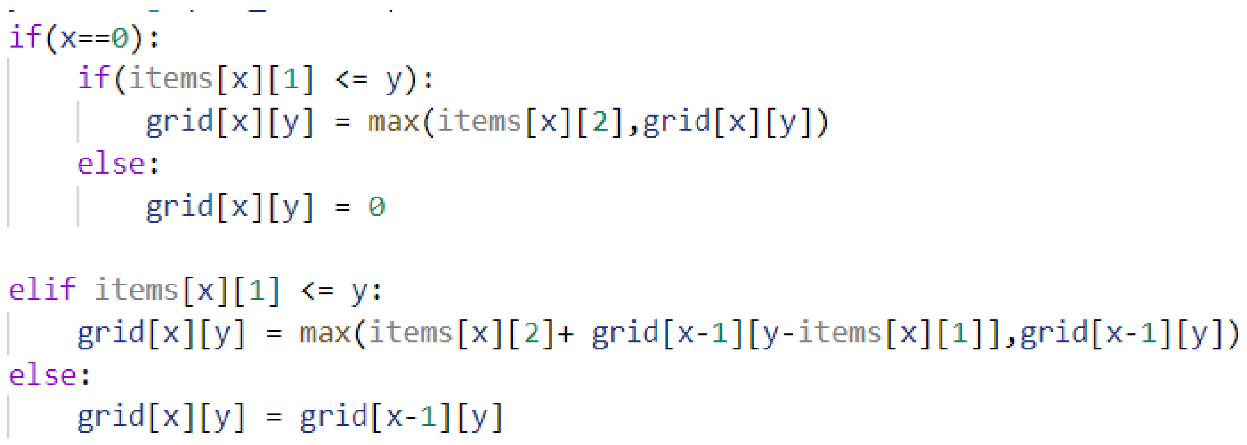
Disable: Yes

Corresponding Pylint Error Code: None

WPS204

Description: Used when an overused expression is found

Examples:



```python
if(x==0):
    if(items[x][1] <= y):
        grid[x][y] = max(items[x][2],grid[x][y])
    else:
        grid[x][y] = 0

elif items[x][1] <= y:
    grid[x][y] = max(items[x][2]+ grid[x-1][y-items[x][1]],grid[x-1][y])
else:
    grid[x][y] = grid[x-1][y]
```

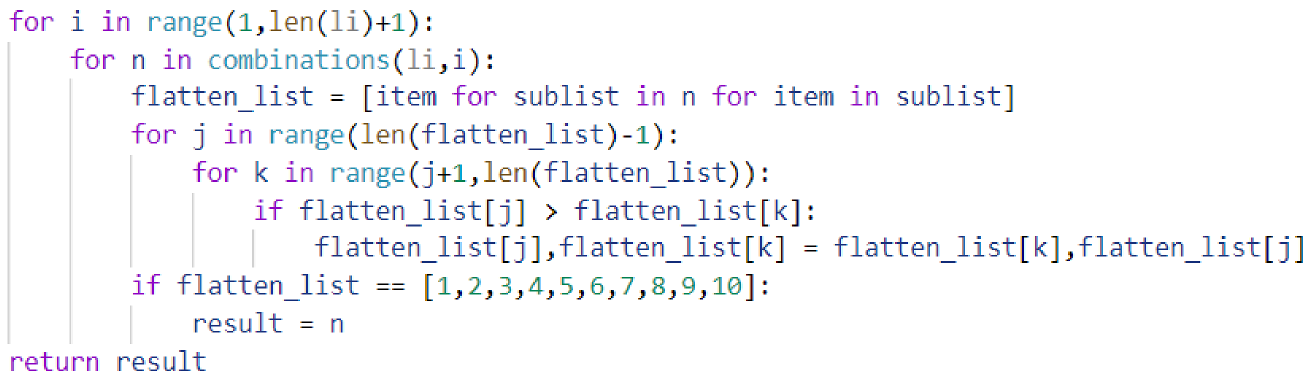
Disable: Yes

Corresponding Pylint Error Code: None

WPS220

Description: Used when deep nesting is found

Examples:



```python
for i in range(1,len(li)+1):
    for n in combinations(li,i):
        flatten_list = [item for sublist in n for item in sublist]
        for j in range(len(flatten_list)-1):
            for k in range(j+1,len(flatten_list)):
                if flatten_list[j] > flatten_list[k]:
                    flatten_list[j],flatten_list[k] = flatten_list[k],flatten_list[j]
        if flatten_list == [1,2,3,4,5,6,7,8,9,10]:
            result = n
return result
```

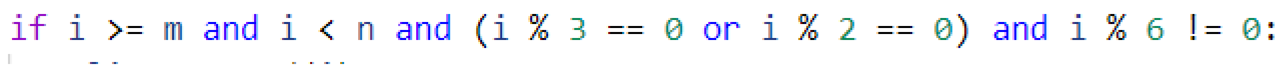
Disable:

Corresponding Pylint Error Code: R1702

WPS222

Description: Used when a condition has too much logic

Examples:



```
if i >= m and i < n and (i % 3 == 0 or i % 2 == 0) and i % 6 != 0:
```

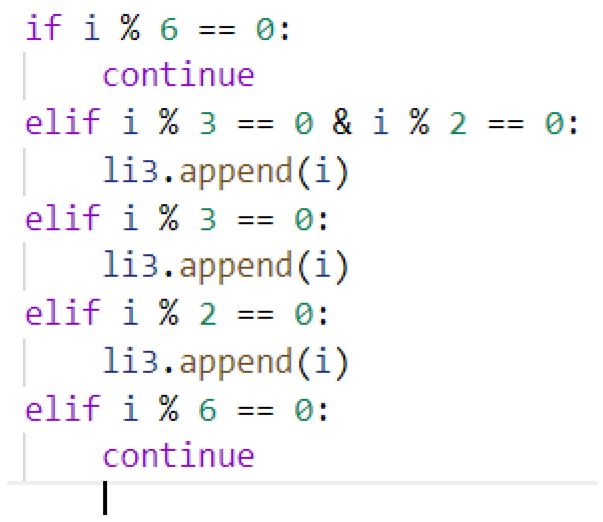
Disable: No

Corresponding Pylint Error Code: None

WPS223

Description: Used when an if statement has too many elif branches

Examples:



```
if i % 6 == 0:
    continue
elif i % 3 == 0 & i % 2 == 0:
    li3.append(i)
elif i % 3 == 0:
    li3.append(i)
elif i % 2 == 0:
    li3.append(i)
elif i % 6 == 0:
    continue
```

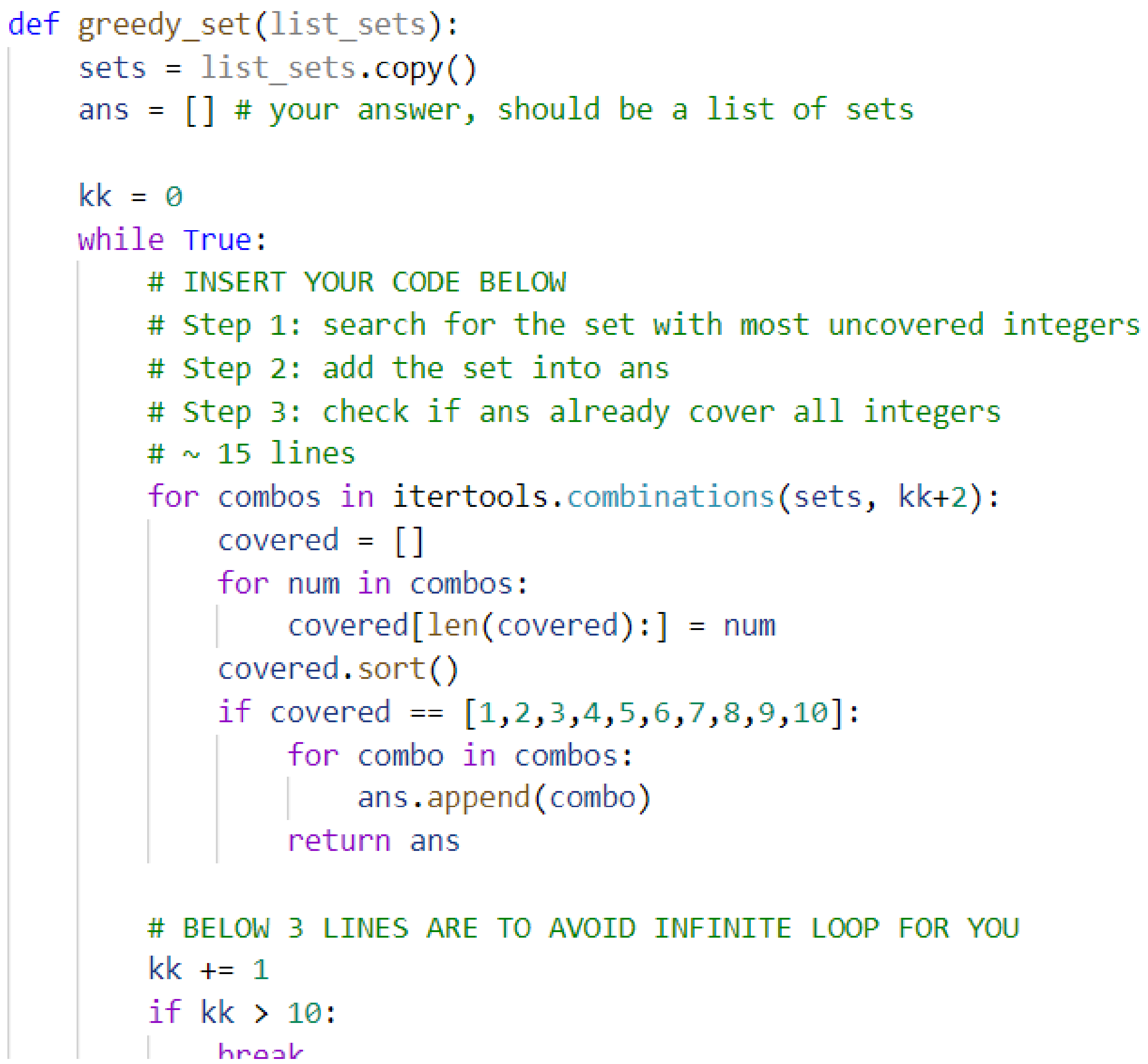
Disable: No

Corresponding Pylint Error Code: None

WPS231

Description: Used when a function with too much cognitive complexity is found. Cognitive complexity is a measure of how difficult a unit of code is to intuitively understand.

Examples:



```python
def greedy_set(list_sets):
    sets = list_sets.copy()
    ans = [] # your answer, should be a list of sets

    kk = 0
    while True:
        # INSERT YOUR CODE BELOW
        # Step 1: search for the set with most uncovered integers
        # Step 2: add the set into ans
        # Step 3: check if ans already cover all integers
        # ~ 15 lines
        for combos in itertools.combinations(sets, kk+2):
            covered = []
            for num in combos:
                covered[len(covered):] = num
            covered.sort()
            if covered == [1,2,3,4,5,6,7,8,9,10]:
                for combo in combos:
                    ans.append(combo)
                return ans

        # BELOW 3 LINES ARE TO AVOID INFINITE LOOP FOR YOU
        kk += 1
        if kk > 10:
            break
```

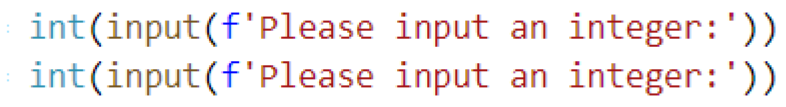
Disable: Yes

Corresponding Pylint Error Code: None

WPS237

Description: Used when an ‘f’ string is too complex

Examples:



```python
int(input(f'Please input an integer:'))
int(input(f'Please input an integer:'))
```

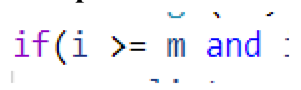
Disable: Yes

Corresponding Pylint Error Code: None

WPS313

Description: Used when a parentheses is found after a keyword

Examples:



```python
if(i >= m and
```

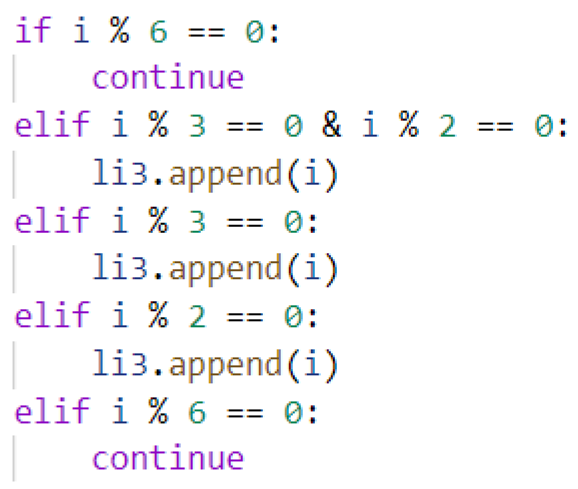
Disable: No

Corresponding Pylint Error Code: C0325

WPS327

Description: Used when there is a useless ‘continue’ at the end of the loop

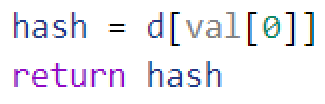
Examples:



```python
if i % 6 == 0:
    continue
elif i % 3 == 0 & i % 2 == 0:
    li3.append(i)
elif i % 3 == 0:
    li3.append(i)
elif i % 2 == 0:
    li3.append(i)
elif i % 6 == 0:
    continue
```

Disable: No
Corresponding Pylint Error Code: None

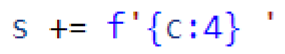
WPS331
Description: Used when variables that are only used as a return value are found.

Examples:

```
hash = d[val[0]]
return hash
```

Disable: Yes

Corresponding Pylint Error Code: None

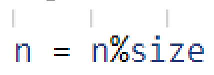
WPS336
Description: Used when explicit string concatenation is found

Examples:

```
s += f'{c:4} '
```

Disable: No

Corresponding Pylint Error Code: None

WPS350
Description: Used when usable augmented assign pattern is found

Examples:

```
n = n%size
```

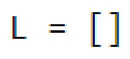
Disable: No

Corresponding Pylint Error Code: None

WPS407

Description: Used when a module constant is mutable. This means that we are able to make changes to a constant variable which contradicts the point of having a constant variable.

Examples:



```
L = []
```

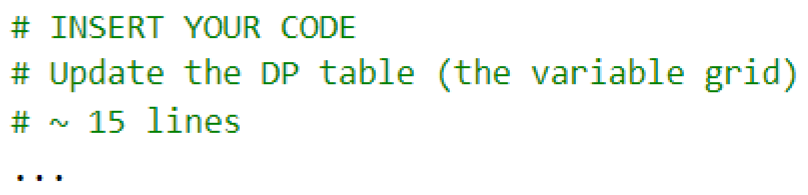
Disable: No

Corresponding Pylint Error Code: None

WPS428

Description: Used when a statement that has no effect is found

Examples:



```
# INSERT YOUR CODE
# Update the DP table (the variable grid)
# ~ 15 lines
...
```

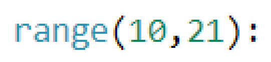
Disable: Yes

Corresponding Pylint Error Code: None

WPS432

Description: Used when there is an unnamed magic number

Examples:



```
range(10,21):
```

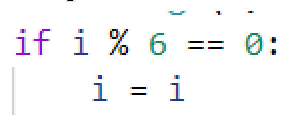
Disable: Yes

Corresponding Pylint Error Code: None

WPS434

Description: Used when a variable is assigned to itself

Examples:



```
if i % 6 == 0:
    i = i
```

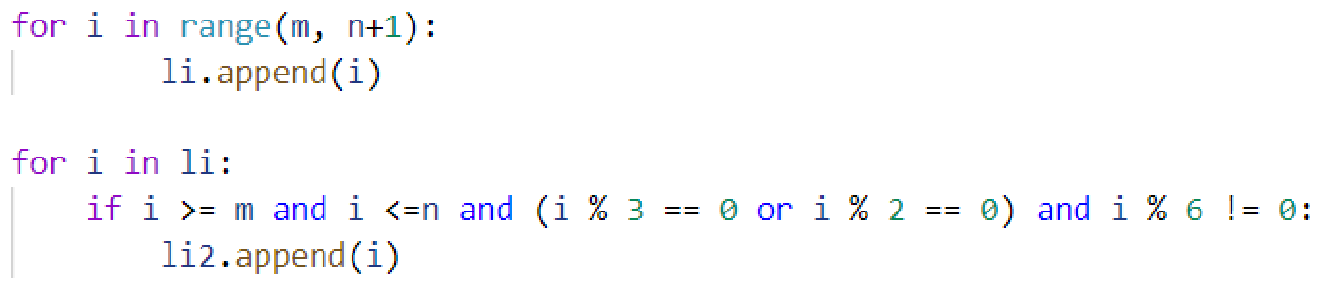
Disable: Yes

Corresponding Pylint Error Code: None

WPS440

Description: Used when several block variables overlap

Examples:



```
for i in range(m, n+1):
        li.append(i)

for i in li:
    if i >= m and i <=n and (i % 3 == 0 or i % 2 == 0) and i % 6 != 0:
        li2.append(i)
```

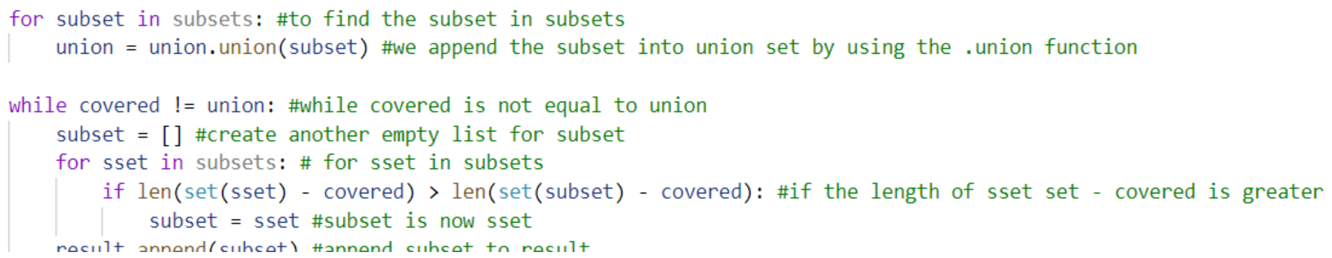
Disable: Yes

Corresponding Pylint Error Code: None

WPS441

Description: Used when Hyperstyle found control variable used after block: subset

Examples:



```
for subset in subsets: #to find the subset in subsets
    union = union.union(subset) #we append the subset into union set by using the .union function

while covered != union: #while covered is not equal to union
    subset = [] #create another empty list for subset
    for sset in subsets: # for sset in subsets
        if len(set(sset) - covered) > len(set(subset) - covered): #if the length of sset set - covered is greater
            subset = sset #subset is now sset
    result.append(subset) #append subset to result
```

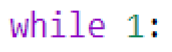
Disable: Yes

Corresponding Pylint Error Code: None

WPS444

Description: Used when an incorrect keyword condition is found

Examples:



```
while 1:
```

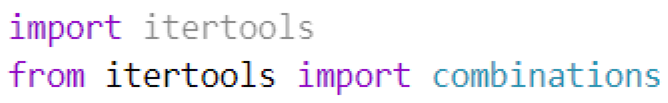
Disable: No

Corresponding Pylint Error Code: None

WPS458

Description: Used when import collision is found

Examples:



```
import itertools
from itertools import combinations
```

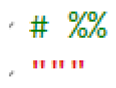
Disable: No

Corresponding Pylint Error Code: W0404

WPS462

Description: Used when wrong multiline string usage is found

Examples:



```
# %%
"""
```

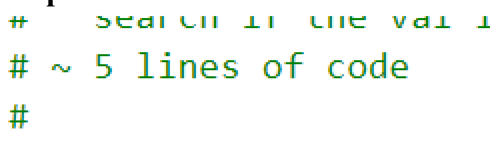
Disable: Yes

Corresponding Pylint Error Code: None

WPS464

Description: Used when found empty comment

Examples:



```
#   search if the val i
# ~ 5 lines of code
#
```

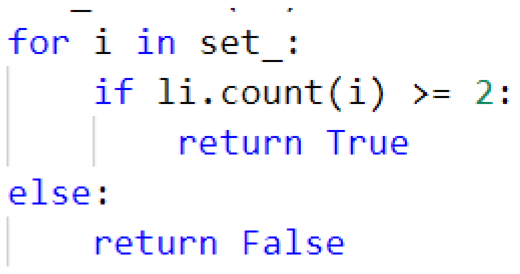
Disable: Yes

Corresponding Pylint Error Code: None

WPS500

Description: Used when found `else` in a loop without `break`

Examples:



```
for i in set_:
    if li.count(i) >= 2:
        return True
else:
    return False
```

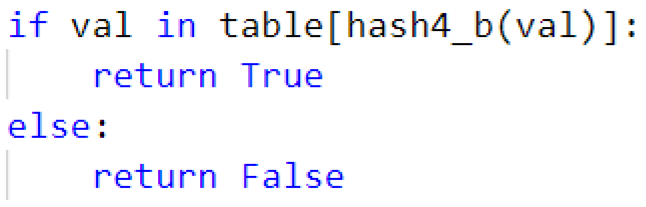
Disable: No

Corresponding Pylint Error Code: None

WPS503

Description: Used when found useless returning `else` statement

Examples:



```
if val in table[hash4_b(val)]:
    return True
else:
    return False
```

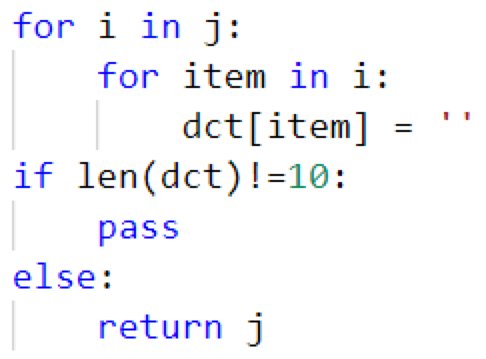
Disable: No

Corresponding Pylint Error Code: R1703

WPS504

Description: Used when found negated condition

Examples:



```
for i in j:
    for item in i:
        dct[item] = ''
if len(dct)!=10:
    pass
else:
    return j
```

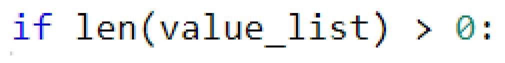
Disable: No

Corresponding Pylint Error Code: None

WPS507

Description: Used when found useless `len()` compare

Examples:



```
if len(value_list) > 0:
```

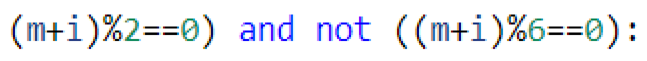
Disable: No

Corresponding Pylint Error Code: None

WPS508

Description: Used when there is an incorrect ‘not’ with compare usage

Examples:



```
(m+i)%2==0) and not ((m+i)%6==0):
```

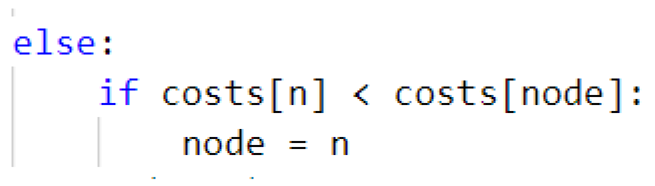
Disable: No

Corresponding Pylint Error Code: None

WPS513

Description: Used when found implicit `elif` condition

Examples:



```
else:
    if costs[n] < costs[node]:
        node = n
```

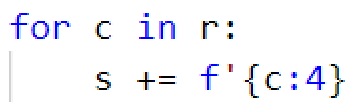
Disable: No

Corresponding Pylint Error Code: None

WPS519

Description: Used when found implicit `sum()` call

Examples:



```
for c in r:
    s += f'{c:4}
```

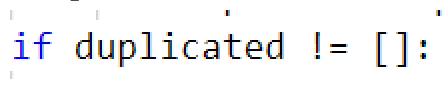
Disable: Yes

Corresponding Pylint Error Code: None

WPS520

Description: Used when found compare with false constant

Examples:



```
if duplicated != []:
```

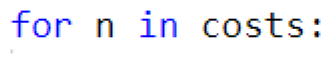
Disable: No

Corresponding Pylint Error Code: None

WPS528

Description: Found implicit `.items()` usage

Examples:



```python
for n in costs:
```

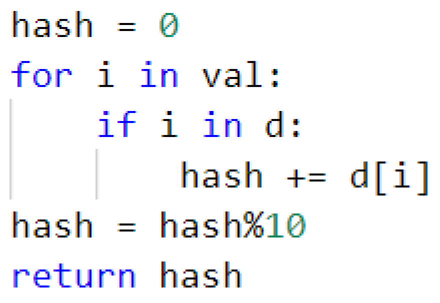
Disable: Yes

Corresponding Pylint Error Code: None

WPS529

Description: Used when found implicit `.get()` dict usage

Examples:



```python
hash = 0
for i in val:
    if i in d:
        hash += d[i]
hash = hash%10
return hash
```

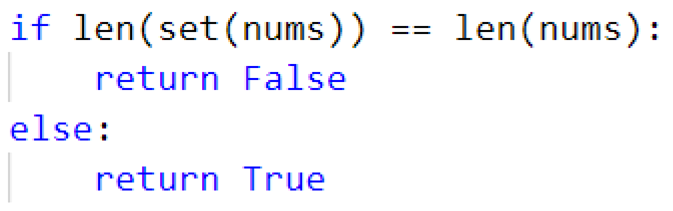
Disable: Yes

Corresponding Pylint Error Code: None

WPS531

Description: Used when found simplifiable returning `if` condition in a function which can be refactored using refactor rule P2.

Examples:



```
if len(set(nums)) == len(nums):
    return False
else:
    return True
```

Disable: No

Corresponding Pylint Error Code: R1703

Qualitative Analysis

- **Hyperstyle generates more specific error code message:**
  - **In Pylint: Variables unused (W0612) / In Hyperstyle: Loop control variables unused (B007)**
  - **In Pylint: Line too long (C0301) / In Hyperstyle: Boolean expression is too long (C001)**
- **Hyperstyle contains many error code messages that are very similar to each other and could be mapped to the same Pylint error code**
  - **Hyperstyle error codes**
    - E111: Used when indentation is not a multiple of a certain number
    - E113: Used when there is unexpected indentation
    - E999: Used when a there's an indentation error
  - **Corresponding Pylint error code**
    - W0311: Used when an unexpected number of indentation's tabulations or spaces has been found.
- **For many instances, Hyperstyle and Pylint catch similar errors/bad practices but by definition they are no the same**
- **However, we also found Hyperstyle and Pylint codes with almost exact definitions but with different error codes, decided to merge them**
- **Hyperstyle catches more detailed Formatting errors than Pylint (eg, whitespaces)**
- **In conclusion, Hyperstyle catches more errors as it goes into more details. However, some of these could be unnecessary most of the time and need to be filtered.**