|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Option** | **Description** | **Input** | **Output** |
| 1 | Combine Import Statements | Combines adjacent import statements into one without changing order of the imports. | import \_\_module\_\_  import jsonLoader  from jsonmodel.validators import jsonModel | import \_\_module\_\_,jsonLoader  from jsonmodel.validators import jsonModel |
| 2 | Remove Pass Statements | Removes pass statements or replaces with literal zero, if a statement is required. | pass  def check():  pass  pass | def check():0 |
| 3 | Remove literal statements (docstrings) | Removes statements that entirely consist of a literal value (including docstrings) or replaces with literal zero, if a statement is required. | """Simple docstring"""  5  b'A'  def test():  "Function documentation" | def test():0 |
| 4 | Remove annotations | Removes function and variable annotations | def test(input:"Input string", output:"Output string"):  a:int = len(input)  b:str  for c in range(a):  b = input[c]  print(b) | def test(input,output):  a=len(input);b:0  for c in range(a):b=input[c];print(b) |
| 5 | Hoist Literals | Replaces string and bytes literals with references to module level variables. | def test\_A():  for m in ['Country', 'Region', 'Area', 'Locality']:  dict.pop(m, None)  if 'XYZ' in dict:  m['Country'] = 'axc'  else:  m['Region'] = 'mvn' | def test\_A():  B='Region';A='Country'  for m in [A,B,'Area','Locality']:dict.pop(m,None)  if'XYZ'in dict:m[A]='axc'  else:m[B]='mvn' |
| 6 | Rename Locals | Shortens any global names | def test\_A(simple\_dict):  if 'XYZ' in simple\_dict:  simple\_dict['Country'] = 'axc'  else:  simple\_dict['Region'] = 'mvn' | def test\_A(simple\_dict):  A=simple\_dict  if'XYZ'in A:A['Country']='axc'  else:A['Region']='mvn' |
| 7 | Rename Globals | Shortens names in the module scope. | import math  a = math.sqrt(679)  c = math.min(a, 39)  print('Ans:')  print(a) | C=print  import math as A  B=A.sqrt(679)  D=A.min(B,39)  C('Ans:')  C(B) |
| 8 | Convert Positional-Only Arguments to Normal Arguments | Converts positional-only arguments into normal arguments by removing the ‘/’ separator in the argument list. | def test(a1, a2, /, \*, a3): pass | def test(a1,a2,\*,a3):pass |