5 Common Python Mistakes and How to Fix Them

Mistake1 : Indentation and Spaces

The **Indentation Error** can occur when the spaces or tabs are not placed properly. If an Indentation Error occurs, the transaltion options of the tabs and spaces on the editor must be changed. Then tabs will automatically be changed to 4 spaces.

```
nums = [11, 30, 44, 54]

for num in nums:
    # Using tabs
    square = num ** 2
    # Using spaces(4)
    print(square)

# Indentation Error : unindent does not match any outer indentation level
```

Mistake2 : Naming Conflicts

The **Import Error** is raised when an import statement has trouble successfully importing the specified module.

```
# math.py in personal directory
from math import radians, sin

rads = radians(90)
print(sin(rads))

# ImportError : cannot import name 'radians' from 'math'
```

The Python interpreter takes outer module first instead of standard library. So ImportError occurs becuase the module of math.py was imported. The name of the math.py should be changed to project.py in this case.

It should also be noted that the module's method name is set to a variable name.

```
from math import radians, sin

radians = radians(90)
rad45 = radians(45)
print(rad45)

# TypeError : 'float' object is not callable
```

Mistakes3 : Mutable Default Args

We must not use mutable default arguments in Python, unless we have really good reason to do so. Because it can lead to all sorts of nasty and horrible bugs.

Mutable default arguments is the case that an empty list was used as a default arguement to a function. In Python, when passing a mutable value as a default argument in a function, the default argument is mutated anytime that value is mutated.

```
# Problem of mutable default args
def add_employee(emp, emp_list=[]):
    emp_list.append(emp)
    print(emp_list)

add_employee('Corey')
# emp_list = ['Corey']
add_employee('John')
# emp_list = ['Corey', 'John']
```

To solve this problem, we must pass mutable default ags eqauls to None.

```
# Solution
def add_employee(emp, emp_list=None):
    if emp_list is None:
        emp_list = []
    emp_list.append(emp)
    print(emp_list)

add_employee('Corey')
# emp_list = ['Corey']
add_employee('John')
# emp_list = ['John']
```

Mistakes4 : Exhausting Iterators

Python3 no longer returns all of those values at once because of memory efficiency. We might use list(zip) to see inside of zip object. This will iterate through the zip object and convert all data into list.

In [7]:
```
names = ['Peter Parker', 'Clark Kent', 'Wade Wilson', 'Bruce wayne']
heroes = ['Spiderman', 'Superman', 'Deadpool', 'Batman']

identities = zip(names, heroes)
print(identities)
for identity in identities:
    print(f"{identity[0]} is actually {identity[1]}!")

# The iterator object has been exhausted!
print(list(identities))
```

```
<zip object at 0x7f4548e3f440>
Peter Parker is actually Spiderman!
Clark Kent is actually Superman!
Wade Wilson is actually Deadpool!
Bruce wayne is actually Batman!
[]
```

Mistakes5 : Importing with *

```
from os import *

rename(filename)
remove()
```

Using asterisk when we import modules can be a bad practice. First of all, it can make our code hard to debug. It can also introduce errors into our code whenever there are two functions with the same name.