CME 211 - 103

Because code should be beautiful

WHAT IS GOOD FORMATTING ?

SPOT THE DIFFERENCES

```python
def is_number( x):
    try:
        int(x      )
        return True
    except: return False

def i_know_how_to_multiply( a,  b   ):
    s=   0
    if (is_number(a ) \
        and is_number( b     )):
        for (i) in range(a   ):
            for (j) in range(   b): s +=      1




    return s

print(i_know_how_to_multiply(4,2))
```

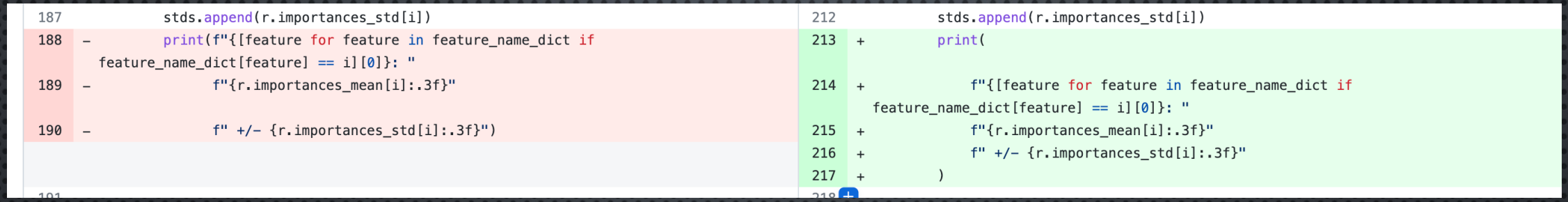
```python
def is_number(x):
    try:
        int(x)
        return True
    except:
        return False


def i_know_how_to_multiply(a, b):
    s = 0
    if is_number(a) and is_number(b):
        for i in range(a):
            for j in range(b):
                s += 1

    return s


print(i_know_how_to_multiply(4, 2))

```

FORMATTING…

```cpp
#include <iostream>
#include <string>

template <typename Traits>
class Information{
    private:
        Traits pf_name;         // fp == public field name
        Traits pf_family;       // fp == public field family

    public:
        void set_information(Traits arg_name, Traits arg_family){
            pf_name = arg_name;
            pf_family == arg_family;
        }

        void get_information(){
            std::cout << "Your name is " << pf_name << " " << pf_family << std::endl;
        }
};

auto main(int argc, char* argv[]) -> decltype(0){
    Information<std::string> o_person;

    o_person.set_information("Milad", "Kahsari Alhadi");
    o_person.get_information();

    return 0;
}
```

- MAKES CODE MORE READABLE
- DEPENDS ON THE LANGUAGE
- IS EVENT MORE IMPORTANT WITH LANGUAGE S THAT DO NOT DEPEND ON INDENTATION.

NUMBER 1 REASON? MERGE CONFLICTS



```
187          stds.append(r.importances_std[i])
188 -        print(f"{[feature for feature in feature_name_dict if
     feature_name_dict[feature] == i][0]}: "
189 -              f"{r.importances_mean[i]:.3f}"
190 -              f" +/- {r.importances_std[i]:.3f}")
```

```
212          stds.append(r.importances_std[i])
213 +        print(
214 +            f"{[feature for feature in feature_name_dict if
     feature_name_dict[feature] == i][0]}: "
215 +            f"{r.importances_mean[i]:.3f}"
216 +            f" +/- {r.importances_std[i]:.3f}"
217 +        )
```

WHEN WORKING ON CODE IN TEAMS, FORMATTING CAN BECOME AN ISSUE QUICKLY

IN PYTHON: USE PYTHON BLACK!

```python
def is_number( x):
    try:
        int(x      )
        return True
    except: return False

def i_know_how_to_multiply( a,  b   ):
    s=   0
    if (is_number(a ) \
        and is_number( b     )):
        for (i) in range(a   ):
            for (j) in range(   b): s +=      1









    return s

print(i_know_how_to_multiply(4,2))
```

```
pip install black
```

```python
def is_number(x):
    try:
        int(x)
        return True
    except:
        return False


def i_know_how_to_multiply(a, b):
    s = 0
    if is_number(a) and is_number(b):
        for i in range(a):
            for j in range(b):
                s += 1

    return s


print(i_know_how_to_multiply(4, 2))
```

```
python black example_formatting.py
```

A FEW FORMATTING RULES

- Double line jumps between functions, single in functions.
- Compactify code in coherent blocks
  - Define variables
  - Perform computation
  - Return
- Add spaces in operations
- Jump lines at the right place when line too long

PRETTY CODE IS MORE MAINTAINABLE

AND IT HELPS THE TAS ☺