Assertions

Assertions : Definition

Dictionary - A confident and forceful statement of fact or belief

Assertions : Definition

Dictionary - A confident and forceful statement of fact or belief

Code - Enforcing the assumptions documented

Assertions : Example

Assertions : Example

```
def greatest_of_two(value1, value2):
```

Assertions : Example

```
def greatest_of_two(value1, value2):
    """Returns whichever value is greater, assuming they are not equal

    params:
    value1(float): First number for comparison
    value2(float): Second number for comparison

    returns(float): greatest among two values
    """
```

Analytics Vidhya

Assertions : Example

```
def greatest_of_two(value1, value2):
    """Returns whichever value is greater, assuming they are not equal

    params:
    value1(float): First number for comparison
    value2(float): Second number for comparison

    returns(float): greatest among two values
    """
```

Analytics Vidhya

Assertions : Example

```
def greatest_of_two(value1, value2):
    """Returns whichever value is greater, assuming they are not equal

    params:
    value1(float): First number for comparison
    value2(float): Second number for comparison

    returns(float): greatest among two values
    """
    if value1 > value2:
        return value1
    else:
        return value2
```

Analytics Vidhya

Assertions : Example

```
def better_greatest_of_two(value1, value2):
    """Returns whichever value is greater, assuming they are not equal

    params:
    value1(float): First number for comparison
    value2(float): Second number for comparison

    returns(float): greatest among two values
    """
    # if values not equal, do nothing : Else return prompt
    assert value1 != value2, "The two values must not be equal"

    if value1 > value2:
        return value1
    else:
        return value2
```

Analytics Vidhya

Assertions : Example

```
def better_greatest_of_two(value1, value2):
    """Returns whichever value is greater, assuming they are not equal

    params:
    value1(float): First number for comparison
    value2(float): Second number for comparison

    returns(float): greatest among two values
    """
    # if values not equal, do nothing : Else return prompt
    assert value1 != value2, "The two values must not be equal"

    if value1 > value2:
        return value1
    else:
        return value2
```

Analytics Vidhya

Assertions : Example

```
better_greatest_of_two(5,5)
```

Assertions : Example

```
better_greatest_of_two(5,5)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-12-4298e51424b2> in <module>
----> 1 better_greatest_of_two(5,5)

<ipython-input-11-47f0c11698be> in better_greatest_of_two(value1, value2)
      9     """
     10     # if values not equal, do nothing : Else return promt
---> 11     assert value1 != value2, "The two values must not be equal"
     12
     13     if value1 > value2:

AssertionError: The two values must not be equal
```

Analytics Vidhya

Assertions for Defensive Programming

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition
- Generally used right before chunk of logical code

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition
- Generally used right before chunk of logical code
- Can be used to check Input

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition
- Generally used right before chunk of logical code
- Can be used to check Input
- Can be used to check output, avoids propagation of bad values

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition
- Generally used right before chunk of logical code
- Can be used to check Input
- Can be used to check output, avoids propagation of bad values
- Can easily detect source of bugs

Assertions for Defensive Programming

- Halts execution on first appearance of unexpected condition
- Generally used right before chunk of logical code
- Can be used to check Input
- Can be used to check output, avoids propagation of bad values
- Can easily detect source of bugs

Notebook

Analytics Vidhya