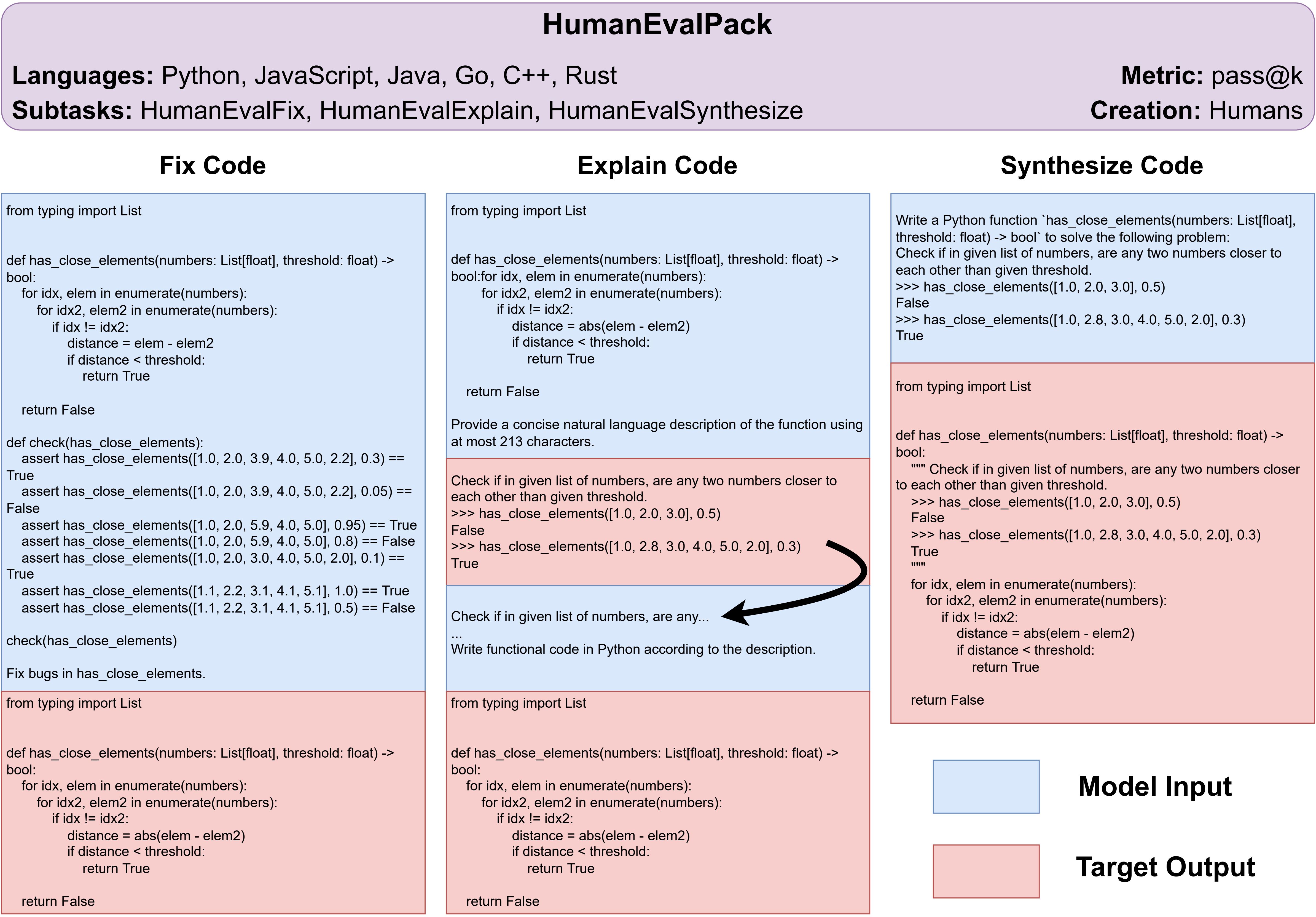

HumanEvalPack

Languages: Python, JavaScript, Java, Go, C++, Rust

Metric: pass@k

Subtasks: HumanEvalFix, HumanEvalExplain, HumanEvalSynthesize

Creation: Humans

Fix Code

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) ->
bool:
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = elem - elem2
                if distance < threshold:
                    return True

    return False

def check(has_close_elements):
    assert has_close_elements([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.3) ==
True
    assert has_close_elements([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.05) ==
False
    assert has_close_elements([1.0, 2.0, 5.9, 4.0, 5.0], 0.95) == True
    assert has_close_elements([1.0, 2.0, 5.9, 4.0, 5.0], 0.8) == False
    assert has_close_elements([1.0, 2.0, 3.0, 4.0, 5.0, 2.0], 0.1) ==
True
    assert has_close_elements([1.1, 2.2, 3.1, 4.1, 5.1], 1.0) == True
    assert has_close_elements([1.1, 2.2, 3.1, 4.1, 5.1], 0.5) == False

check(has_close_elements)

Fix bugs in has_close_elements.
```

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) ->
bool:
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True

    return False
```

Explain Code

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) ->
bool:for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True

    return False

Provide a concise natural language description of the function using
at most 213 characters.
```

```
Check if in given list of numbers, are any two numbers closer to
each other than given threshold.
>>> has_close_elements([1.0, 2.0, 3.0], 0.5)
False
>>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
True
```

```
Check if in given list of numbers, are any...
...
Write functional code in Python according to the description.
```

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) ->
bool:
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True

    return False
```

Synthesize Code

```
Write a Python function `has_close_elements(numbers: List[float],
threshold: float) -> bool` to solve the following problem:
Check if in given list of numbers, are any two numbers closer to
each other than given threshold.
>>> has_close_elements([1.0, 2.0, 3.0], 0.5)
False
>>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
True
```

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) ->
bool:
    """ Check if in given list of numbers, are any two numbers closer
to each other than given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
    for idx, elem in enumerate(numbers):
        for idx2, elem2 in enumerate(numbers):
            if idx != idx2:
                distance = abs(elem - elem2)
                if distance < threshold:
                    return True

    return False
```

🟦 **Model Input**

🟥 **Target Output**