```
class NGramIterator:
    def __init__(self, text, n):
        """Initialize with input text and n-gram size."""
        self.tokens = text.split()  # Simple tokenization
        self.n = n
        self.index = 0  # keep track of the current position

    def __iter__(self):
        """Return the iterator object."""
        return self

    def __next__(self):
        """Return the next n-gram as a string or raise StopIteration."""
        if self.index + self.n > len(self.tokens):  # stop when no more n-grams can
be made
            raise StopIteration

        # create n-gram
        ngram = " ".join(self.tokens[self.index:self.index + self.n])
        self.index += 1  # next position
        return ngram


# Example usage
sample_text = "It’s fun to learn language technology"

# Create iterators for different n-grams
bigram_iterator = NGramIterator(sample_text, 2)
trigram_iterator = NGramIterator(sample_text, 3)
fourgram_iterator = NGramIterator(sample_text, 4)

# results for bigrams
print("Bigrams:")
for bigram in bigram_iterator:
    print(bigram)

# results for trigrams
print("\nTrigrams:")
for trigram in trigram_iterator:
    print(trigram)

# results for four-grams
print("\nFour-grams:")
for fourgram in fourgram_iterator:
    print(fourgram)
```