// PASS — the silent keeper

```
for letter in "Bull":
    pass    # valid — no error, no output

if score >= 60:
    pass    # placeholder — come back later
```

Does nothing. Holds the space. Not interchangeable with continue.

// CONTINUE — the polite skip

```
for letter in "Bull":
    if letter == "u":
        continue    # skip, move to next
    print(letter)

# B / l / l
```

Skips the current iteration. Loop keeps going.

// BREAK — the emergency exit

```
for letter in "Bull":
    if letter == "u":
        break    # stop everything
    print(letter)

# B
```

Stops the loop entirely. Always use with a condition.

// SIDE BY SIDE

```
pass      — does nothing, next line runs
continue  — skips iteration, next item
break     — stops the loop entirely
```

Three different tools. Three different jobs.

// FOR...ELSE

```
for letter in "Bull":
    if letter == "x":
        break
else:
    print("No x found.")    # runs

# else runs   — loop completed without break
# else skipped — break interrupted the loop
# else runs   — sequence was empty
```

else = 'not interrupted'. Not 'nothing matched'.

// SEARCH PATTERN

```
word = input('Enter a word: ')
target = input('Enter a letter: ')

for letter in word:
    if letter == target:
        print(f'Found it.')
        break
else:
    print(f'Not found.')
```

The classic use case for break + for...else.

// COMMON MISTAKES

```
! pass != continue — pass does nothing
! break needs a condition — alone stops at iter 1
! else skipped when break fires
! continue and break only work inside loops
! code after continue/break in same block = unreachable
```