```
delta = np.random.choice([-1,1], steps)
```

Which of the following snippets are equivalent to the above NumPy code (steps >= 0). All NumPy vectors should be Python lists.

A.
```
delta = random.choice([-1,1])
```

B.
```
delta = steps * random.choice([-1,1])
```

C.
```
delta = []
for i in range(2):
    delta.append(random.choice([-1,1]))
```

D.
```
delta = []
for i in range(steps):
    delta.append(random.choice([-1,1]))
```

Answer: D
The NumPy code creates a vector of length size of random integers that are either -1 or 1. So we want to create a list of length size of similar random integers using the random.choice function.

```
delta = np.random.choice([-1,1], steps)
walk = np.cumsum(delta)
```

Which of the following snippets are equivalent to the above NumPy code (steps is >= 0). NumPy vectors should be Python lists.

A.
```
pos = 0
walk = []
for i in range(steps):
  pos += random.choice([-1, 1])
  walk.append(pos)
```

B.
```
walk = []
for i in range(steps):
  delta=random.choice([-1,1])
  walk.append(delta)
```

C.
```
walk = []
for i in range(steps):
  if random.randint(0,1) == 0:
    walk.append(-1)
  else:
    walk.append(1)
```

D.
```
pos = 0
for i in range(steps):
  pos += random.choice([-1, 1])
```

Answer: A

delta is an array of randomly sample +1, -1, the cumsum function computes the cumulative sum, i.e., for [a, b, c] it computers [a, a+b, a+b+c]. Answer A implements the latter via the pos accumulator.