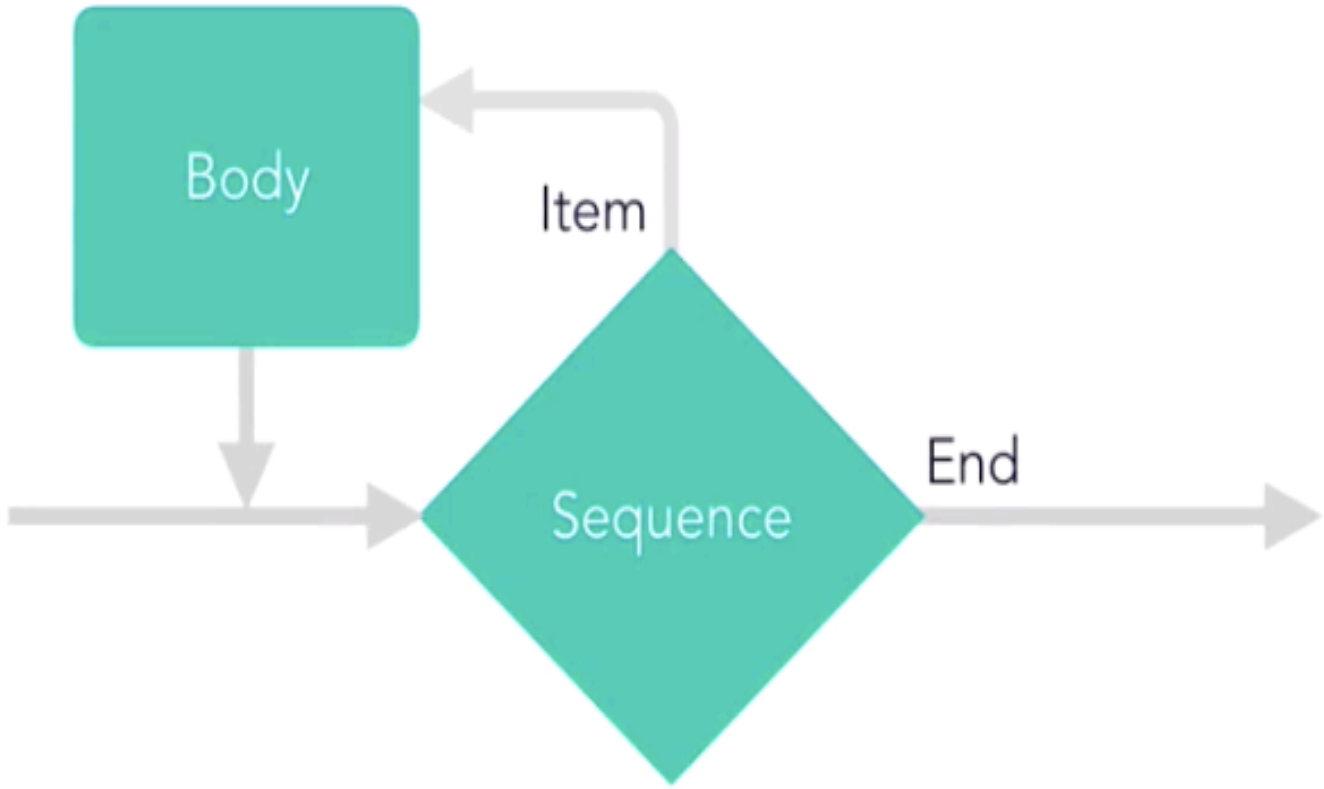
Python Loops

Overview

For Loops

- Uses a sequence e.g. iterator, list, tuple
- Loop is executed for each item in sequence
- Loop ends when the sequence is exhausted



For loop

- Uses a sequence
- The index of the first object starts with zero and so on

```
animals = ('bear', 'bunny','dog', 'cat', 'velociraptor')

for pet in animals:
    print(pet)
```

```
bear
bunny
dog
cat
velociraptor
```

```
1 animals = ('bear', 'bunny','dog', 'cat', 'velociraptor')
2
3 for i in range(5):
4     print(animals[i])
```

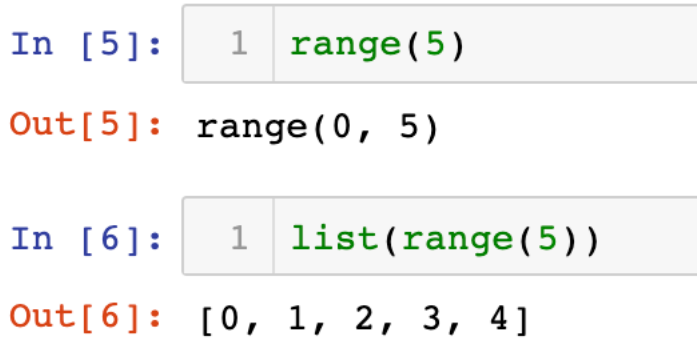
```
bear
bunny
dog
cat
velociraptor
```

Range Function

Python has a built-in function called range that creates an iterator for a list of numbers.=

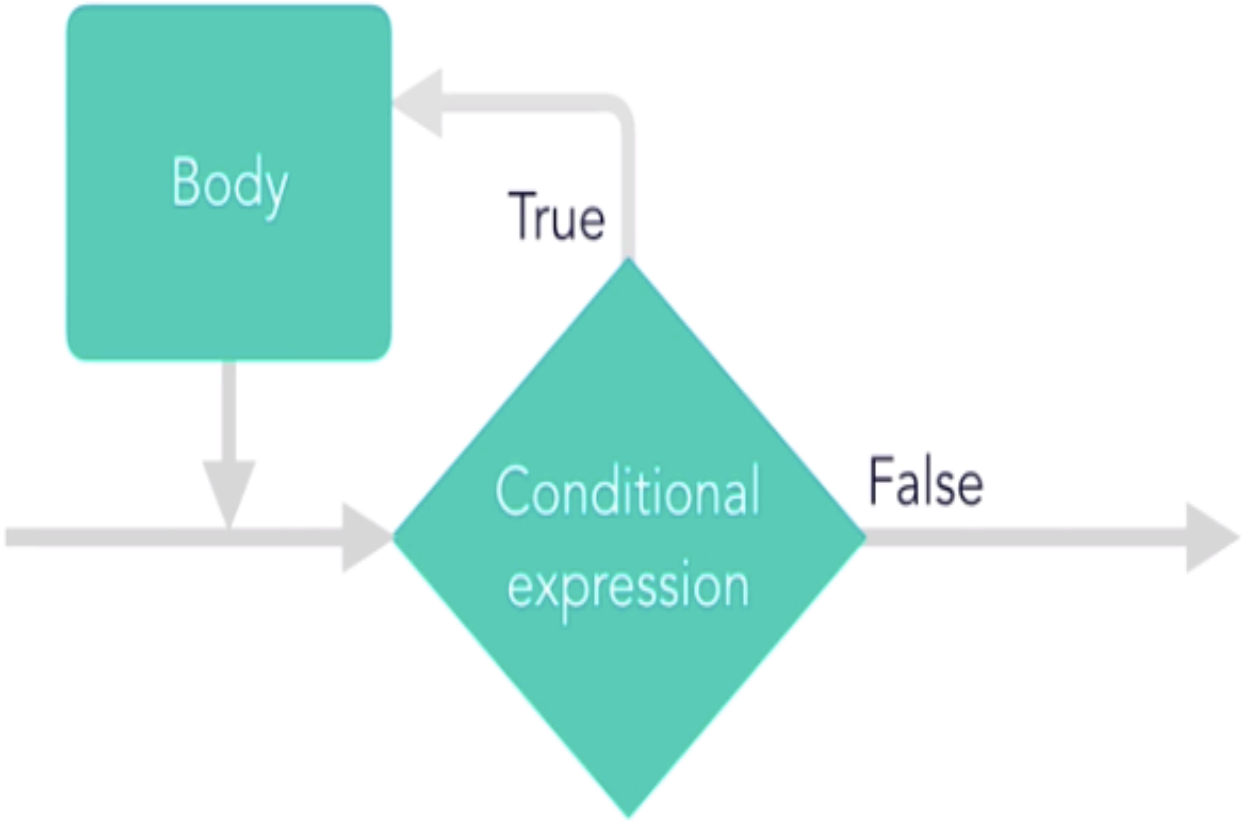
For example:

- - range(3) iterator for [0, 1, 2],
- - range(2, 5) iterator for [2, 3, 4],
- - range(5,2,-1) iterator for [5,4,3]



While Loops

- Uses conditional expression
- Executes when the condition is True
- Continuous looping process while condition is true



While loops

- Conditional expression- while variable ‘pw’ is not equal to the secret word ‘swordfish’
- Will continue the loop till this condition is met

```
In [*]: secret = 'swordfish'
        pw = ''

        while pw != secret:
            pw = input("What's the secret word? ")
```

```
What's the secret word?
```

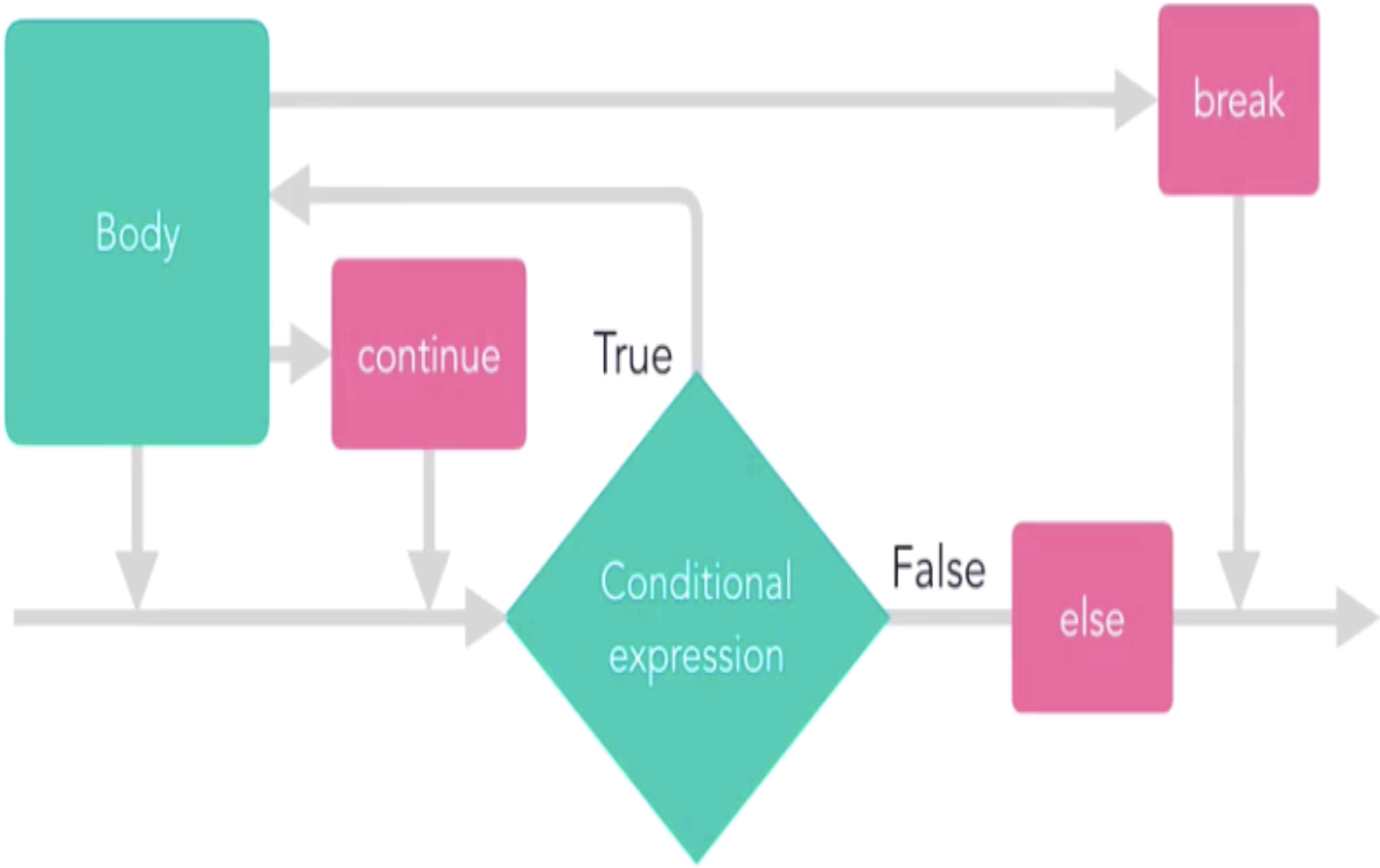
Loops and Number of Iterations

- For loops repeat the body as many times as the length of the sequence it iterates through (if not interrupted by a condition).
- Since the sequence has a finite length, it won’t go to infinite (unless the sequence is being dynamically updated in each iteration)

- While loops repeat the body as long as its condition is met.
- Therefor, if the variables in condition statement are not updated (or updated but the condition is still met) then the loop will go to infinity (infinite loop)

Additional controls

- Three additional controls- Continue, Break and else.
- Continue: used to shortcut a loop and start again
- Break: used to break out of the loop prematurely
- Else: executes only if loop ends normally



Additional controls

- Loop continues till the condition is met and maximum attempt are exhausted

```
secret = 'swordfish'
pw = ''
auth = False
count = 0
max_attempt = 5

while pw != secret:
    count +=1
    pw = input(f"{count}:What's the secret word? ")
```

```
secret = 'swordfish'
pw = ''
auth = False
count = 0
max_attempt = 5

while pw != secret:
    count +=1
    if count > max_attempt: break
    pw = input(f"{count}:What's the secret word? ")

else:
    auth = True
    print ("Authorized" if auth else "Calling the FBI...")
```

Additional controls

- Same control works in for loop
- All three additional controls are extremely useful

```
animals = ('bear', 'bunny','dog', 'cat', 'velociraptor')

for pet in animals:
    if pet == 'dog': continue
    if pet == 'velociraptor': break
    print(pet)
else:
    print('That is all of the animals')
```

```
bear
bunny
cat
```

Making “smart” loops

The trick is “knowing” something about the whole loop when you are stuck writing code that only sees one entry at a time

Set some variables to initial values

for thing in data:

- check for something
- calculate a value
- update a variable

Look at the variables

Nested Loops