Python Conditions and Loops

FPT Education
FPT UNIVERSITY

- Conditional Statement
- Loop Statement
- Iterators
- Problem Solving : Cross Selling and Up Selling

Python Conditions and Loops

FPT Education
FPT UNIVERSITY

- How to ask Python questions; how to make reasonable use of the answers
  - a mechanism which will allow you to do something if a condition is met, and not do it if it isn’t.
  - To make such decisions, Python offers a special instruction. Due to its nature and its application, it's called a conditional instruction (or conditional statement).

1. Conditional Statement

- **If Statement** *: Use if statement to execute a block of Python code, if the condition is true.*
- *Any non-zero value or nonempty container is considered TRUE, whereas Zero, None, and empty container is considered FALSE.*
- *Example :*

```
x, y = 7, 5
if x > y:
    print('x is greater')
    # Prints x is greater
```

```
if true_or_not:
    do_this_if_true
```

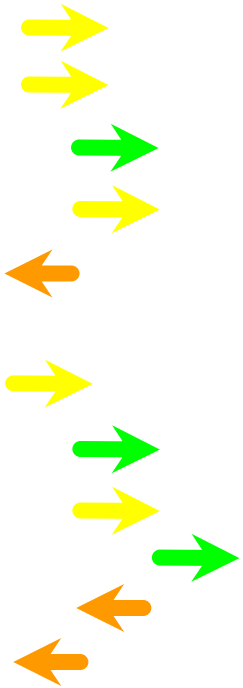
- the `if` keyword;
- one or more white spaces;
- an expression (a question or an answer) whose value will be interpreted solely in terms of `True` (when its value is non-zero) and `False` (when it is equal to zero);
- a **colon** followed by a newline;
- an **indented** instruction

Indentation

- Increase indent indent after an if statement or for statement (after :)
- Maintain indent to indicate the scope of the block (which lines are affected by the if/for)
- Reduce indent back to the level of the if statement or for statement to indicate the end of the block
- Blank lines are ignored - they do not affect indentation
- Comments on a line by themselves are ignored with regard to indentation

increase / maintain after if or for

decrease to indicate end of block



```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

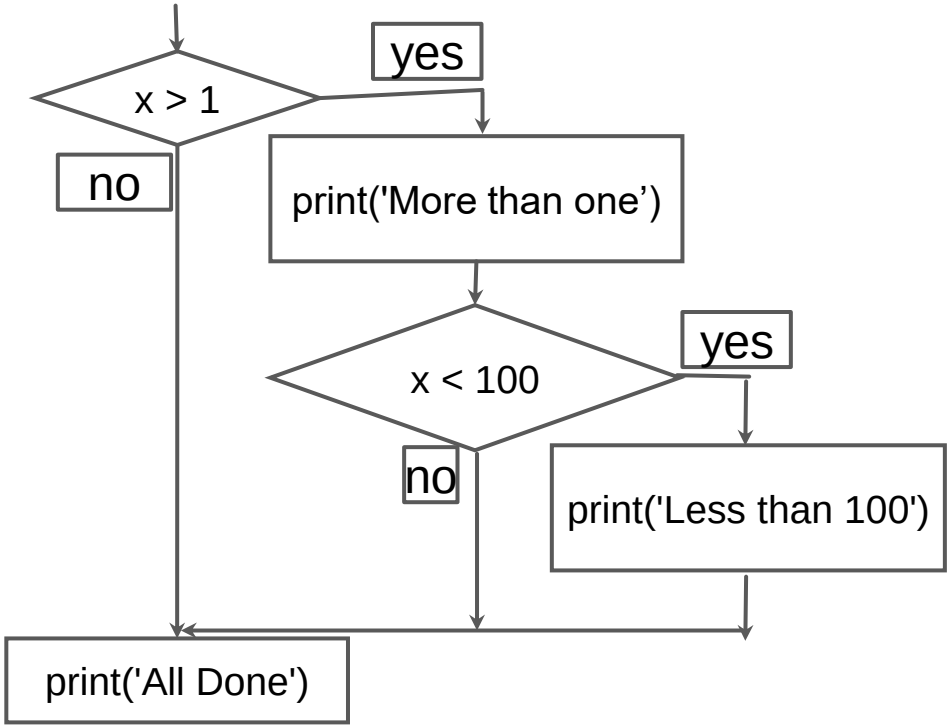
Think About begin/end Blocks

FPT Education
FPT UNIVERSITY

```
x = 5
if x > 2 :
    print('Bigger than 2')
    print('Still bigger')
print('Done with 2')

for i in range(5) :
    print(i)
    if i > 2 :
        print('Bigger than 2')
    print('Done with i', i)
print('All Done')
```

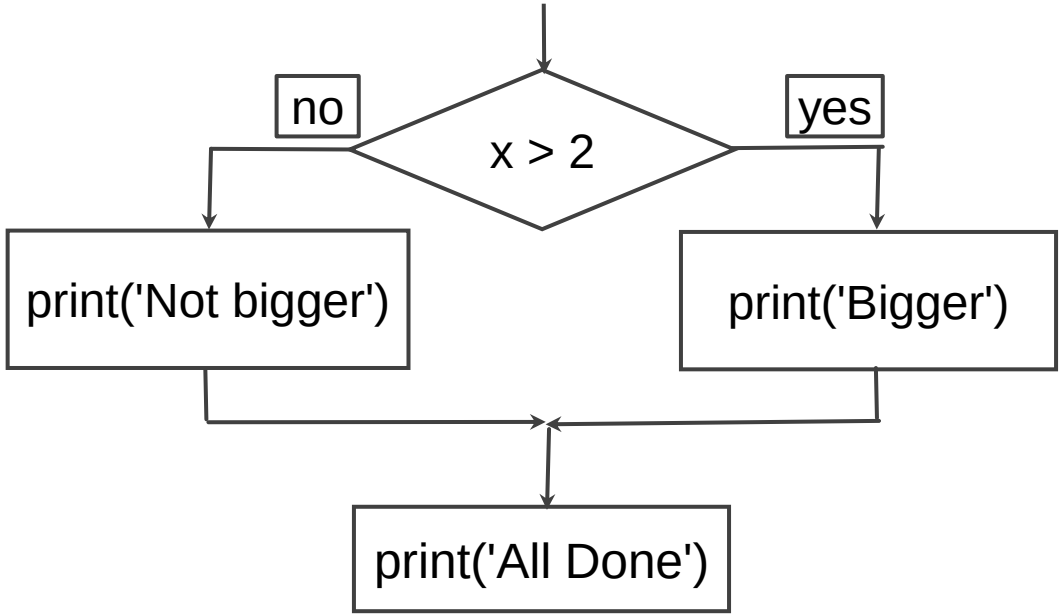
Nested Decisions

```
x = 42
if x > 1 :
    print('More than one')
    if x < 100 :
        print('Less than 100')
print('All done')
```



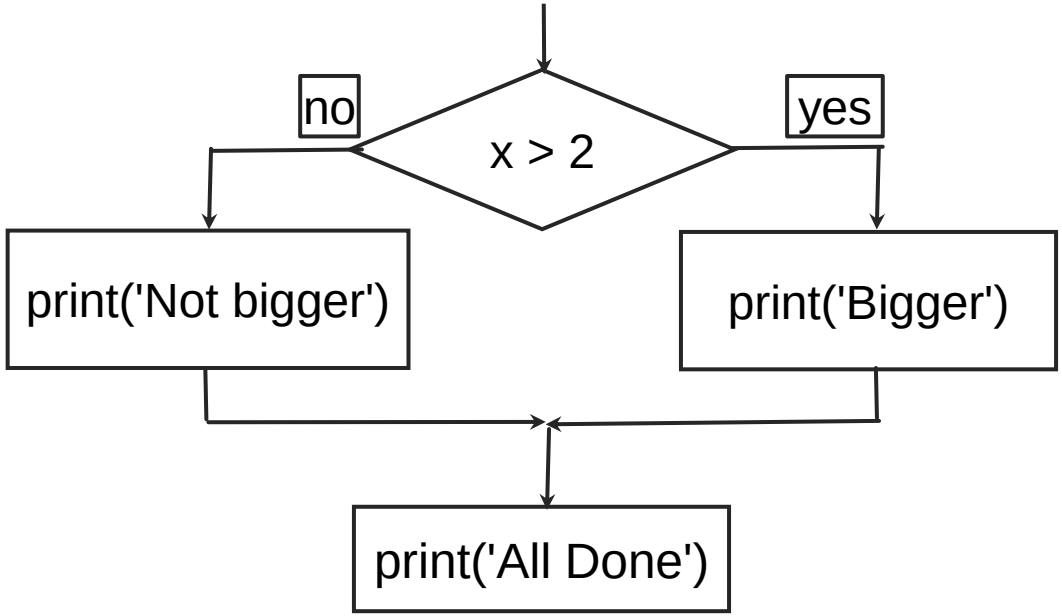
Two-way Decisions

- Sometimes we want to do one thing if a logical expression is true and something else if the expression is false
- It is like a fork in the road - we must choose one or the other path but not both



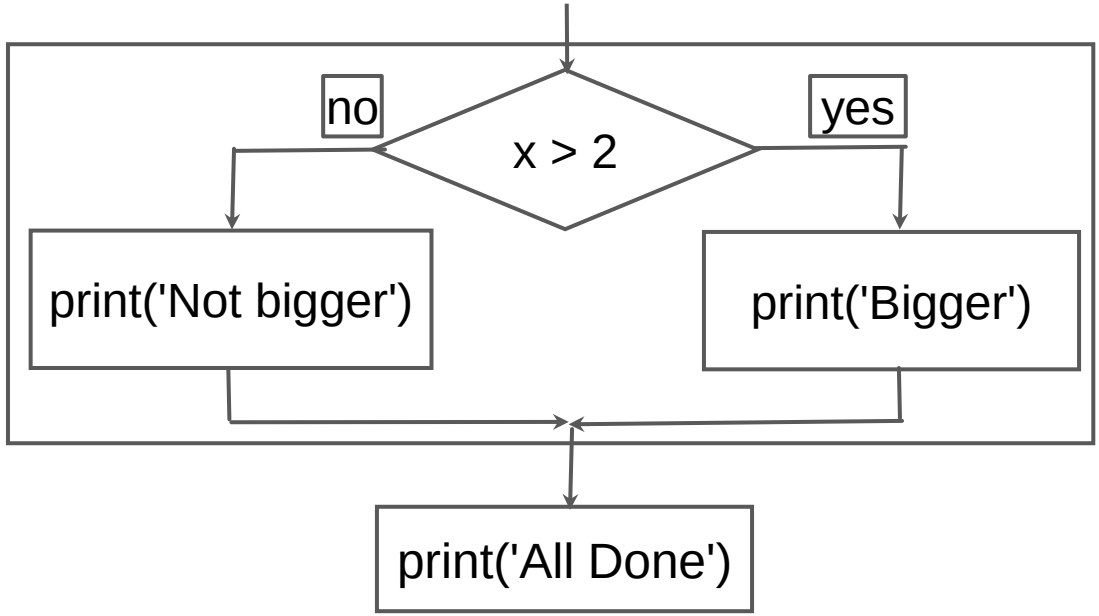
Two-way Decisions with else:

```
x = 4

if x > 2 :
    print('Bigger')
else :
    print('Smaller')

print('All done')
```



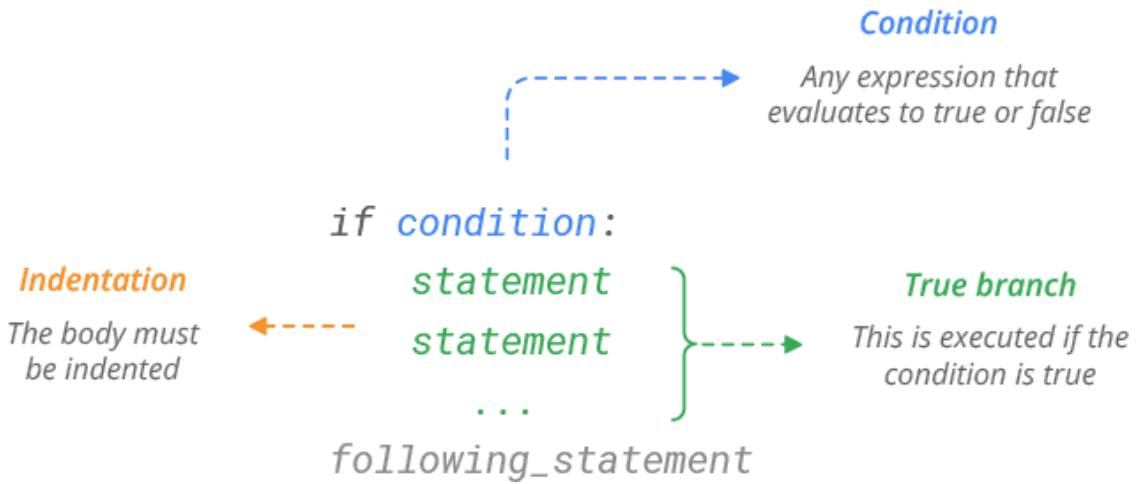
Visualize Blocks

```
x = 4

if x > 2 :
    print('Bigger')
else :
    print('Smaller')

print('All done')
```



Conditional Statement

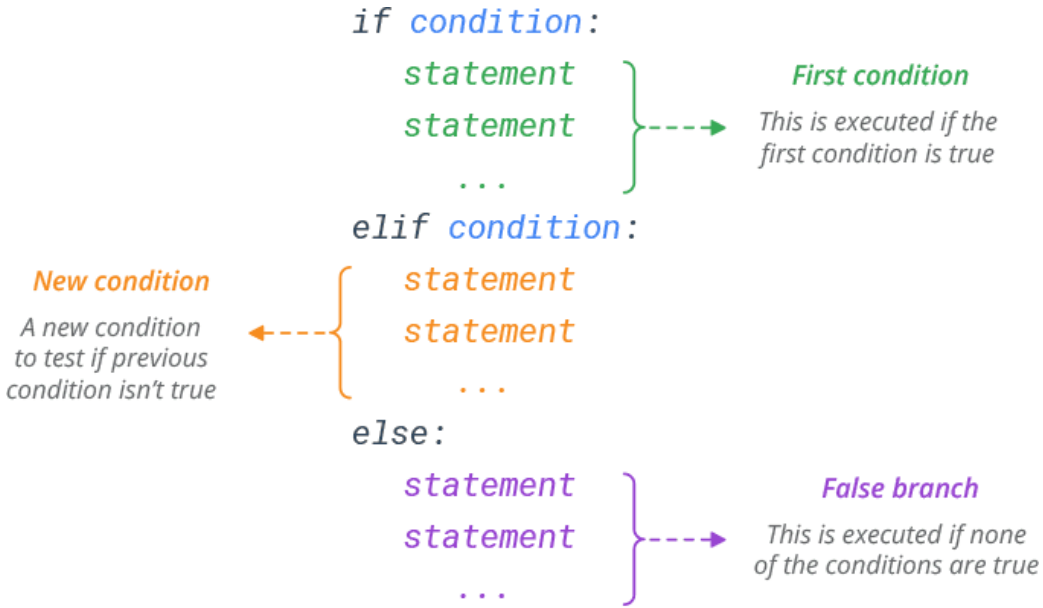
Syntax If



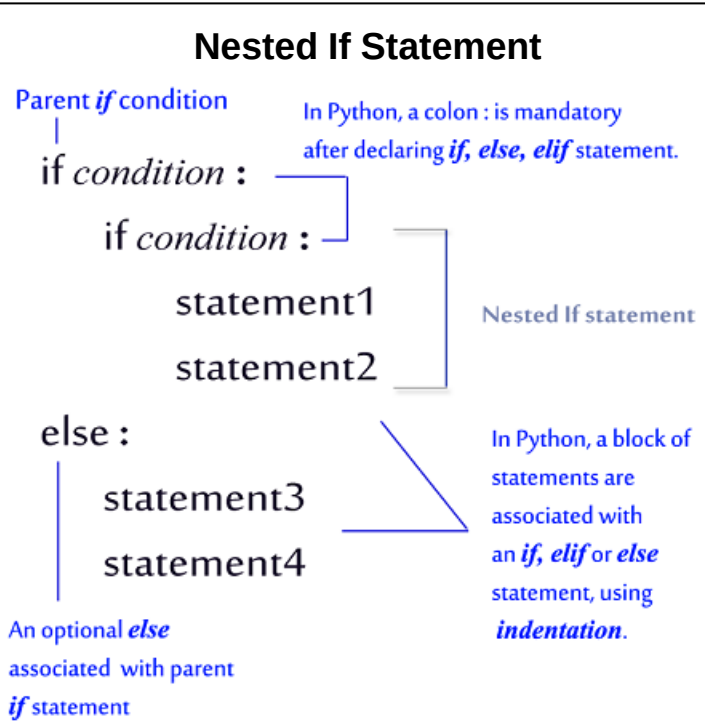
Conditional Statement

FPT Education
FPT UNIVERSITY

Syntax elif



Conditional Statement



Conditional Statement

FPT Education
FPT UNIVERSITY

Comparison operators

Operator	Meaning	Example
==	Equals	if x == y
!=	Not equals	if x != y
>	Greater than	if x > y
>=	Greater than or equal to	if x >= y
<	Less than	if x < y
<=	Less than or equal to	if x <= y

Multi-way Puzzles

Which will never print regardless of the value for x?

```
if x < 2 :
    print('Below 2')
elif x >= 2 :
    print('Two or more')
else :
    print('Something else')
```

```
if x < 2 :
    print('Below 2')
elif x < 20 :
    print('Below 20')
elif x < 10 :
    print('Below 10')
else :
    print('Something else')
```

Loop Statement

- **For in Loop Statement :** Python for loop can iterate over a sequence of items. The structure of a for loop in Python is different than that in C++ or Java.
- **Example:**

```
# Prints all letters except 'e' and 's'
for letter in 'geeksforgeeks':
        if letter == 'e' or letter == 's':
                continue
        else:
                print('Current Letter :', letter)
```

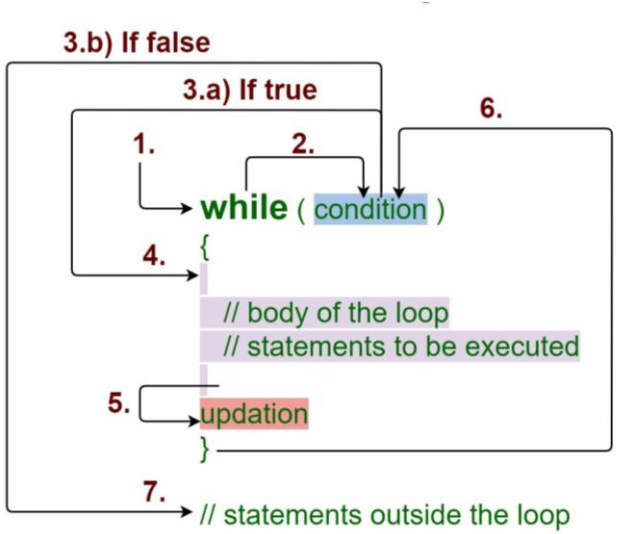
Loop Statement

- **While in Loop Statement :** A while loop in python iterates till its condition becomes False. In other words, it executes the statements under itself while the condition it takes is True**.**
- **Example:**

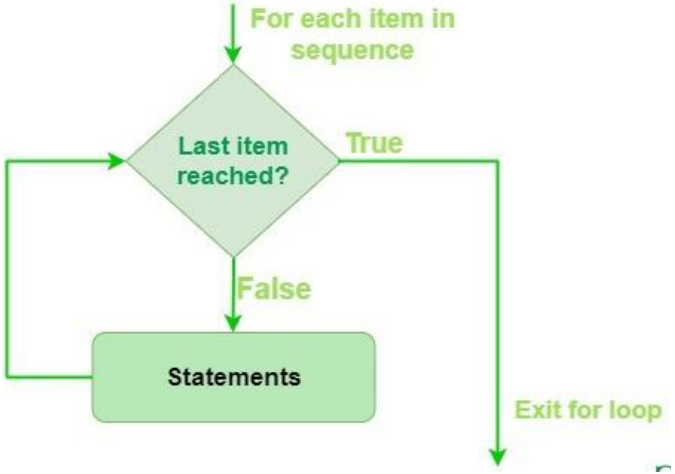
```
i = 1
while True:
        print(i)
        i = i + 1
        if(i > 3):
                break
```

Loop Statement

While In Loop



For In Loop



The break and continue statements

- It appears that it's unnecessary to continue the loop as a whole; you should refrain from further execution of the loop's body and go further;
- It appears that you need to start the next turn of the loop without completing the execution of the current turn.

Loop Statement

- **Break in Loop Statement :** If the break statement is inside a nested loop (loop inside another loop), the break statement will terminate the innermost loop.
- **Continue in Loop Statement :** The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.
- Example :

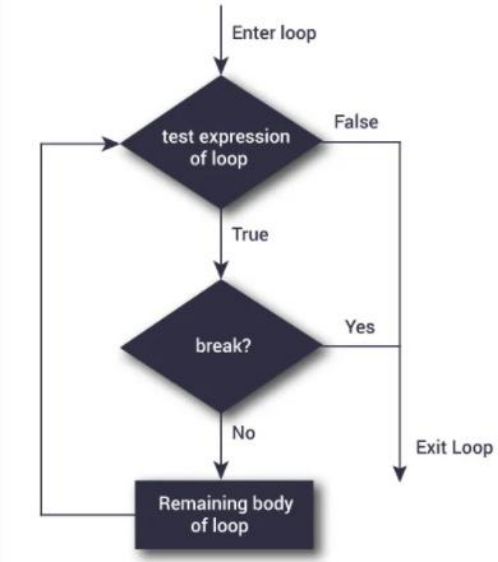
```
for val in "string":
        if val == "i":
            break
        print(val)
```

```
for val in "string":
        if val == "i":
            continue
        print(val)
```

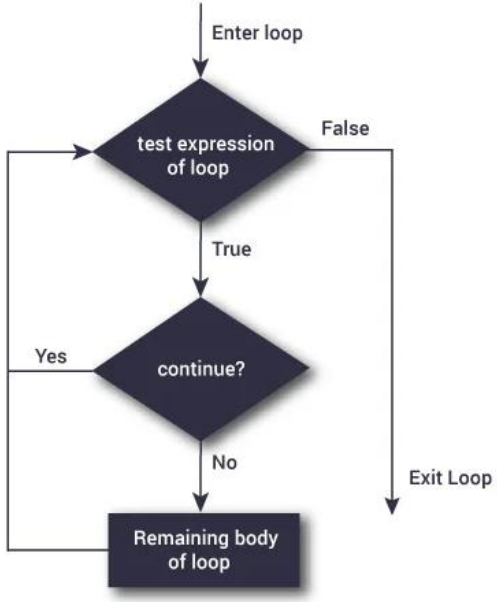
Loop Statement

FPT Education
FPT UNIVERSITY

Break In Loop



Continue In Loop



The while loop and the else branch

FPT Education
FPT UNIVERSITY

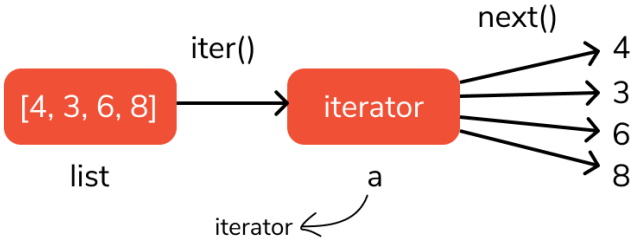
- As you may have suspected, loops may have the else branch too, like ifs.
- The loop's else branch is always executed once, regardless of whether the loop has entered its body or not.

```
i = 5
while i < 5:
    print(i)
    i += 1
else:
    print("else:", i)
```

```
i = 111
for i in range(2, 1):
    print(i)
else:
    print("else:", i)
```

3. Iterators

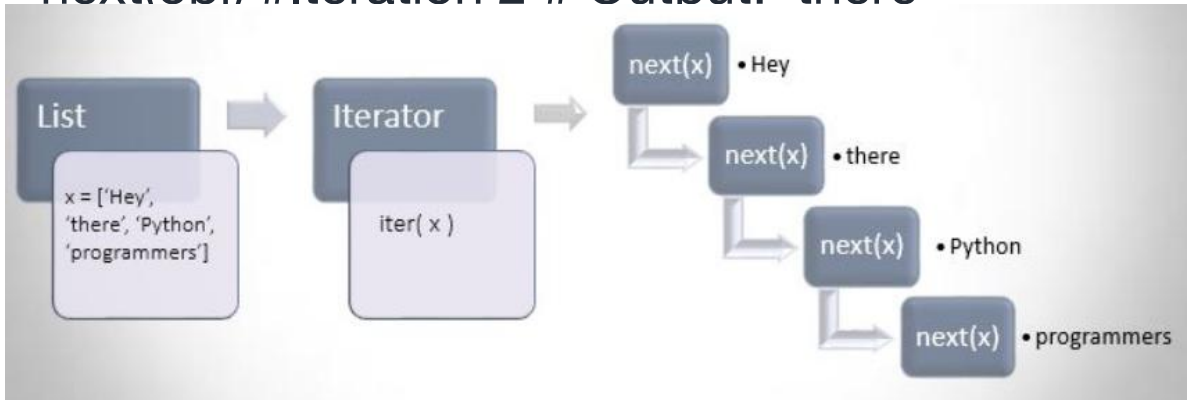
- Iterator simply is an object that can be iterated upon. It allows programmers to access or traverse through all the elements of the collection without any deeper understanding of its structure.
- Python iterators implement iterator protocol which consists of two special methods __iter__() and __next__(). The __iter__() method returns an iterator object where as __next__() method returns the next element from the sequence



Iterators

- Example:

```
x = ['Hey','there','Python','programmers']
obj = iter(x) #using iter function for x
next(obj) #Iteration 1 # Output: 'Hey'
next(obj) #Iteration 2 # Output: 'there'
```



4. Problem Solving : Cross Selling and Up Selling Problem

- Cross-selling is a sales technique used to get a customer to spend more by purchasing a product that's related to what's being bought already.
- Up-selling is a sales technique where a seller induces the customer to purchase more expensive items, upgrades or other add-ons in an attempt to make a more profitable sale.