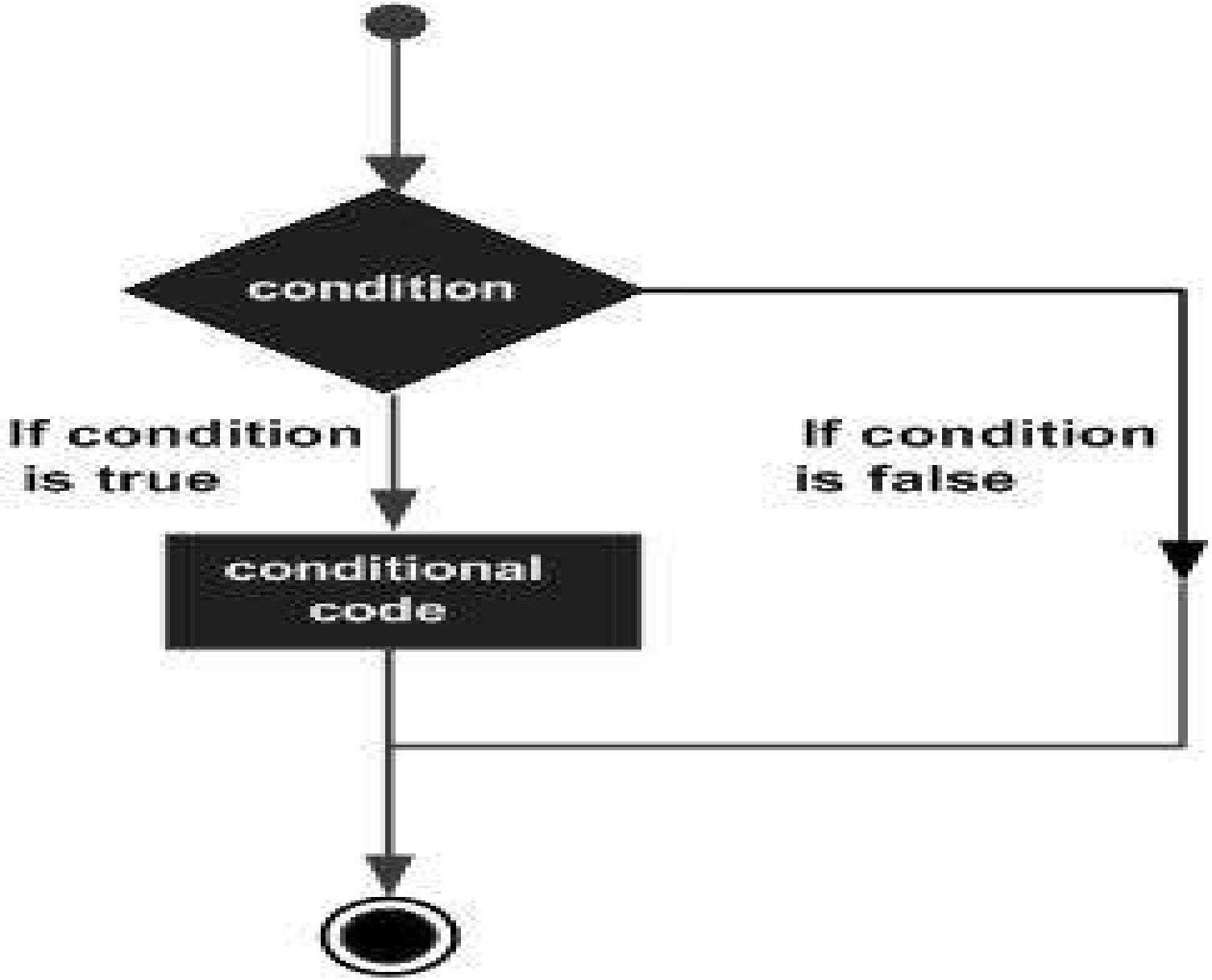
Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.
Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise.


condition
If condition
is true
If condition
is false
conditional
code

1	if statementsAn **if statement** consists of a boolean expression followed by one or more statements.
2	if...else statementsAn **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
3	nested if statementsYou can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Nested If

You can have `if` statements inside `if` statements, this is called *nested* `if` statements.

Example

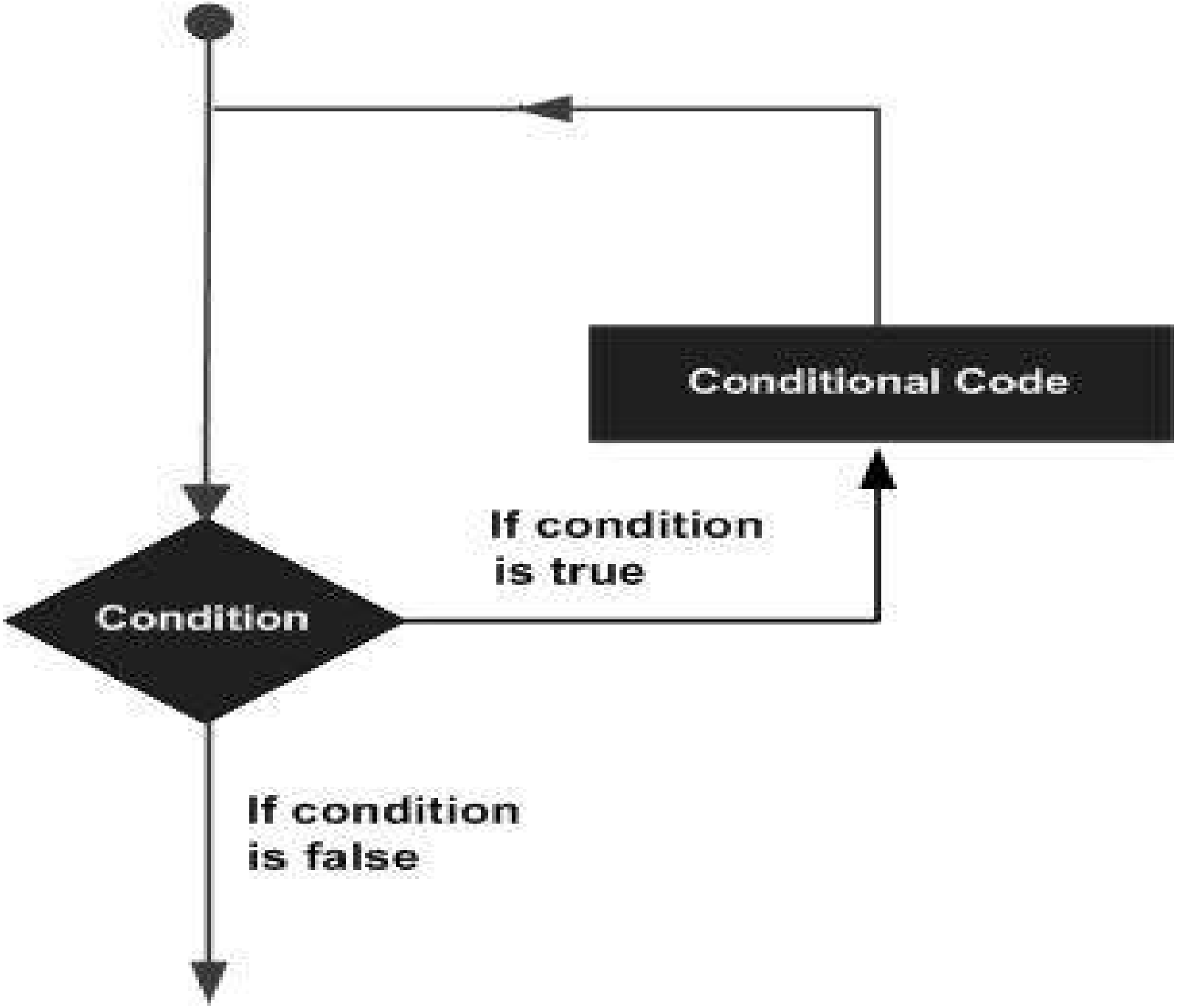
```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times.


Conditional Code
If condition
is true
Condition
If condition
is false

Sr.No.	Loop Type & Description
1	while loopRepeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
2	for loopExecutes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	nested loopsYou can use one or more loop inside any another while, for or do..while loop.

Python Loops

Python has two primitive loop commands:

while loops
for loops

The while Loop

With the while loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Python For Loops

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string). This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

```
fruits =["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Nested Loops

A nested loop is a loop inside a loop.

The "inner loop" will be executed one time for each iteration of the "outer loop":

Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```