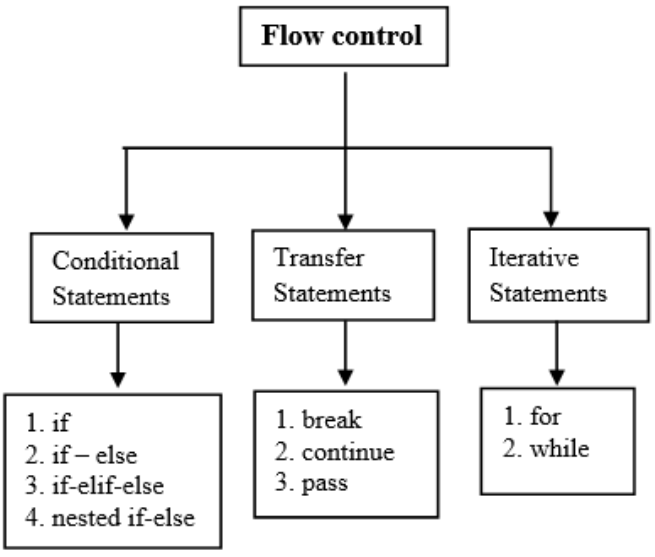
Python Operators Precedence (highest to lowst)

Precedence level	Operator	Meaning
1 (Highest)	`()`	Parenthesis
2	`**`	Exponent
3	`+x, -x ,~x`	Unary plus, Unary Minus, Bitwise negation
4	`*, /, //, %`	Multiplication, Division, Floor division, Modulus
5	`+, -`	Addition, Subtraction
6	`<<, >>`	Bitwise shift operator
7	`&`	Bitwise AND
8	`^`	Bitwise XOR
9	`|`	Bitwise OR
10	`==, !=, >, >=, <, <=`	Comparison
11	`is, is not, in, not in`	Identity, Membership
12	not	Logical NOT
13	and	Logical AND
14 (Lowest)	or	Logical OR

Python Data Types

Data type	Description	Example
`int`	To store integer values	`n = 20`
`float`	To store decimal values	`n = 20.75`
`complex`	To store complex numbers (real and imaginary part)	`n = 10+20j`
str	To store textual/string data	`name = 'Jessa'`
bool	To store boolean values	`flag = True`
list	To store a sequence of mutable data	`l = [3, 'a', 2.5]`
tuple	To store sequence immutable data	`t =(2, 'b', 6.4)`
dict	To store key: value pair	`d = {1:'J', 2:'E'}`
set	To store unorder and unindexed values	`s = {1, 3, 5}`
frozenset	To store immutable version of the set	`f_set=frozenset({5,7})`
range	To generate a sequence of number	`numbers = range(10)`
bytes	To store bytes values	`b=bytes([5,10,15,11])`

Python Control Flow Statements



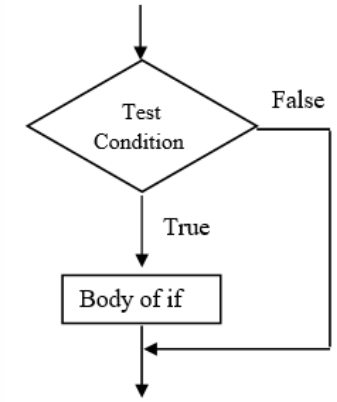


Fig. Flowchart of if statement

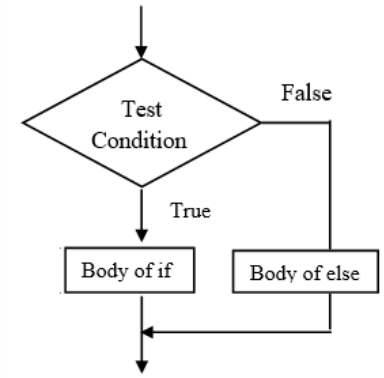


Fig. Flowchart of if-else

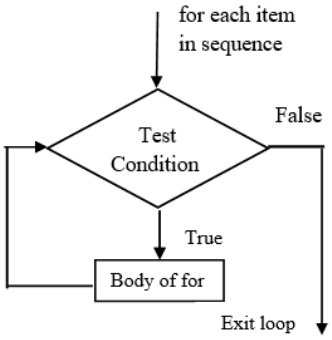


Fig. Flowchart of for loop

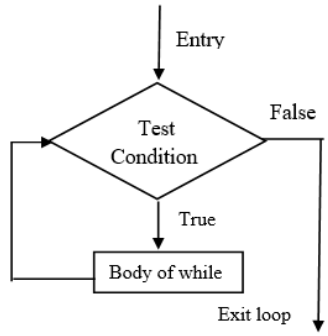


Fig. Flowchart of while loop

Python for loop

A for loop is **used for iterating over a sequence and iterables** (like range, list, a tuple, a dictionary, a set, or a string).

```
for i in range(5):
    statement 1
    statement 2
    ...
    statement n
else:
    statement(s)
```

Definite iterations. (Total 5 iterations)

Indentation: Loop body is must be properly indented

Body of for loop: Execute till the last item of a sequence

Else Block (optional): Execute only when for loop executes normally

Python While loop

While loops **repeat the same code as long as a certain condition is true**

```
while Condition:
    statement 1
    statement 2
    ...
    statement n
else:
    statement(s)
```

Expression that returns True or False

Indentation: Loop body is must be properly indented

Body of while loop: Execute as long as condition is True

Else Block (optional): Execute only when while loop executes normally

Loop Control Statements

Statement	Description
`break`	Terminate the current loop. Use the break statement to come out of the loop instantly.
`continue`	Skip the current iteration of a loop and move to the next iteration
`pass`	Do nothing. Ignore the condition in which it occurred and proceed to run the program as usual

Python Number Methods

Method	Description
`ceil()`	To round the number UP to the nearest integer value
`floor()`	To round a number DOWN to the nearest integer value.
`degrees()`	To convert the angle in radians to degrees.
`radians()`	To convert the angle in degrees to radians.
`factorial()`	To find the factorial of any positive integer
`fabs()`	To find a number's absolute value as a float.
`trunc()`	To truncate a number to its nearest integer
`pow()`	Accepts two arguments, x, and y, and returns the value of x raised to power y
`isfinite()` and `isinf()`	To find whether a number is finite or not
`isclose()`	To find whether two numbers are close or not using
`getcontext()`	To set the precision of a decimal and set the flags to mention to round the digits up or Down.
`limit_denominator()`	To limit the number of digits in the denominator.
`from_float()`	To create fractions from a floating number

List Operations

Operation	Description
`x in l1`	Check if the list `l1` contains item `x`.
`x not in l2`	Check if list `l1` does not contain item `x`.
`l1 + l2`	Concatenate the lists `l1` and `l2`. Creates a new list containing the items from `l1` and `l2`.
`l1 * 5`	Repeat the list `l1` 5 times.
`l1[i]`	Get the item at index `i`. Example `l1[2]` is 30.
`l1[i:j]`	List slicing. Get the items from index `i` up to index `j` (excluding `j`) as a List. An example `l1[0:2]` is `[10, 20]`
`l1[i:j:k]`	List slicing with step. Returns a List with the items from index `i` up to index j taking every k-th item. An example `l1[0:4:2]` is [10, 30].
`len(l1)`	Returns a count of total items in a list.
`l2.count(60)`	Returns the number of times a particular item (60) appears in a list. The answer is 2.
`l1.index(30)`	Returns the index number of a particular item (30) in a list. The answer is 2.
`l1.index(30, 2, 5)`	Returns the index number of a particular item (30) in a list. But search Returns the item with maximum value from a list. The answer is 60 only from index number 2 to 5.
`min(l1)`	Returns the item with a minimum value from a list. The answer is 10.
`max(l1)`	Returns the item with maximum value from a list. The answer is 60.
`l1.append(100)`	Add item at the end of the list
`l1.append([2, 5, 7])`	Append the nested list at the end
`l1[2] = 40`	Modify the item present at index 2
`l1.remove(40)`	Removes the first occurrence of item 40 from the list.
`pop(2)`	Removes and returns the item at index 2 from the list.
`l1.clear()`	Make list empty
`l3= l1.copy()`	Copy `l1` into `l2`

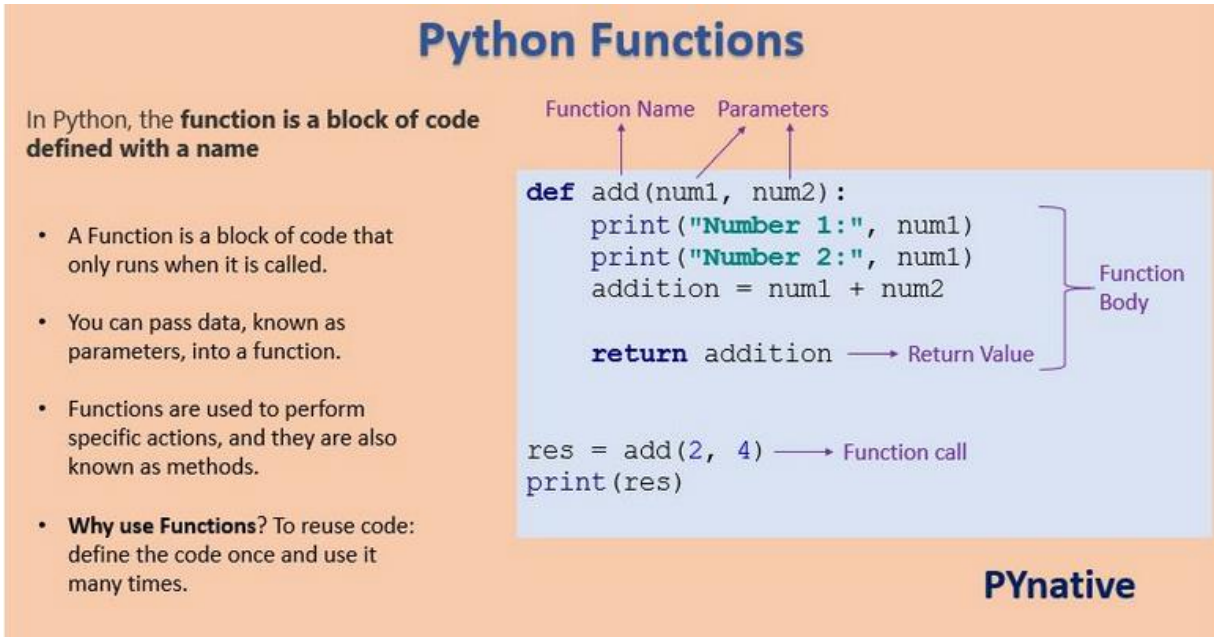
Python Set Operations

Operation	Definition	Operator	Method
Union	All the items of both Sets will be returned. Only the duplicate items will be dropped.	`|`	`union()`
Intersection	Only the items common in both sets will be returned.	`&`	`intersection()`
Difference	Return the unique elements in the first set which is not in the second set.	`-`	`difference()`
Symmetric Difference	Return the elements of both sets which is not common.	`^`	`symmetric_difference()`

Python Dictionary Operations

Operations	Description
`dict({'a': 10, 'b': 20})`	Create a dictionary using a `dict()` constructor.
`d2 = {}`	Create an empty dictionary.
`d1.get('a')`	Retrieve value using the key name `a`.
`d1.keys()`	Returns a list of keys present in the dictionary.
`d1.values()`	Returns a list with all the values in the dictionary.
`d1.items()`	Returns a list of all the items in the dictionary with each key-value pair inside a tuple.
`len(d1)`	Returns number of items in a dictionary.
`d1['d'] = 40`	Update dictionary by adding a new key.
`d1.update({'e': 50, 'f': 60})`	Add multiple keys to the dictionary.
`d1.setdefault('g', 70)`	Set the default value if a key doesn't exist.
`d1['b'] = 100`	Modify the values of the existing key.
`d1.pop('b')`	Remove the key `b` from the dictionary.
`d1.popitem()`	Remove any random item from a dictionary.
`d1.clear()`	Removes all items from the dictionary.
`'key' in d1.keys()`	Check if a `key` exists in a dictionary.
`d1.update(d2)`	Add all items of dictionary `d2` into `d1`.
`d3= {**d1, **d2}`	Join two dictionaries.
`d2 = d1.copy()`	Copy dictionary `d1` into `d2`.
`max(d1)`	Returns the key with the maximum value in the dictionary `d1`
`min(d1)`	Returns the key with the minimum value in the dictionary `d1`

Python Functions



For further information and explanation see pynative.com