**Range:**

The range () function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Ex. for in range (1, 10):

Print (i)

In **2.x** we have range and x-range, range allocates each memory location for each values where x-range will have only one object

In **3.x** we have only range and it gives only one object

**Type Conversion:**

Converting one data type to other datatype.

Ex: a=10.4

int (a)

**Note:**

Any datatype to int and dict cannot be done

Datatypes of Ordered and Unordered

1. Ordered(Immutable)
   1. Int()
   2. Float()
   3. Complex()
   4. Bool()
   5. Str()
   6. Type()
2. Unordered(Mutable)
   1. Set()
   2. Dict()
   3. List()

Operators:

Operators is nothing but which performs specific task or operations on two operands.

(or)

Operators are used to perform operations on variables and values.

Ex. a + b

Unary operator: Ex. a+=1

Binary operator: Ex. a+b

Ternary operator: Ex. c=a+b

**Types:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Try it** |
| + | Addition | x + y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_add) |
| - | Subtraction | x - y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_sub) |
| \* | Multiplication | x \* y | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_mult) |
| / | Division | x / y Ex: 10/20=0.5(Gives floating value also) | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_div) |
| % | Modulus | x % y Ex: 10%4=2(Reminder) | [Try it »](https://www.w3schools.com/python/showpython.asp?filename=demo_oper_mod) |
| \*\* | Exponentiation | x \*\* y Ex: 2\*\*5=32 |  |
| // | Floor division | x // y Ex: 10/20=0(Takes only integer) |  |

1. **Arithmetic operators :**

**2. Assignment operators:**

* Comparison operators
* Logical operators

**Identity operators:** It checks whether the memory address is same

* + 1. **Is**
    2. **Is not**

**Membership operators:**

* + 1. **In**
    2. **Not in**

**Bitwise operators:**

There are six types.

They are

1. Bitwise Complementary(~):

It represents negative values

Ex: a=10

0000 1010

~ is 1111 0101

2. Bitwise and

3. Bitwise or

4. Bitwise X-OR

5. Left Shift

Ex: 00001010 x<<2

Ans: 00101000

6. Right Shift

Ex: 00001010 x>>2

Ans: 00000010

**Operator Precedency:**

It is nothing but which operator is first evaluated.

Order is

* 1. ()
  2. \*\*
  3. -,~
  4. \*,/,%,||
  5. +,-

|  |  |
| --- | --- |
| **Operator** | **Description** |
| () | Parenthesis |
| \*\* | Exponentiation (raise to the power) |
| ~ + - | Complement, unary plus and minus (method names for the last two are +@ and -@) |
| \* / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND'td> |
| ^ | | Bitwise exclusive `OR' and regular `OR' |
| <= < > >= | Comparison operators |
| <> == != | Equality operators |
| = %= /= //= -= += \*= \*\*= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |

**Control Flow Statements:**

A program’s control flow is the order in which the program’s code executes.

1. Decision making
2. Looping
3. Breaking
4. **Decision Making:**

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions

1. If
2. If…..else
3. If…elif…..else
4. Nested if

|  |  |
| --- | --- |
| **Sr.No.** | **Statement & Description** |
| 1 | [**if statements**](https://www.tutorialspoint.com/python/python_if_statement.htm)  An **if statement** consists of a boolean expression followed by one or more statements. |
| 2 | [**if...else statements**](https://www.tutorialspoint.com/python/python_if_else.htm)  An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE. |
| 3 | [**if...elif……..else statements**](https://www.tutorialspoint.com/python/python_if_else.htm)  An **if statement** is failed then it checks elif…..even elif is failed , then else will be printed. |
| 4 | [**nested if statements**](https://www.tutorialspoint.com/python/nested_if_statements_in_python.htm)  You can use one **if** or **else if** statement inside another **if** or **else if**statement(s). |

**Suite:**

Statements inside condition is called suite.

Ex: if (a>b):

Statement 1,

|  |
| --- |
| Suite |

Statement 2,

Statement 3

1. [**nested if statements**](https://www.tutorialspoint.com/python/nested_if_statements_in_python.htm):

x=10  
if x>0:  
 if x%2==0:  
 print("Given Number is Even Number")  
else:  
 print("Given Number is Even Number")

**Looping:**

1. While loop
2. For loop
3. **While Loop:**

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

x=0  
while x<10:  
 print(x)  
 x=x+1

x=0  
while x<10:  
 print(x)  
 # x = x + 1  
 break  
print("a")

1. **For loop:**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

Assignment:

1. Program for number and string palindrome using while loop

n=int(input("Enter number:"))  
temp=n  
rev=0  
tot=0  
while(n>0):  
 dig=n%10  
 rev=rev\*10+dig  
 n=n//10  
 tot=tot+dig  
if(temp==rev):  
 print("The number is a palindrome!")  
else:  
 print("The number isn't a palindrome!")  
  
 print("The number isn't a palindrome!")  
print("The reverse num is:", rev) #to get reverse of a number  
print("The sum of digits:", tot) # to print sum of digits

1. Fibonacci series

a,b=0,1  
print(a)  
while b>0:  
 print(b)  
 a,b=b,a+b  
 if a>10:  
 break

1. Prime Numbers Ex: range(100,200)

start = 1  
End = 100  
for num in range(start, End + 1):  
 if num > 1:  
 for i in range(2, num):  
 if num % i==0:  
 break  
 else:  
 print(num)

start\_num = int(input("Enter starting number:"))  
End\_num = int(input("Enter ending number:"))  
for num in range(start\_num, End\_num + 1):  
 if num > 1:  
 for i in range(2, num):  
 if num % i==0:  
 break  
 else:  
 print(num)

1. **How to use else block in for loop**

**If for loop is executed without any break or abnormal termination else block will be executed successfully.**

for i in range(5):  
 print(i)  
else:  
 print("executed")

**The abnormal termination has happened……so else block will not be executed**

for i in range(5):  
 print(i)  
 if i==4:  
 break  
else:  
 print("executed")

1. **If we give search ……it should tell at which index its present…..if it is not there it should tell not there in the list**

cities=['hyd','chennai','bangalore','calicut']  
index=0  
search\_city=input("Enter the city name: ")  
for city in cities:  
 if city == search\_city:  
 print(city,"city found at index",index)  
 break  
 index=index+1  
else:  
 print("City is not found")

**For Loop:**

It is an iterating method

a = "Welcome Mr. Mohan Reddy P"  
for range in a:  
 print(range)

**Nested Loop:**

Loop inside loop

Ex: Print 5th Table

for i in range(5,6):  
 for j in range(1,11):  
 print("%d\*%d=%d" %(i,j,i\*j))

1. Print \*

for i in range(10):  
 for j in range(1,i+1):  
 print("\*",end="")  
 print() # to get \n (new line)

1. **Break:**

Break is used to break from the loop based on condition.

for i in range(8):  
 if i==4:  
 break  
 print(i)

1. **Continue:**

It skips the current iteration and goes to next iteration

for i in range(8):  
 if i==4:  
 continue  
 print(i)

**Pass:**

The pass keyword is used as a placeholder. Pass is a null statement.

If you have an empty function definition, like in the example above, you would get an error without the Pass statement.

The pass keyword can also be used in empty class definitions.

def test():  
 pass  
test()

**Assert:**

The assert keyword is used when debugging code.

The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an Assertion Error.

You can write a message to be written if the code returns False, check the example below.

x=-10  
assert x>0,"Wrong input"  
print(x)

**Return:**

Used to return values from function

def add(a,b):  
 return a+b  
c=add(10,20)  
print(c)

def test(a,b,c):  
 return a,b,c  
a,b,c=test(10,20,30)  
print("a =",a,"b =",b,"c =",c)

Assignment:

1. Print \* in Pyramid shape
2. L=[10,10,20,20,30]