Python is a widely used high-level, interpreted programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

Key Features of Python
Easy to Learn and Use: Python's simple and readable syntax makes it beginner-friendly.
Cross-Platform Compatibility: Python runs seamlessly on Windows, macOS, and Linux.
Extensive Libraries: Includes robust libraries for tasks like web development, data analysis, and machine learning.
Dynamic Typing: Variable types are determined automatically at runtime, simplifying code writing.
Versatile: Supports multiple programming paradigms, including object-oriented, functional, and procedural programming.
Open Source: Python is free to use, distribute, and modify.
Python is a case-sensitive language

Why Learn Python?
Whether you are a beginner or an experienced developer, both have their own benefits.

For Beginners:
Easy Syntax: Python syntax is like plain English, which allows you to focus on logic instead of worrying about complex rules.
Built-in Libraries for Beginners: Python has beginner friendly libraries like random, re, os etc, which can be used while learning fundamentals.
Error Friendly: Python's error messages are easy to understand and debug.
Project Oriented Learning: You can start making simple projects while learning the Python basics.

For Experienced:
Easy Career Transition: If you know any other programming language, moving to Python is super easy.
Great for Upskilling: Moving to Python expands your skill sets and gives opportunity to work in areas like AI, Data Science, web development etc.
High Demand of Python in Emerging tech: Python is widely used in trending domains, like Data Science, Machine Learning, Cloud Computing etc.
Bridge Between Roles: For software developers working with different language, learning Python can help you integrate advanced features like AI in your projects.

In [3]:
```
print("Hello, World!")
```

```
Hello, World!
```

How does this work:

print() is a built-in function in Python that tells the program to display something on the screen. We need to add the string in parenthesis of print() function that we are displaying on the screen.
“Hello, World!” is a string text that we want to display. Strings are always enclosed in quotation marks.

Python Comments:

Comments in Python are the lines in the code that are ignored by the interpreter during the execution of the program.
Comments enhance the readability of the code.
Comment can be used to identify functionality or structure the code-base.
Comment can help understanding unusual or tricky scenarios handled by the code to prevent accidental removal or changes.
Comments can be used to prevent executing any specific part of your code, while making changes or testing.
Example:

```
#I am single line comment

""" Multi-line comment used
print("Python Comments") """
```

Explanation:

In Python, single line comments starts with hashtag symbol #.
Python ignores the string literals that are not assigned to a variable. So, we can use these string literals as Python Comments.

Indentation in Python:

In Python, indentation is used to define blocks of code. It tells the Python interpreter that a group of statements belongs to a specific block. All statements with the same level of indentation are considered part of the same block. Indentation is achieved using whitespace (spaces or tabs) at the beginning of each line.

Famous Application Built using Python
YouTube: World’s largest video-sharing platform uses Python for features like video streaming and backend services.
Instagram: This popular social media app relies on Python’s simplicity for scaling and handling millions of users.
Spotify: Python is used for backend services and machine learning to personalize music recommendations.
Dropbox: The file hosting service uses Python for both its desktop client and server-side operations.
Netflix: Python powers key components of Netflix’s recommendation engine and content delivery systems (CDN).
Google: Python is one of the key languages used in Google for web crawling, testing, and data analysis.
Uber: Python helps Uber handle dynamic pricing and route optimization using machine learning.
Pinterest: Python is used to process and store huge amounts of image data efficiently.

Taking input in Python Python input() function is used to take user input. By default, it returns the user input in form of a string.

Printing Output using print() in Python At its core, printing output in Python is straightforward, thanks to the print() function. This function allows us to display text, variables and expressions on the console. Let's begin with the basic usage of the print() function:

In [4]:
```
name = input("Enter your name: ")

print("Hello,", name, "! Welcome!")
```

```
Enter your name: Tanmay
Hello, Tanmay ! Welcome!
```

In [5]:
```
# Single variable
s = "Bob"
print(s)

# Multiple Variables
s = "Alice"
age = 25
city = "New York"
print(s, age, city)
```

```
Bob
Alice 25 New York
```

Take Multiple Input in Python : We are taking multiple input from the user in a single line, splitting the values entered by the user into separate variables for each value using the split() method. Then, it prints the values with corresponding labels, either two or three, based on the number of inputs provided by the user.

In [6]:
```
# taking two inputs at a time
x, y = input("Enter two values: ").split()
print("Number of boys: ", x)
print("Number of girls: ", y)

# taking three inputs at a time
x, y, z = input("Enter three values: ").split()
print("Total number of students: ", x)
print("Number of boys is : ", y)
print("Number of girls is : ", z)
```

```
Enter two values: 10 20
Number of boys:  10
Number of girls:  20
Enter three values: 10 20 30
Total number of students:  10
Number of boys is :  20
Number of girls is :  30
```

In [7]:
```python
#Take Conditional Input from user in Python

# Prompting the user for input
age_input = input("Enter your age: ")

# Converting the input to an integer
age = int(age_input)

# Checking conditions based on user input
if age < 0:
    print("Please enter a valid age.")
elif age < 18:
    print("You are a minor.")
elif age >= 18 and age < 65:
    print("You are an adult.")
else:
    print("You are a senior citizen.")
```

```
Enter your age: 10
You are a minor.
```

In [8]:
```python
# Taking input as int
# Typecasting to int
n = int(input("How many roses?: "))
print(n)
```

```
How many roses?: 3
3
```

In [9]:
```python
# Taking input as float
# Typecasting to float
price = float(input("Price of each rose?: "))
print(price)
```

```
Price of each rose?: 45.5
45.5
```

In [10]:
```
#Find DataType of Input in Python

a = "Hello World"
b = 10
c = 11.22
d = ("Geeks", "for", "Geeks")
e = ["Geeks", "for", "Geeks"]
f = {"Geeks": 1, "for":2, "Geeks":3}


print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
print(type(f))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'tuple'>
<class 'list'>
<class 'dict'>
```

```
Output Formatting
Output formatting in Python with various techniques including the format()
method, manipulation of the sep and end parameters, f-strings and the versatile
% operator. These methods enable precise control over how data is displayed,
enhancing the readability and effectiveness of your Python programs.
```

In [11]:
```
amount = 150.75
print("Amount: ${:.2f}".format(amount))
```

```
Amount: $150.75
```

In [12]:
```
# end Parameter with '@'
print("Python", end='@')
print("GeeksforGeeks")


# Seprating with Comma
print('G', 'F', 'G', sep='')

# for formatting a date
print('09', '12', '2016', sep='-')

# another example
print('pratik', 'geeksforgeeks', sep='@')
```

```
Python@GeeksforGeeks
GFG
09-12-2016
pratik@geeksforgeeks
```

In [13]:
```
name = 'Tushar'
age = 23
print(f"Hello, My name is {name} and I'm {age} years old.")
```

```
Hello, My name is Tushar and I'm 23 years old.
```

```
Using % Operator :
We can use ‘%’ operator. % values are replaced with zero or more value of
elements. The formatting using % is similar to that of ‘printf’ in the C
programming language.
%d –integer
%f – float
%s – string
%x –hexadecimal
%o – octal
```

In [14]:
```
# Taking input from the user
num = int(input("Enter a value: "))

add = num + 5

# Output
print("The sum is %d" %add)
```

```
Enter a value: 5
The sum is 10
```

In [16]:
```
# In Python 2, there are two functions to take user input: `input()` and `raw_inp
# In Python 3, `input()` is the only available function (equivalent to `raw_input

# Python 2:
# input() - Evaluates the user input as a Python expression.
# raw_input() - Takes input as a string (safe to use).

# Example in Python 2:
# num = input("Enter a number: ")   # If user enters 5, it is treated as an integ
# text = raw_input("Enter a text: ")  # If user enters 5, it is treated as a stri

# Python 3:
# input() behaves like raw_input() in Python 2, always returning a string.
# If needed, we explicitly convert it using int(), float(), etc.

# Example in Python 3:
# num = int(input("Enter a number: "))  # Converts input string to an integer.
# text = input("Enter a text: ")  # Stores input as a string.

# Summary:
# Python 2:
# - input() -> Evaluates the input as a Python expression (unsafe if used careles
# - raw_input() -> Always returns a string (safe to use).

# Python 3:
# - input() -> Equivalent to raw_input() in Python 2 (always returns a string).
# - raw_input() is removed in Python 3.
```

```
A variable is essentially a name that is assigned to a value. Unlike many other
programming languages, Python variables do not require explicit declaration of
type. The type of the variable is inferred based on the value assigned.

Rules for Naming Variables :
To use variables effectively, we must follow Python’s naming rules:
Variable names can only contain letters, digits and underscores (_).
A variable name cannot start with a digit.
Variable names are case-sensitive (myVar and myvar are different).
Avoid using Python keywords (e.g., if, else, for) as variable names.
```

```
Valid Example:
age = 21
_colour = "lilac"
total_score = 90


Invalid Example:
1name = "Error"  # Starts with a digit
class = 10       # 'class' is a reserved keyword
user-name = "Doe"  # Contains a hyphen
```

In [18]:
```
#Multiple assignment
a = b = c = 100
print(a, b, c)

x, y, z = 1, 2.5, "Python"
print(x, y, z)
```

```
100 100 100
1 2.5 Python
```

In [19]:
```
# Casting variables
s = "10"  # Initially a string
n = int(s)  # Cast string to integer
cnt = 5
f = float(cnt)  # Cast integer to float
age = 25
s2 = str(age)  # Cast integer to string

# Display results
print(n)
print(f)
print(s2)
```

```
10
5.0
25
```

In [20]:
```
# Define variables with different data types
n = 42
f = 3.14
s = "Hello, World!"
li = [1, 2, 3]
d = {'key': 'value'}
bool = True

# Get and print the type of each variable
print(type(n))
print(type(f))
print(type(s))
print(type(li))
print(type(d))
print(type(bool))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'list'>
<class 'dict'>
<class 'bool'>
```

In [25]:
```
#Variables defined inside a function are local to that function.
def f():
    loc = "I am local"
    print(loc)

f()
print(loc)
# print(a)  # This would raise an error since 'local_var' is not accessible outsi
```

```
I am local

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11916\3744718481.py in <module>
      5
      6 f()
----> 7 print(loc)
      8 # print(a)  # This would raise an error since 'local_var' is not access
ible outside the function

NameError: name 'loc' is not defined
```

In [26]:
```
#Variables defined outside any function are global and can be accessed inside fur
a = "I am global"

def f():
    global a
    a = "Modified globally"
    print(a)

f()
print(a)
```

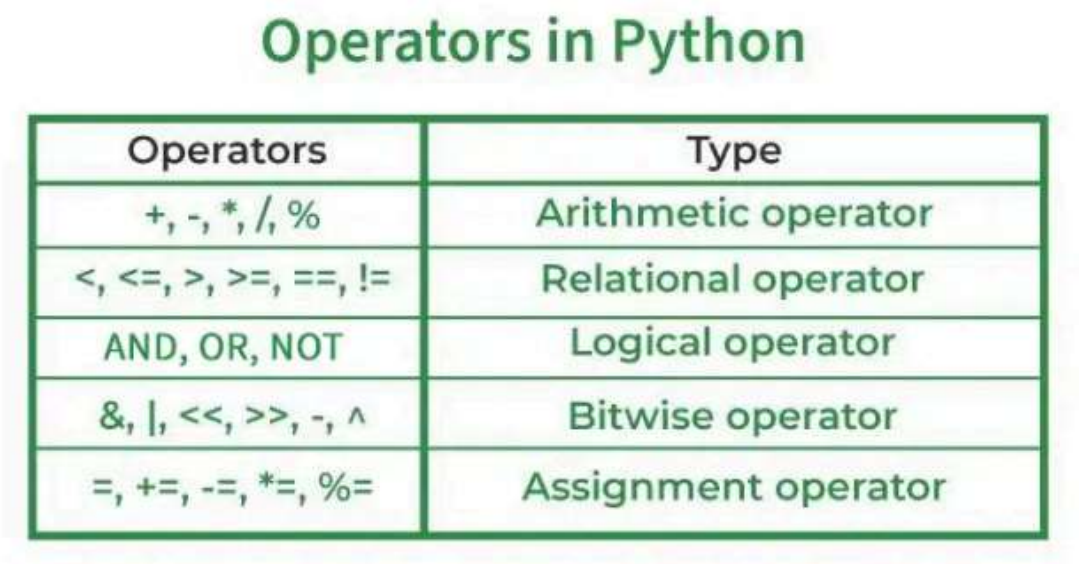
```
Modified globally
Modified globally
```

In [28]:
```
#We can remove a variable from the namespace using the del keyword.
#This effectively deletes the variable and frees up the memory it was using.

# Assigning value to variable
x = 10
print(x)

# Removing the variable using del
del x

# Trying to print x after deletion will raise an error
print(x)  # Uncommenting this line will raise NameError: name 'x' is not defined
```

```
10

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11916\3851667791.py in <module>
     10
     11 # Trying to print x after deletion will raise an error
---> 12 print(x)  # Uncommenting this line will raise NameError: name 'x' is no
t defined

NameError: name 'x' is not defined
```

In [30]:
```
#Swapping Two Variables
a, b = 5, 10
a, b = b, a
print(a, b)

#Counting Characters in a String
word = "Python"
length = len(word)
print("Length of the word:", length)
```

```
10 5
Length of the word: 6
```



In [32]:
```
# Variables
a = 15
b = 4

# Division
print("Division:", a / b)

# Floor Division
print("Floor Division:", a // b)

# `/` (Single Slash) - Performs **floating-point division** (returns a decimal re
# `//` (Double Slash) - Performs **floor division** (rounds down to the nearest w
```

```
Division: 3.75
Floor Division: 3
```

In [34]:
```
#is          True if the operands are identical
#is not      True if the operands are not identical

#Example of Identity Operators in Python:

a = 10
b = 20
c = a

print(a is not b)
print(a is c)




#in            True if value is found in the sequence
#not in        True if value is not found in the sequence

#Examples of Membership Operators in Python:

x = 24
y = 20
list = [10, 20, 30, 40, 50]

if (x not in list):
    print("x is NOT present in given list")
else:
    print("x is present in given list")

if (y in list):
    print("y is present in given list")
else:
    print("y is NOT present in given list")
```

```
True
True
x is NOT present in given list
y is present in given list
```

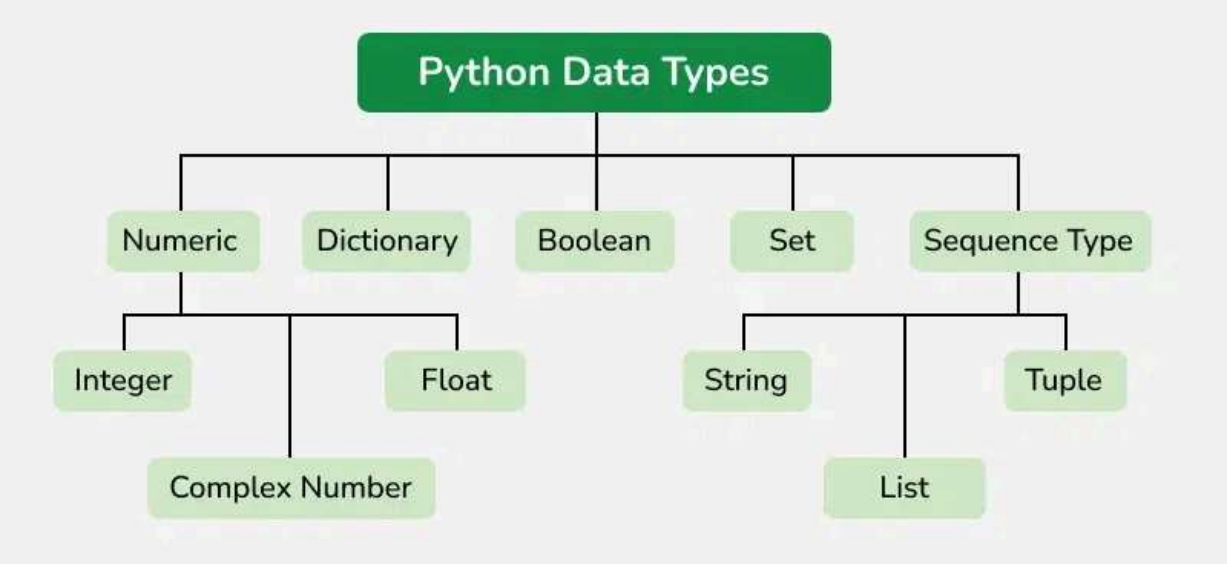
In [35]:
```
#Ternary Operator in Python:
a, b = 10, 20
min = a if a < b else b

print(min)
```

```
10
```

In [36]:
```
import keyword

# printing all keywords at once using "kwlist()"
print("The list of keywords is : ")
print(keyword.kwlist)
```

```
The list of keywords is :
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'aw
ait', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'fi
nally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonloca
l', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

```
Datatype :

Numeric - int, float, complex
Sequence Type - string, list, tuple
Mapping Type - dict
Boolean - bool
Set Type - set, frozenset
Binary Types - bytes, bytearray, memoryview
```



In [37]:
```
a = 5
print(type(a))

b = 5.0
print(type(b))

c = 2 + 4j
print(type(c))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

In [39]:
```
#String Data Type
#Python Strings are arrays of bytes representing Unicode characters. In Python, t
#a character is a string of length one. It is represented by str class.

s = 'Welcome to the Geeks World'
print(s)

# check data type
print(type(s))

# access string with index
print(s[0])
print(s[2])
print(s[-1])
```

```
Welcome to the Geeks World
<class 'str'>
W
l
d
```

In [42]:
```
#List Data Type
#Lists are just like arrays, declared in other languages which is an ordered coll
#It is very flexible as the items in a list do not need to be of the same type.

# Empty list
a = []

# list with int values
a = [1, 2, 3]
print(a)

# list with mixed int and string
b = ["Geeks", "For", "Geeks", 4, 5]
print(b)


a = ["Geeks", "For", "Geeks"]
print("Accessing element from the list")
print(a[0])
print(a[2])

print("Accessing element using negative indexing")
print(a[-1])
print(a[-3])

a[0]="Hi"
print("Modified value :",a[0])
```

```
[1, 2, 3]
['Geeks', 'For', 'Geeks', 4, 5]
Accessing element from the list
Geeks
Geeks
Accessing element using negative indexing
Geeks
Geeks
Modified value : Hi
```

In [45]:
```
#Tuple Data Type
#Just like a list, a tuple is also an ordered collection of Python objects.
#The only difference between a tuple and a list is that tuples are immutable. Tup

tup1 = tuple([1, 2, 3, 4, 5])

# access tuple items
print(tup1[0])
print(tup1[-1])
print(tup1[-3])

tup1[0]=5
print(tup1[0])
```

```
1
5
3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_11916\2369860807.py in <module>
     10 print(tup1[-3])
     11
---> 12 tup1[0]=5
     13 print(tup1[0])

TypeError: 'tuple' object does not support item assignment
```

In [46]:
```
#Set Data Type in Python
#In Python Data Types, Set is an unordered collection of data types that is itera
#The order of elements in a set is undefined though it may consist of various ele

# initializing empty set
s1 = set()

s1 = set("GeeksForGeeks")
print("Set with the use of String: ", s1)

s2 = set(["Geeks", "For", "Geeks"])
print("Set with the use of List: ", s2)
```

```
Set with the use of String:  {'r', 'e', 's', 'k', 'o', 'F', 'G'}
Set with the use of List:  {'For', 'Geeks'}
```

In [47]:
```
#Dictionary Data Type
#A dictionary in Python is a collection of data values, used to store data values
#unlike other Python Data Types that hold only a single value as an element, a Di
#Key-value is provided in the dictionary to make it more optimized. Each key-valu
#by a colon : , whereas each key is separated by a ‘comma’.

# initialize empty dictionary
d = {}


d = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print(d)

# creating dictionary using dict() constructor
d1 = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})
print(d1)
```

```
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
```

In [48]:
```
d = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# Accessing an element using key
print(d['name'])

# Accessing a element using get
print(d.get(3))
```

```
For
Geeks
```

In [49]:
```
fruits = ["apple", "banana", "orange"]
print(fruits)
fruits.append("grape")
print(fruits)
fruits.remove("orange")
print(fruits)
```

```
['apple', 'banana', 'orange']
['apple', 'banana', 'orange', 'grape']
['apple', 'banana', 'grape']
```

Feature	List	Tuple	Set	Dictionary
Mutable	✅ Yes	❌ No	✅ Yes (Add/Remove)	✅ Yes (Keys & Values)
Ordered	✅ Yes (Indexed)	✅ Yes (Indexed)	❌ No (Unordered)	✅ Yes (Python 3.7+)
Duplicates Allowed	✅ Yes	✅ Yes	❌ No	❌ No (Keys must be unique)
Indexing	✅ Yes	✅ Yes	❌ No (Unordered)	✅ Yes (Keys act as index)
Definition	`[]` (Square Brackets)	`()` (Parentheses)	`{}` or `set()`	`{key: value}` (Curly Braces)
Use Case	Ordered, modifiable collection	Ordered, immutable collection	Unique elements, fast lookup	Key-value pairs, fast lookup

In [50]:
```
#conditional Statement

age = 25

if age <= 12:
    print("Child.")
elif age <= 19:
    print("Teenager.")
elif age <= 35:
    print("Young adult.")
else:
    print("Adult.")
```

```
Young adult.
```

In [51]:
```
age = 70
is_member = True

if age >= 60:
    if is_member:
        print("30% senior discount!")
    else:
        print("20% senior discount.")
else:
    print("Not eligible for a senior discount.")
```

```
30% senior discount!
```

In [52]:
```
#Ternary Operator
# Assign a value based on a condition
age = 20
s = "Adult" if age >= 18 else "Minor"

print(s)
```

```
Adult
```

In [ ]:
```
def check_number(x):
    match x:
        case 10:
            print("It's 10")
        case 20:
            print("It's 20")
        case _:
            print("It's neither 10 nor 20")

check_number(10)
check_number(30)
```

In [59]:
```
#Loops in Python

#While Loop in Python
#In Python, a while loop is used to execute a block of statements repeatedly unti
#When the condition becomes false, the line immediately after the loop in the pro

cnt = 0
while (cnt < 3):
    cnt = cnt + 1
    print("Hello Geek")
```

```
Hello Geek
Hello Geek
Hello Geek
In Else Block
```

In [60]:
```
cnt = 0
while (cnt < 3):
    cnt = cnt + 1
    print("Hello Geek")
else:
    print("In Else Block")
```

```
Hello Geek
Hello Geek
Hello Geek
In Else Block
```

In [62]:
```
#infinite loop
"""count = 0
while (count == 0):
    print("Hello Geek")"""
```

Out[62]:
```
'count = 0\nwhile (count == 0):\n    print("Hello Geek")'
```

In [63]:
```
#For Loop in Python
#For loops are used for sequential traversal. For example: traversing a list or s
#In Python, there is “for in” loop which is similar to foreach loop in other lang

n = 4
for i in range(0, n):
    print(i)

    #for loop that iterates over a range from 0 to n-1 (where n = 4).
```

```
0
1
2
3
```

In [64]:
```
li = ["geeks", "for", "geeks"]
for i in li:
    print(i)

tup = ("geeks", "for", "geeks")
for i in tup:
    print(i)

s = "Geeks"
for i in s:
    print(i)

d = dict({'x':123, 'y':354})
for i in d:
    print("%s  %d" % (i, d[i]))

set1 = {1, 2, 3, 4, 5, 6}
for i in set1:
    print(i),
```

```
geeks
for
geeks
geeks
for
geeks
G
e
e
k
s
x  123
y  354
1
2
3
4
5
6
```

In [65]:
```
list = ["geeks", "for", "geeks"]
for index in range(len(list)):
    print(list[index])
```

```
geeks
for
geeks
```

In [66]:
```
list = ["geeks", "for", "geeks"]
for index in range(len(list)):
    print(list[index])
else:
    print("Inside Else Block")
```

```
geeks
for
geeks
Inside Else Block
```

In [67]:
```
for i in range(1, 5):
    for j in range(i):
        print(i, end=' ')
    print()
```

```
1
2 2
3 3 3
4 4 4 4
```

In [68]:
```
#Continue Statement
#The continue statement in Python returns the control to the beginning of the loo

for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
        continue
    print('Current Letter :', letter)
```

```
Current Letter : g
Current Letter : k
Current Letter : f
Current Letter : o
Current Letter : r
Current Letter : g
Current Letter : k
```

In [69]:
```
#Break Statement
#The break statement in Python brings control out of the loop.

for letter in 'geeksforgeeks':
    if letter == 'e' or letter == 's':
        break

print('Current Letter :', letter)
```

```
Current Letter : e
```

In [71]:
```
#Pass Statement
#We use pass statement in Python to write empty loops. Pass is also used for empt

for letter in 'geeksforgeek':
    pass
print('Last Letter :', letter)
```

```
Last Letter : k
```

In [72]:
```
#This Python code manually iterates through a list of fruits using an iterator.
#It prints each fruit’s name one by one and stops when there are no more items ir

fruits = ["apple", "orange", "kiwi"]
iter_obj = iter(fruits)
while True:
    try:
        fruit = next(iter_obj)
        print(fruit)
    except StopIteration:
        break
```

```
apple
orange
kiwi
```

In [ ]:
```

```