**Python**

**# Data Types #**

*# 1 - Number Data Type*x = 10 *# Int*x = 10.2 *# Float*x = 10 > 3 *# Bool*x = 10j *# Complex  
  
# 2 - String Data Type  
# String define in " " OR '*

*# Strings in Python can be created using single quotes or double quotes or even triple quotes.  
# Individual characters of a String can be accessed by using the method of Indexing.  
# In Python, Updation or deletion of characters from a String is not allowed. This is because Strings are immutable, hence elements of a String cannot be changed once it has been assigned. Only new strings can be reassigned to the same name.'*x = **'Hello'** *# Str*x = **"Hello"** *# Str*print(x) *# Print string. Output -> Hello*print(len(x)) *# len(variable\_name) for length of string. Output -> 5*print(type(x)) *# type(variable\_name) for type of variable. Output -> <class 'str'>*print(x[2]) *# show string letter using index. Output -> l*print(x[2:4]) *# x[index\_start\_point : index\_End\_point].  
# x[-1] for last index in a string x[-2] for second last index of string.... so on  
# Output -> ll  
# x[2] = 'p' String cannot change. Output -> 'str' object does not support item assignment*print(x.upper()) *# string convert into Uppercase. Output -> HELLO*print(x.lower()) *# String convert into Lowercase. Output -> hello*print(x.find(**'wor'**)) *# Find the index position of letter or string in the variable. Output -> 6*

*# Formatting of String  
# Default order*String1 = **"{} {} {}"**.format(**'Geeks'**, **'For'**, **'Life'**)  
print(String1) *# Output -> Geeks For Life  
# Positional Formatting*String1 = **"{1} {0} {2}"**.format(**'Geeks'**, **'For'**, **'Life'**)  
print(String1) *# Output -> For Geeks Life  
# Keyword Formatting*String1 = **"{l} {f} {g}"**.format(g=**'Geeks'**, f=**'For'**, l=**'Life'**)  
print(String1) *# Output -> Life For Geeks*

*# String alignment*String1 = **"|{:<10}|{:^10}|{:>10}|"**.format(**'Geeks'**, **'for'**, **'Geeks'**)  
print(String1) *# Output -> |Geeks | for | Geeks|  
# To demonstrate aligning of spaces*String1 = **"**\n**{0:^16} was founded in {1:<4}! {2:>10}"**.format(**"GeeksforGeeks"**, 2009, 22222)  
print(String1) *# Output -> GeeksforGeeks was founded in 2009! 22222*

*# show alternate character*i = **'Hello world'**print(i[0:16:2]) *# Output -> Hlowrd*

*# Reverse string*xx = **"Hello World"***# After second colon number is negative then it return reverse string*print(xx[::-1]) *# Output -> dlrow olleH*print(xx[::-2]) *# Output -> drWolH   
# In above eg. first reverse string then return alternate character*

print(xx.endswith(**'ld'**)) *# Output -> True*print(xx.replace(**'World'**, **'Sandip'**)) *# Output -> Hello Sandip*

*# 3 - List Data Type  
# Ordered, can be changed. Duplicate entries are present*

*# A single list may contain DataTypes like Integers, Strings, as well as Objects.*

*# Lists are mutable, and hence, they can be altered even after their creation.*

*# append() method - only works for addition of elements at the end of the List.  
# insert() method - for addition of element at the desired position.  
# extend() method - add multiple elements at the same time at the end of the list.*

*# remove() method - Elements can be removed from the List by using built-in remove() function but an Error arises if element doesn’t exist in the set. Remove() method only removes one element at a time, to remove range of elements, iterator is used. The remove() method removes the specified item.*

*# pop method - Pop() function can also be used to remove and return an element from the set, but by default it removes only the last element of the set, to remove element from a specific position of the List, index of the element is passed as an argument to the pop() method.*

*# We can merge all Data Types in a list*x = [**'John'**, **'Sam'**, **'Andrew'**, **'John'**]  
x = [1, 3, 4, 6, 77, 3, 5, 6, **'John'**, **"Smith"**, {3, 6, 7}]  
x = [1, 3, 5, 6, 43j, {3, 56, 7, 7}]  
x.append(**'Sam'**) *# append for add the value in the last index*print(x) *# Output -> [1, 3, 5, 6, 43j, {56, 3, 7}, 'Sam']*x.insert(2, **"Sss"**) *# variable\_name.insert(Index, add\_value). Add the value on mention index in the list*print(x) *# Output -> [1, 3, 'Sss', 5, 6, 43j, {56, 3, 7}, 'Sam']*

x.extend([8, **'Geeks'**, **'Always'**])  
print(x) *# Output -> [1, 3, 'Sss', 5, 6, 43j, {56, 3, 7}, 'Sam', 8, 'Geeks', 'Always']*

x.remove(**'Sam'**)  
print(x) *# Output -> [1, 3, 'Sss', 5, 6, 43j, {56, 3, 7}, 8, 'Geeks', 'Always']*

x.pop(5)  
print(x) *# Output -> [1, 3, 'Sss', 5, 6, {56, 3, 7}, 'Sam', 8, 'Geeks', 'Always']*x.reverse() *# reverse the list*print(x) *# Output -> ['Sam', {56, 3, 7}, 43j, 6, 5, 'Sss', 3, 1]*

*# 4 - Dictionary Data Type  
# Unordered, can be changed. No duplicate entries are present.  
# In that Key Value pair. Key can not be duplicate and Value can be duplicate. Key and Value can be any Data Type*x = {  
 1: **'John'**,  
 2: **'Smith'**,  
 **'third'**: 232,  
 **'John'**: [53, 43]  
}  
print(x) *# Output -> {1: 'John', 2: 'Smith', 'third': 232, 'John': [53, 43]}*print(x[**'John'**]) *# We can access the value using Key. Output -> [53, 43]*x[**'third'**] = [233, 724] *# Update the value using Key*print(x) *# Output -> {1: 'John', 2: 'Smith', 'third': [233, 724], 'John': [53, 43]}  
  
  
# 5 - Tuple Data Type  
# Ordered , can not be changed. Duplicate entries are present.*x = (3, 6, 7, 3, 8, 3, 8, 356, 8, **'Smith'**)  
print(x.count(3)) *# count function show how many time value present in Tuple  
  
# 6 - Set Data Type  
# Unordered. No Duplicate entries are present.  
# Collection of List and Dictionary.*x = {4, 153, 83, 7, 2, 7, **'Sam'**, **'John'**}  
*# x[2] Set does not support indexing  
  
  
# ---- All Data Types ----*x = 10 *# Int*x = 10.2 *# Float*x = 10 > 3 *# Bool*x = 10j *# Complex*x = **'Hello'** *# Str*x = **"Hello"** *# Str*x = [1, 3, 4, 6, 77, 3, 5, 6, **'John'**, **"Smith"**, {3, 6, 7}] *# List*x = {  
 1: **'John'**,  
 2: **'Smith'**,  
 **'third'**: 232,  
 **'John'**: [53, 43]  
} *# Dictionary*x = (3, 6, 7, 3, 8, 3, 8, 356, 8, **'Smith'**) *# Tuple*x = {4, 153, 83, 7, 2, 7, **'Sam'**, **'John'**} *# Set*

**# Array #**

*# Python Arrays and lists have the same way of storing data.  
# Arrays take only a single data type elements but lists can have any type of data.  
# Therefore, other than a few operations, the kind of operations performed on them are different.  
# Array are muttable  
  
# Create an array in 3 ways  
# 1st method to create an Array*import array  
i = array.array(**'i'**,[2,4,6,7,8,3])  
print(i) *# Output -> array('i', [2, 4, 6, 7, 8, 3])*print(type(i)) *# Output -> <class 'array.array'>  
  
# 2nd method to create an Array*import array as arr  
i = arr.array(**'d'**,[2.0,4.3,3.2])  
print(i) *# Output -> array('d', [2.0, 4.3, 3.2])*print(type(i)) *# Output -> <class 'array.array'>  
  
# 3rd method to create an Array*from array import \*  
i = array(**'i'**, [2,6,3,7,4])  
print(i) *# Output -> array('i', [2, 6, 3, 7, 4])*print(type(i)) *# Output -> <class 'array.array'>  
  
# Accessing array elements*from array import \*  
i = array(**'i'**, [2,6,3,7,4])  
print(i[2]) *# Output -> 3  
  
# Array operation  
# 1 - Finding length of Array*import array as arr  
i = arr.array(**'d'**, [2.3,3.5,5.8,2.0])  
print(len(i)) *# Output -> 4  
  
# 2 - Adding element to an Array  
# append() -> Used when you want to add a single element at the end of an array  
# extend() -> Used when you want to add more than one element at the end of an array  
# insert() -> Used when you want to add an element at a specific position in an array*import array as arr  
i = arr.array(**'i'**, [2,4,5,7,6])  
i.append(33333)  
print(i) *# Output -> array('i', [2, 4, 5, 7, 6, 33333])*i.extend([9,5,333,6,42])  
print(i) *# Output -> array('i', [2, 4, 5, 7, 6, 33333, 9, 5, 333, 6, 42])*i.insert(3,431)  
print(i) *# Output -> array('i', [2, 4, 5, 431, 7, 6, 33333, 9, 5, 333, 6, 42])  
  
# 3 - Removing elements of an Array  
# pop() -> Used when you want to remove an element and return it  
# remove() -> Used when you want to remove an element with a specific value without returning it*import array as arr  
i = arr.array(**'i'**, [1,2,4,5,7,6])  
print(i.pop()) *# Output -> 6*print(i) *# Output -> array('i', [1, 2, 4, 5, 7])*print(i.pop(2)) *# Output -> 4*print(i) *# Output -> array('i', [1, 2, 5, 7])*i.remove(2) *# put the value which is you want to remove*print(i) *# Output -> array('i', [1, 5, 7])  
  
# 4 - Array Concatenation*import array as arr  
i = arr.array(**'i'**, [1,2,4,5,7,6])  
p = arr.array(**'i'**, [99,98,97,96])  
q = i + p  
print(q) *# Output -> array('i', [1, 2, 4, 5, 7, 6, 99, 98, 97, 96])  
  
  
# 5 - SLicing an Array*import array as arr  
i = arr.array(**'i'**, [1,2,4,5,7,6])  
print(i[0:4]) *# Output -> array('i', [1, 2, 4, 5])  
  
# 6 - For loop*import array as arr  
i = arr.array(**'i'**, [1,2,55,4,5,7,6])  
for x in i[1:3]:  
 print(x) *# Output -> 2 55. Or show all value then use i[1:3] replace with i  
  
# 7 - While loop*import array as arr  
i = arr.array(**'i'**, [1,2,55,4,5,7,6])  
b = 0 *# initialize the iterator*while b < i[1]: *# all value show using len(i)* print(i[b]) *# Output -> 1 2* b = b + 1 *# b += 1*

**# Hash Table and Hashmap #**

*# Hash table or a Hashmap is a type of data structure that maps keys to its value pairs  
# create dictionary in 2 method  
# Method 1*my\_dect = {**'John'**: **'001'**, **'Smith'**: **'002'**, **'Andrew'**: **'003'**}  
print(my\_dect) *# Output -> {'John': '001', 'Smith': '002', 'Andrew': '003'}*print(type(my\_dect)) *# Output -> <class 'dict'>  
  
# Method 2*new\_dects = dict(Smith=**'001'**,John=**'002'**)  
print(new\_dects) *# Output -> {'Smith': '001', 'John': '002'}*print(type(new\_dects)) *# Output -> <class 'dict'>  
  
# Nested Dictionary*emp\_details= {**'Employee'**: {**'John'**:{ **'ID'**: **'001'**, **'Salary'**: **'1000'**, **'Designation'**: **'TL'**},  
 **'Smith'**:{ **'ID'**: **'002'**, **'Salary'**: **'2000'**, **'Designation'**: **'Analyst'**},  
 }  
 }  
print(emp\_details)  
print(type(emp\_details)) *# Output -> <class 'dict'>  
  
# Operation on Hash table  
# Accessing values*print(my\_dect[**'John'**]) *# Output -> 001*print(my\_dect.keys()) *# Output -> dict\_keys(['John', 'Smith', 'Andrew'])*print(my\_dect.values()) *# Output -> dict\_values(['001', '002', '003'])*print(my\_dect.get(**'Smith'**)) *# Output -> 002*for x in my\_dect:  
 print(x) *# Output -> John Smith Andrew*for x in my\_dect.values():  
 print(x) *# Output -> 001 002 003*for x,y in my\_dect.items():  
 print(x,y) *# Output -> John 001 Smith 002 Andrew 003  
  
# Updating values*my\_dect[**'John'**] = **'004'**my\_dect[**'Kris'**] = **'005'**print(my\_dect) *# Output -> {'John': '004', 'Smith': '002', 'Andrew': '003', 'Kris': '005'}  
  
# Deleting*print(my\_dect.pop(**'Smith'**)) *# Output -> 002*print(my\_dect) *# Output -> {'John': '004', 'Andrew': '003', 'Kris': '005'}*print(my\_dect.popitem()) *# Output -> ('Kris', '005'). popitem() delete last key value pair*print(my\_dect) *# Output -> {'John': '004', 'Andrew': '003'}*del my\_dect[**'Andrew'**]  
print(my\_dect) *# Output -> {'John': '004'}  
  
# Converting Dictionary into a Dataframe  
# DataFrame is a 2-D data structure that consist of columns of various types.  
# It is very similar to a Python Dictionary and you can even convert into a pandas dataframe*import pandas as pd  
df = pd.DataFrame(emp\_details[**'Employee'**])  
print(df)  
*# Output ->  
# John Smith  
# ID 001 002  
# Salary 1000 2000  
# Designation TL Analyst*

**# OPERATORS #**

*# Arithmetic, Assignment, Comparison, Logical, Membership, Identity, Bitwise  
  
# Arithmetic Operators  
# Arithmetic operators are used to perform arithmetic between variables  
# Addition(+), Subtraction(-), Multiplication(\*), Division(/), Modules(%), Exponentiation(\*\*), Floor Division(//)*x = 10  
y = 20  
print(x + y) *# Output -> 30*print(x - y) *# Output -> -10*print(x \* y) *# Output -> 200*print(x / y) *# Output -> 0.5*print(x % y) *# Output -> 10*print(x \*\* y) *# Output -> 100000000000000000000*print(x // y) *# Output -> # 0  
  
# Assignment Operator  
# Assignment operator are used to assign values*x = 100  
x += 10 *# x = x + 10*print(x)  
x -= 10 *# x = x - 10*print(x)  
x \*= 10 *# x = x \* 10*print(x)  
x /= 10 *# x = x / 10*print(x)  
  
*# Comparison Operator  
# Comparison operators are used to compare to values  
# Equal(==), Not Equal(!=), Greater Than(>), Less Than(<), Greater than OR Equal(>=), Less Than OR Equal(<=)  
  
# Logical Operator  
# Logical Operator - and, or, not  
# Logical operator are used to combine condition statements  
# Conditional statement*val = 10  
num = 10  
if val == num:  
 print(**'Equal'**)  
elif val > num:  
 print(**'Greater'**)  
else:  
 print(**'Smaller'**)  
  
x = 10  
print(x > 10 and x > 5) *# Output -> False*print(x > 10 or x > 5) *# Output -> True*print(not(x > 10 or x > 5)) *# Output -> False  
  
# Identity Operator  
# Identity operator are used to compare objects  
# Two Identity operator 'is', and 'is not'  
# is - Returns true if both variables are same objects  
# is not - Returns true if both variables are not same objects*x = 10  
y = 20  
print(x is y) *# Output -> False*print(x is not y) *# Output -> True  
  
# Membership Operator  
# Membership operator are used to check if a sequence is present in an object  
# Two Membership operator 'in' and 'not in'  
# in - Return true if a sequence with the specific value is present in the object  
# not in - Return true if a sequence with the specific value is not present in the object*x = [10, 20, 30]  
print(10 in x) *# Output -> True*print(20 not in x) *# Output -> False  
  
# Bitwise Operator  
# Bitwise operator are used to compare binary operator  
# Bitwise AND(&) - Sets each bit to 1 if both bits are 1.  
# Bitwise OR(|) - Sets each bit to 1 if one of the bits is 1.  
# Bitwise XOR(^) - Sets each bit to 1 if only one of the bits is 1.  
# Bitwise NOT(~) - Inverts all bits.  
# Left Shift(<<) - Shift left by pushing in zeroes from the right and left the leftmost bits fell off  
# Right Shift(>>) - Shift left by pushing copies of the leftmost bit in from the left, and let the rightmost bit fall off*print(10 & 12) *# Output -> 8. => 10 - 1010, 12 - 1100 => 1010 1100 => 1000 => 8*

Program

For Loop

*# Access list using for loop*list1 = [**'Sandip'**, **'John'**, **'Marry'**]  
list2 = [[**'Sandip'**,1],[**'John'**,2],[**'Swap'**,3]]  
for item, item3 in list2:  
 print(item, item3)  
**'''Output  
Sandip 1  
John 2  
Swap 3  
'''***# Access Dictionary using for loop*list3 = {**'sandip'**:**'Salunhkhe'**,  
 **'Temp'**:**'woz'**,  
 **'Queen'**:**'pop'**}  
for pips, pips2 in list3.items():  
 print(pips,**'value is'**, pips2)  
**'''Output  
sandip value is Salunhkhe  
Temp value is woz  
Queen value is pop  
'''**

*# print the all values greater than 6*list4 = [3,4,6,2,4,**'sandip'**,22,**'John'**,4,53]  
for items in list4:  
 if str(items).isnumeric() and items > 6:  
 print(items)  
**'''Output  
22  
53'''**

While Loop

*# While Loop*i = 0  
while i < 45:  
 print(i)  
 i += 1

*# show value which is greater than 4 upto less than 45*i = 0  
while(True):  
 if i+1<5:  
 i = i + 1  
 continue  
  
 print(i+1, end=**' '**)  
 if(i==44):  
 break  
 i = i + 1  
*# Output -> 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45*

*# User enter a number upto if number is greater than 100 then stop that loop*lists = []  
i =0  
while i < 100:  
 i = int(input(**"Enter a number"**))  
 lists.append(i)  
print(lists)

*# Other method OR*

*# User enter a number upto if number is greater than 100 then stop that loop*lists = []  
while(True):  
 input1 = int(input(**'Enter a number'**))  
 lists.append(input1)  
 if input1 > 100:  
 print(**'Value is greater than 100'**)  
 break  
 else:  
 continue  
print(lists)

*# Guess the number with guess chances*guess = 18  
times = 9  
while times > 0:  
 user\_val = int(input(**'Enter a number**\n**'**))  
  
 if user\_val > 18:  
 print(**'Value is greater than guess value'**)  
 elif user\_val < 18:  
 print(**'Value is less than guess value'**)  
 else:  
 print(**'Guess number is correct'**)  
 print(**'Number of the guesses he took to finish '**, 10 - times)  
 break  
 times = times - 1  
 print(**'No. of guesses left '**,times)  
else:  
 print(**'Game over'**)