Variables in Python

What is a variable?

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory. example

```
a = 10 # a is a variable with value 10
b = 20 # b is a variable with value 20

name = "Kushal" # name is a variable with value Kushal
```

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example −

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
Married = True         # A boolean
```

Python input and output

Python Output Using print() function

We use the print() function to output data to the standard output device (screen). We can also output data to a file, but this will be discussed later. An example of its use is given below.

```
print("Hello World")
```

formatted output in python

```
name = "Kushal"
age = 20
print("My name is %s and my age is %d"%(name,age))
```

f string in python

```
name = "Kushal"
age = 20
print(f"My name is {name} and my age is {age}")
```

Python Input

Up until now, our programs were static. The value of variables was defined or hard coded into the source code.

To allow flexibility we might want to take the input from the user. In Python, we have the input() function to allow this. The syntax for input() is

```
input([prompt])
name = input("Enter your name")
print (name)
```

where prompt is the string we wish to display on the screen. It is optional.

conditional statement

if statement

if statement is used to check the condition and execute the statement if the condition is true. the condition runs on boolean value if the condition is true then the statement is executed otherwise not.

```
if condition:
    statement
```

if else statement

if else statement is used to check the condition and execute the statement if the condition is true otherwise execute the else statement.

```
if condition:
    statement
else:
    statement
```

if elif else statement

if elif else statement is used to check the condition and execute the statement if the condition is true otherwise execute the elif statement if the elif statement is also false then execute the else statement.

```
if condition:
    statement
elif condition:
    statement
else:
    statement
```

example of if elif else statement

```
a = 10
b = 20
if a > b:
    print("a is greater than b")
elif a < b:
    print("a is less than b")
else:
    print("a is equal to b")
```

Looping in python

looping is used to execute the statement multiple times.

for loop

for loop is used to execute the statement multiple times. syntax of for loop

```
for i in range(start,stop,step):
    statement
```

example 1

```
for i in range(10): # here 10 is the stop
    print(i)
```

example 2

```
for i in range(1,10): # here 1 is the start and 10 is the stop
    print(i)
```

example 3

```
for i in range(1,10,2): # here 2 is the step size
    print(i)
```

Python Built in data strycture

buiult in data structure are the data structure which are already defined in python.

list

list is a data structure which is used to store multiple values in a single variable. syntax of list

```
list_name = [value1,value2,value3]
```

example of list

```
list1 = [1,2,3,4,5]
list2 = ["kushal","Aaditya","Prashna"]
```

list indexing

in python list indexing starts from 0. example of list indexing

```
list1 = [1,2,3,4,5]
print(list1[0]) # output 1
print(list1[1]) # output 2
print(list1[2]) # output 3
print(list1[3]) # output 4
```

mutability of list

mutable means changeable. list is mutable data structure. we can change the value of list. example of mutability of list

```
list1 = [1,2,3,4,5]
list1[0] = 10
print(list1) # output [10,2,3,4,5]
```

list slicing

list slicing is used to get the sublist from the list. syntax of list slicing

```
list_name[start:stop:step]
```

example of list slicing

```
list1 = [1,2,3,4,5]
print(list1[0:3]) # output [1,2,3]
print(list1[0:5:2]) # output [1,3,5]
```

list methods

list methods are the methods which are used to perform the operation on the list.

append()

append() method is used to add the element at the end of the list. syntax of append() method

```
list_name.append(value)
```

example of append() method

```
list1 = [1,2,3,4,5]
list1.append(6)
print(list1) # output [1,2,3,4,5,6]
```

insert()

insert() method is used to add the element at the specified index. syntax of insert() method

```
list_name.insert(index,value)
```

example of insert() method

```
list1 = [1,2,3,4,5]
list1.insert(0,10)
print(list1) # output [10,1,2,3,4,5]
```

remove()

remove() method is used to remove the element from the list. syntax of remove() method

```
list_name.remove(value)
```

example of remove() method

```
list1 = [1,2,3,4,5]
list1.remove(1)
print(list1) # output [2,3,4,5]
```

pop()

pop() method is used to remove the element from the specified index. syntax of pop() method

```
list_name.pop(index)
```

example of pop() method

```
list1 = [1,2,3,4,5]
list1.pop(0)
print(list1) # output [2,3,4,5]
```

clear()

clear() method is used to clear the list. syntax of clear() method

```
list_name.clear()
```

example of clear() method

```
list1 = [1,2,3,4,5]
list1.clear()
print(list1) # output []
```

tuples

tuple is a data structure which is used to store multiple values in a single variable. syntax of tuple

```
tuple_name = (value1,value2,value3)
```

example of tuple

```
tuple1 = (1,2,3,4,5)
tuple2 = ("kushal","Aaditya","Prashna")
```

tuple indexing

in python tuple indexing starts from 0. *example of tuple indexing*

```
tuple1 = (1,2,3,4,5)
print(tuple1[0]) # output 1
print(tuple1[1]) # output 2
print(tuple1[2]) # output 3
print(tuple1[3]) # output 4
```

immutability of tuple

immutable means unchangeable. tuple is immutable data structure. we can not change the value of tuple. *example of immutability of tuple*

```
tuple1 = (1,2,3,4,5)
tuple1[0] = 10
print(tuple1) # output TypeError: 'tuple' object does not support item
assignment
```

List vs tuple

List	Tuple
List is mutable	Tuple is immutable
List is slower than tuple	Tuple is faster than list
List is defined in square brackets	Tuple is defined in round brackets
List is used when we want to change the value of the variable	Tuple is used when we do not want to change the value of the variable

Dictionary

dictionary is a data structure which is used to store multiple values in a single variable. value in the dictionary is stored in the form of key value pair. syntax of dictionary

```
dictionary_name = {key1:value1,key2:value2,key3:value3}
```

example of dictionary

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
```

dictionary indexing

we can access the value of dictionary using the key. *example of dictionary indexing*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
print(dictionary1["name"]) # output kushal
print(dictionary1["age"]) # output 20
print(dictionary1["city"]) # output pokhara
```

accessing the value of dictionary using the get() method. *example of accessing the value of dictionary using the get() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
print(dictionary1.get("name")) # output kushal
print(dictionary1.get("age")) # output 20
print(dictionary1.get("city")) # output pokhara
```

dictionary methods

dictionary methods are the methods which are used to perform the operation on the dictionary.

keys()

keys() method is used to get the keys of the dictionary. *example of keys() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
print(dictionary1.keys()) # output dict_keys(['name', 'age', 'city'])
```

values()

values() method is used to get the values of the dictionary. *example of values() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
print(dictionary1.values()) # output dict_values(['kushal', 20, 'pokhara'])
```

items()

items() method is used to get the items of the dictionary. *example of items() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
print(dictionary1.items()) # output dict_items([('name', 'kushal'), ('age',
20), ('city', 'pokhara')])
```

update()

update() method is used to update the dictionary. *example of update() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
dictionary1.update({"name":"Aaditya"})
print(dictionary1) # output {'name': 'Aaditya', 'age': 20, 'city':
'pokhara'}
```

pop()

pop() method is used to remove the element from the dictionary. *example of pop() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
dictionary1.pop("name")
print(dictionary1) # output {'age': 20, 'city': 'pokhara'}
```

popitem()

popitem() method is used to remove the last element from the dictionary. *example of popitem() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
dictionary1.popitem()
print(dictionary1) # output {'name': 'kushal', 'age': 20}
```

clear()

clear() method is used to clear the dictionary. *example of clear() method*

```
dictionary1 = {"name":"kushal","age":20,"city":"pokhara"}
dictionary1.clear()
print(dictionary1) # output {}
```

set

set is a data structure which is used to store multiple values in a single variable. values in sets are unordered and unindexed. we cannot store duplicate values in the set. syntax of set

```
set_name = {value1,value2,value3}
```

example of set

```
set1 = {1,2,3,4,5}
```

set indexing

we cannot access the value of set using the index because set is unordered and unindexed. *example of set indexing*

```
set1 = {1,2,3,4,5}
print(set1[0]) # output TypeError: 'set' object is not subscriptable
```

set methods

set methods are the methods which are used to perform the operation on the set.

add()

add() method is used to add the element in the set. *example of add() method*

```
set1 = {1,2,3,4,5}
set1.add(6)
print(set1) # output {1,2,3,4,5,6}
```

update()

update() method is used to update the set. *example of update() method*

```
set1 = {1,2,3,4,5}
set1.update([6,7,8,9,10])
print(set1) # output {1,2,3,4,5,6,7,8,9,10}
```

remove()

remove() method is used to remove the element from the set. *example of remove() method*

```
set1 = {1,2,3,4,5}
set1.remove(1)
print(set1) # output {2,3,4,5}
```

Arrays in python

array is a data structure which is used to store multiple values in a single variable. in python List is used as an array.