Variables in Python

- A variable is a location in memory used to store some data(value).
- They are given unique name to differentiate between diffferent memory locations.
- The rules for writing a variable name is same as the rules for writing identifiers in Python.
- We don't need to declare a variable before using it.In Python, we simply assign a value to a variable and it will exist.We don't even have to declare type of variable.This is handled internally according to the type of value we assign to the variable.

Variable Assignments

- We use the assignment operator (=) to assign values to a variable.

In [10]:
```
a = 10
b = 5.5
c = "CCT"
```

Multiple Assignment

In [11]:
```
a, b, c= 10, 5.5, "CCT"
```

In [17]:
```
# Assigning the same value to multiple variabls at once
a = b = c = "CCT"
```

Storage Location

In [14]:
```
x = 50
print(id(x))  #Print address of variable x
```

```
140043194941504
```

In [15]:
```
y = 50
print(id(x))  #Print address of variable y
```

```
140043194941504
```

Observation:

x and y point to same memory location

In [16]:
```
y = 2
print(id(y))  #Print address of variable y
```

```
140043194939968
```

Data Types in Python

- Data Type is a classification of data which tells the compiler or interpreter how the programmer intends to use the data.
- Every value in Python has a datatype.Since everything is an object in Python programming,data types are actually classes and variable are instance (object) of these classes.

Python Numbers

- Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.
  - Integers can be of any length, it is only limited by the memory available.
  - A floating point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points. 1 is integer, 1.0 is floating point number.
  - Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part.
- We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class. #### *Here are some examples:*

In [18]:
```
a = 10
print(a, "is of type", type(a))
```

```
10 is of type <class 'int'>
```

In [19]:
```
a = 2.0
print(a, "is of type", type(a))
```

```
2.0 is of type <class 'float'>
```

In [20]:
```
a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))
```

```
(1+2j) is complex number? True
```

In [21]:
```
 b = 0.1234567890123456789
 b
```

Out[21]:
```
0.12345678901234568
```

Notice that the float variable b got truncate.

Boolean

Boolean represents the truth values *False* and *True*

In [19]:
```
a = True
print(type(a))
```

```
<class 'bool'>
```

Python Strings

- String is sequence of Unicode characters.
- We can use single quotes or double quotes to represent strings.
- Multi-Line strings cen be denoted using triple quotes, ''' or """.
- A String in Python consists of a series or sequence of characters - letters, numbers and special characters.
- Strings can be indexed-often synonymously called subscripted as well.
- Similar to C, the first character of a string has the index 0.
- Like list and tuple, slicing operator [ ] can be used with string.
- Strings are immutable.

In [28]:
```
s = '''This is online
                Python Course'''

print(s)
print(type(s))
```

```
This is online
                Python Course
<class 'str'>
```

In [29]:
```
s[0]
```

Out[29]:
```
'T'
```

In [30]:
```
#Slicing
s[5:]
```

Out[30]:
```
'is online    \n               Python Course'
```

In [33]:
```
# Methods to print last element
print(s[-1])
print(s[len(s)-1])
```

```
e
e
```

Python List

- List is an ordered sequence of items.
- It is one of the most used datatype in Python and is very flexible.
- All the items in a list do not need to be of the same type.

Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].

In [39]:
```
a = [1, 2.2, 'CCT']
b = [] #Declaring empty list
```

In [40]:
```
a,b
```

Out[40]:
```
([1, 2.2, 'CCT'], [])
```

We can use the slicing operator [ ] to extract an item or a range of items from a list. Index starts form 0 in Python.

In [37]:
```
a = [5,10,15,20,25,30,35,40]

# a[2] = 15
print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]
print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]
print("a[5:] = ", a[5:])
```

```
a[2] =  15
a[0:3] =  [5, 10, 15]
a[5:] =  [30, 35, 40]
```

Lists are mutable, meaning, value of elements of a list can be altered.

In [38]:
```
a = [1,2,3]
a[2] = 10
a
```

Out[38]:
```
[1, 2, 10]
```

Python Tuple

- Tuple is an ordered sequence of items same as list.
- The only difference is that tuples are immutable.
- Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically.
- It is defined within parentheses () where items are separated by commas.

```
In [41]: t = (5,'CCT', 1+3j)

In [43]: t
Out[43]: (5, 'CCT', (1+3j))
```

We can use the slicing operator [] to extract items but we cannot change its value.

```
In [44]: t[1]
Out[44]: 'CCT'

In [46]: t[1:3]
Out[46]: ('CCT', (1+3j))

In [47]: t[2] = 'Python'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-a04a95a2b6a2> in <module>()
----> 1 t[2] = 'Python'

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

Python Set

- Set is an unordered collection of unique items.
- Set is defined by values separated by comma inside braces { }.
- Items in a set are not ordered.

In [48]:
```
a = {5,2,3,1,4}

# printing set variable
print("a = ", a)

# data type of variable a
print(type(a))
```

```
a =  {1, 2, 3, 4, 5}
<class 'set'>
```

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates.

In [49]:
```
a = {1,2,2,3,3,3}
```

In [50]:
```
a
```

Out[50]:
```
{1, 2, 3}
```

Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [ ] does not work.

In [51]:
```
a[1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-51-3ef3908cabc7> in <module>()
----> 1 a[1]

TypeError: 'set' object does not support indexing
```

Python Dictionary

- Dictionary is an unordered collection of key-value pairs.
- It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.
- In Python, dictionaries are defined within braces {} with each item being a pair in the form key:value. Key and value can be of any type.

In [2]:
```
d = {'a':'apple','b':'bat'}
```

In [3]:
```
print(type(d))
```

```
<class 'dict'>
```

We use key to retrieve the respective value.

In [4]:
```
print("d['a'] = ", d['a']);
```

```
d['a'] =  apple
```

In [5]:
```
print("d['b'] = ", d['b']);
```

```
d['b'] =  bat
```

In [6]:
```
# Generates error
print("d['c'] = ", d['c']);
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-6-5219d86b3b48> in <module>()
      1 # Generates error
----> 2 print("d['c'] = ", d['c']);

KeyError: 'c'
```

In [ ]:

In [ ]: