Print Hello World!

In [4]:
```
print("Hello World!")
```
```
Hello World!
```

Variables

In [5]:
```
a=10
b=20
a+b
print(a+b)
```
```
30
```

String

In [9]:
```
# double quote
s="arya"
print(s)

# single quote
s1='arya'
print(s1)

# triple quote
s2='''
a
ry
a'''
print(s2)

# advantage with triple quote is we can print in diff lines without explicitly mentioning it
```
```
arya
arya

a
ry
a
```

String Indexing in 'python'

In [14]:
```
s="arya"
print(s[0])
print(s[1])
print(s[2])
print(s[3])

print()

print(s[-1])
print(s[-2])
print(s[-3])
print(s[-4])
```
```
a
r
y
a

a
y
r
a
```

In [16]:
```
s='''My
Name
Is
Arya'''

# print(s)

s
```
Out[16]:
```
'My\nName\nIs\nArya'
```

How Strings are stored

In [20]:
```
s="arya"
a="arya"

print(id(s))
print(id(a)) # addresses where it stores "arya" are same in case of a and s both
```
```
2335726790768
2335726790768
```

Immutability of strings in Python

strings are immutable in Python, we can retrieve a char from a string but cannot modify that position of the string

Functions on strings

split

In [25]:
```
str="my name is arya"

li=str.split() # automatically splits on the basis of space if no arguments passed
print(li)

str1="my,name,is,arya"
li1=str1.split(',')
print(li1)

li2=str1.split(',', 1) # the other argument allows us to split the pecified number of times
print(li2)
```
```
['my', 'name', 'is', 'arya']
['my', 'name', 'is', 'arya']
['my', 'name,is,arya']
```

replace

In [34]:
```
str="my name is arya"
str1 = str.replace("arya", "tito")

print(str) # will not make any change in original string (due to immutability principle)

print(str1)

str2 = str.replace("arjo", "tito") # no "arjo" present
print(str2)

strx="my name is arya arya arya"
strx1 = strx.replace("arya", "tito")
print(strx1)

strx2 = strx.replace("arya", "tito", 2) # specifying the no. of times we want to change
print(strx2)
```
```
my name is arya
my name is tito
my name is arya
my name is tito tito tito
my name is tito tito arya
```

find

In [42]:
```
str="my name is arya"
idx = str.find("na");
print(idx) # returns the start index if substring is present

idx1 = str.find("nae") # else returns -1
print(idx1)

str1="my name is arya arya arya"
idx2 = str.find("arya")
print(idx2)

idx3 = str1.find("arya", 12, 20) # from 12 to 20 if "arya" is present
print(idx3)
```
```
3
-1
11
16
```

lower and upper

In [47]:
```
str="my name is arya"
str1=str.lower();
str2=str.upper();

print(str1)
print(str2)
```
```
my name is arya
MY NAME IS ARYA
```

starts-with

In [52]:
```
str="my name is arya"
ans=str.startswith("my", 3, 10)
ans1=str.startswith("my")

print(ans)
print(ans1)
```
```
False
True
```

slicing on string

[start : end : steps]

In [2]:
```
str="arya"
str[1:4] # it'll consider steps to be 1 if we don't specify
```
Out[2]:
```
'rya'
```

In [4]:
```
str[1:4:2]
```
Out[4]:
```
'ra'
```

In [5]:
```
str[6:4:3] # start index is greater, so it'll return an empty string
```
Out[5]:
```
''
```

In [6]:
```
str[-3:-1] # 3rd last element to 1st last(last) element
```
Out[6]:
```
'ry'
```

In [7]:
```
str[1:]
```
Out[7]:
```
'rya'
```

In [8]:
```
str[:3]
```
Out[8]:
```
'ary'
```

In [9]:
```
str[1:9] # will go till the last
```
Out[9]:
```
'rya'
```

negative stride

In [17]:
```
str[4:0:-1]
```
Out[17]:
```
'ayr'
```

In [18]:
```
str[5::-1] # basically reverse
```
Out[18]:
```
'ayra'
```

In [22]:
```
str=str[:1:-1]
print(str)
```
```
ay
```

In [24]:
```
str1='hello'
print(str1[-1:])
```
```
o
```

In [26]:
```
s="abcdefghi"
s[-3:-1]
```
Out[26]:
```
'gh'
```

Tuple

Tuples are used to store multiple items in a single variable.

In [28]:
```
a=(1,2)
print(a)
```
```
(1, 2)
```

In [29]:
```
type(a)
```
Out[29]:
```
tuple
```

In [30]:
```
b, c=3, 4
print(b)
print(c)
```
```
3
4
```

In [31]:
```
e=3, 4 # even if we don't put parentheses
print(e)
print(type(e))
```
```
(3, 4)
<class 'tuple'>
```

retrieving data from tuple

In [34]:
```
print(e[0])
print(e[-1])
```
```
3
4
```

In [38]:
```
# slicing
e=(1, 2, 3, 4, 5, 6)
print(e[0:4])
print(e[0:4:2])
print(e[-4:-1:])
```
```
(1, 2, 3, 4)
(1, 3)
(3, 4, 5)
```

difference between tuples and lists

the basic difference between tuple and list, is same as that of string and char array. </br> tuple is just the immutable version of list. </br> Thus, if we want, we can delete a tuple but not an element in a tuple if it exists.

In [39]:
```
a=1, 2, 3
b=4, 5, 6, 7
for i in a:
    print(i)
```
```
1
2
3
```

In [43]:
```
print(1 in a)
print(1 in b)
```
```
True
False
```

In [45]:
```
len(b)
```
Out[45]:
```
4
```

In [47]:
```
c=a+b
print(c)
```
```
(1, 2, 3, 4, 5, 6, 7)
```

In [49]:
```
# tuple of tuples
d=(a, b)
print(d)
```
```
((1, 2, 3), (4, 5, 6, 7))
```

In [51]:
```
e=a*4 # also a tuple
e
```
Out[51]:
```
(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
```

In [53]:
```
print(min(a))
print(max(a))
```
```
1
3
```

In [54]:
```
# tuple can contain multiple data types
x=(1, 2, "abc", (3, 4))
x
```
Out[54]:
```
(1, 2, 'abc', (3, 4))
```

but in the above example we cannot find min(x) and max(x)

In [57]:
```
# list to tuple
a=[1, 2, 3, 4]
tuple(a)
```
Out[57]:
```
(1, 2, 3, 4)
```

THANK YOU!