In [45]:
```
# Old formatting style (% Operator)

name = "TOM"
surname = "Thomson"
print('hello %s' % name)

# if we want to take multiple inputs
print('hello %s %s!' %(name,surname))
```

```
hello TOM
hello TOM Thomson!
```

In [ ]:
```

```

In [ ]:
```
# New formatting (str.format)
```

In [3]:
```
# .format method
name = "Mike"
age = 14
s = "My name is {}. My age is {}".format(name,age)

print(s)
```

```
My name is Mike. My age is 14
```

In [ ]:
```

```

In [4]:
```
# using f string method

name = "Mike"
age = 14
s2 = f"My name is {name}. My age is {age}"
print(s2)
```

```
My name is Mike. My age is 14
```

In [6]:
```
# changine the positional arguments using format method

name = "Mike"
age = 14
s = "My age is {} \nMy name is {}".format(age,name)
    # interchange positional arg in format
print(s)
```

```
My age is 14
My name is Mike
```

In [8]:
```
# changing the positional argumnet passing : indices

name = "Mike"
age = 14
s3 = "My name is {1}. \nMy age is {0}".format(age,name)
    # interchange positional arg in format
print(s3)
```

```
My name is Mike.
My age is 14
```

In [27]:
```
# we can pass anything inside format string,
# No matter what it is

nest_dic = {1:
               {
                   "name":"mike",
                   "age":"14"
               }
            }


print("Hey my details are {}".format(nest_dic))

nested_list = [1,["mike","14"]]

print(f"Hey my new details are {nested_list}")
```

```
Hey my details are {1: {'name': 'mike', 'age': '14'}}
Hey my new details are [1, ['mike', '14']]
```

In [30]:
```
nested_list = ["details",["mike","14"]]

print(nested_list,type(nested_list))

nest_dic = {1:
               {
                   "name":"mike",
                   "age":"14"
               }
            }

print(nest_dic,type(nest_dic))

# changing the keys of the nested dic by using variables

a = "NAME : "
b = "AGE : "
nest_dic2 = {1:
               {
                   a:"mike",
                   b:"14"
               }
            }
print(nest_dic2,type(nest_dic2))
```

```
['details', ['mike', '14']] <class 'list'>
{1: {'name': 'mike', 'age': '14'}} <class 'dict'>
{1: {'NAME : ': 'mike', 'AGE : ': '14'}} <class 'dict'>
```

In [ ]:
```

```

In [55]:
```
# conversion Flags [DOUBTQ]

'''
String formatting statements:
'{0}'.format(a)
'{0!s}'.format(a)

'{0}'.format(a) will use the result of a.__format__() to display the valu
'{0!s}'.format(a) will use the result of a.__str__() to display the value
'{0!r}'.format(a) will use the result of a.__repr__() to display the valu
'''

a = [1, 2, "µ"]
b = [1, 2, 3]

# there no difference between them when there is only a single argument
print("The list is {}".format(a))

print("The list is {0}".format(a))


# convert he 3 item of the list to string
print("The list after changing the 3rd item to string is {0!s}".format(b)
```

```
The list is [1, 2, 'µ']
The list is [1, 2, 'µ']
The list after changing the 3rd item to string is [1, 2, 3]
```

In [1]:
```
# Conversion Flags ::

b = "6464"
a = [1, 2, "µ"]

print("{!s}".format(b))

print("{}".format(b))

print("{0}".format(b))

print("{0!s}".format(b))

print("{0!r}".format(b))

print("{!a}".format(b))

##########XXX###########

print("{!s}".format(a))

print("{}".format(a))

print("{0}".format(a))

print("{0!s}".format(a))

print("{0!r}".format(a))

print("{!a}".format(a))

##########XXX###########

product = "Story Book"
rs = 23.3434
print(f"\n{product} for only {rs:.2f} ")
```

```
6464
6464
6464
6464
'6464'
'6464'
[1, 2, 'µ']
[1, 2, 'µ']
[1, 2, 'µ']
[1, 2, 'µ']
[1, 2, 'µ']
[1, 2, '\xb5']

Story Book for only 23.34
```

In [ ]:

In [5]:
```python
# Formatting Types ::

'''
Left Aligned : {:<20}
Right Aligned : {:>20}
Center Alogned : {:^20}
'''

a = "this is a left {:<20} string"
print(a.format("aligned"))

b = "this is a right {:>20} string"
print(b.format("aligned"))

c = "this is a right {:^20} string"
print(c.format("aligned"))
```

```
this is a left aligned              string
this is a right              aligned string
this is a right       aligned        string
```

In [ ]:

In [ ]:

In [ ]:

In [14]:
```python
        # String Interpolation / f-Strings
# f-string added in 3.6+
name = "Mike"
print("My name is {}".format(name))

# normal f-string
surname = "Thomson"
print(f"My name is {name} {surname}")

print(f"My name is {name+' '+surname} !")

# using fstring to solve math problem

a = 10
b = 20
print(f"Sum of {a} and {b} is {a+b}")
```

```
My name is Mike
My name is Mike Thomson
My name is Mike Thomson !
Sum of 10 and 20 is 30
```

In [26]:
```python
# create a greet functn using fstring

def greet(name,surname):
    return f"Good Morning {name} {surname}"

greet("TOM",surname)
```

Out[26]:
```
'Good Morning TOM Thomson'
```

In [29]:
```
# create a greet fucntn withour using fstring

def greet2(name,surname):
    return "Good Morning "+name+" "+surname

greet2("TOM","Thomson")
```

Out[29]:
```
'Good Morning TOM Thomson'
```

In [ ]:

In [18]:
```
# Template String

from string import Template
t = Template('Hey, $name!') # $ sign is imp
# if removed normal string will be printed
t.substitute(name=name)
```

Out[18]:
```
'Hey, Mike!'
```

In [23]:
```
name = "Harsh"
from string import Template
t = Template('Hey, $n!')
t.substitute(n=name) # variable is stored in n
```

Out[23]:
```
'Hey, Harsh!'
```

In [ ]: