Different Way to use print function in python

In [1]:
```
print (5*0.6 - 0.5*9.81*0.6**2)
```

```
1.2342
```

In [2]:
```
print("Hello World!!!")
```

```
Hello World!!!
```

In [4]:
```
v0 = 5
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2
print (y)
```

```
1.2342
```

In [6]:
```
v0 = 5
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2
print ('At t=%g s, the height of the ball is %.2f m.' % (t, y))
```

```
At t=0.6 s, the height of the ball is 1.23 m.
```

In [9]:
```
v0 = 5
g = 9.81
t = 0.6
y = v0*t - 0.5*g*t**2

print ("""
At t=%f s, a ball with
initial velocity v0=%.3E m/s
is located at the height %.2f m.
""" % (t, v0, y))
```

```
At t=0.600000 s, a ball with
initial velocity v0=5.000E+00 m/s
is located at the height 1.23 m.
```

In [10]:
```
# use of \n
print ( """y(t) is
the position of
our ball.""")
print ('y(t) is\nthe position of\nour ball')
```

```
y(t) is
the position of
our ball.
y(t) is
the position of
our ball
```

In [8]:
```
#use of ;
v0 = 3; g = 9.81; t = 0.6
y = v0*t - 0.5*g*t**2
print (y)
#OR
v0=3;g=9.81;t=0.6;y=v0*t-0.5*g*t**2;print (y)
```

```
0.034199999999999786
0.034199999999999786
```

In [11]:
```
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

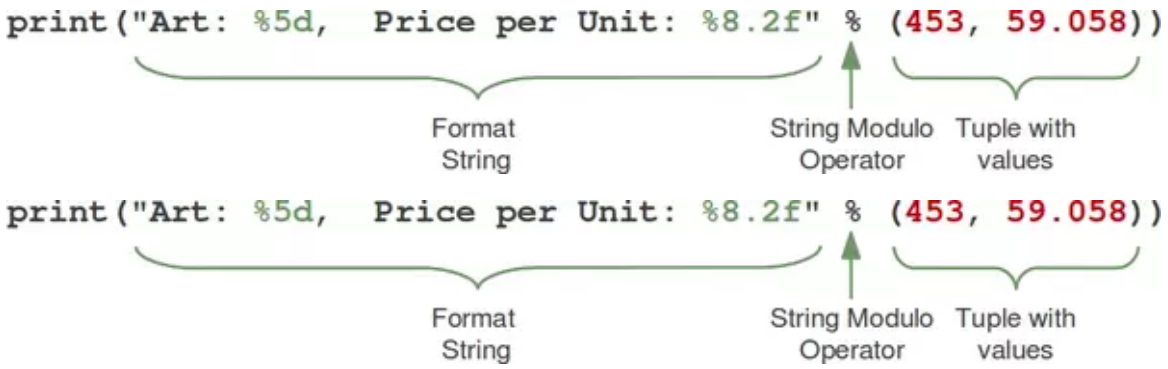
In [12]:
```
print('V','S','U', sep= '\t', end='\n')
```

```
V	S	U
```

In [15]:
```
print('V','S','U', sep=",\t", end='X')
print('V','S','U', sep=",\t", end='\n')
```

```
V,	S,	UXV,	S,	U
```

Print using string modulo operator.

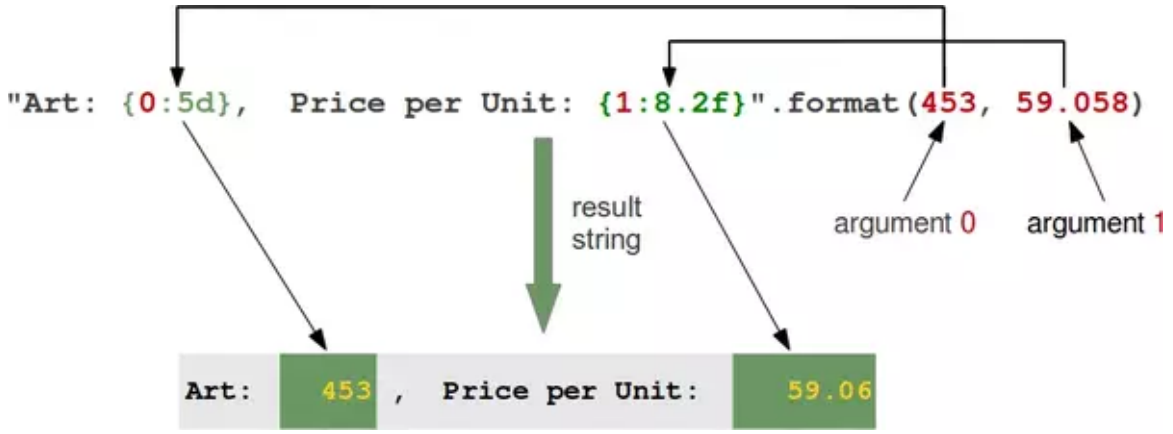


The general syntax for a format placeholder is %[flags][width][.precision]type

In [5]:
```
i = 62
r = 189876575.7654675432

# Print out numbers with quotes "" such that we see the
# width of the field
print ('"%d"'  %i)     # minimum field
print ('"%5d"' % i )     # field of width 5 characters
print ('"%05d"' % i)    # pad with zeros

print ('"%g"' % r)       # r is big number so this is scientific notation
print ('"%G"' % r)       # E in the exponent
print ('"%e"' % r )      # compact scientific notation
print ('"%E"' % r)       # compact scientific notation
print ('"%20.2E"' % r )  # 2 decimals, field of width 20
print ('"%30g"' % r)     # field of width 30 (right-adjusted)
print ('"%-30g"' % r)    # left-adjust number
print ('"%-30.3g"' % r ) # 3 decimals

print ('%s' % i)   # can convert i to string automatically
print ('%s' % r)

print ('%r' % r)

# Use %% to print the percentage sign
print ('%g %% of %.2f Euro is %.2f Euro' % \
      (5.1, 346, 5.1/100*346))

temp=30
print("'%+-5d'"%temp)
```

```
"62"
"   62"
"00062"
"1.89877e+08"
"1.89877E+08"
"1.898766e+08"
"1.898766E+08"
"            1.90E+08"
"                  1.89877e+08"
"1.89877e+08                  "
"1.9e+08                      "
62
189876575.76546755
189876575.76546755
5.1 % of 346.00 Euro is 17.65 Euro
'+30  '
```

Print using format function



In [28]:
```
a=5;b=2
print('a={0}'.format(a)) #a=5
print('a={0} and b={1}'.format(a, b)) #a=5 and b=2
print('a={1} and b={0}'.format(a, b)) #a=2 and b=5
print('a={} and b={}'.format(a, b))     #a=5 and b=2 #auto field numbering
print('bin={0:b}, oct={0:o}, hex={0:x}'.format(12)) #bin=1100, oct=14, hex=c
print('bin={0:b}, oct={1:o}, hex={1:x}'.format(12,10)) #bin=1100, oct=12, hex=a
print('bin={:b}, oct={:o}, hex={:X}'.format(12,10,10)) #bin=1100, oct=12, hex=A
```

```
a=5
a=5 and b=2
a=2 and b=5
a=5 and b=2
bin=1100, oct=14, hex=c
bin=1100, oct=12, hex=a
bin=1100, oct=12, hex=A
```

In [3]:
```
a=5
b=2
print("a={0:d} and b={1:d}".format(a, b))                    #a=5 and b=2
print("a={0:3d} and b={1:5d}".format(a, b))                  #a=  5 and b=    2
print("a={0:>3d} and b={1:>5d}".format(a, b))                #a=  5 and b=    2
print("a={0:<3d} and b={1:<5d}".format(a, b))                #a=5   and b=2
print("a={:<{}d} and b={:<{}d}".format(a,3,b,5))              #???
print("a={0:03d} and b={1:05d}".format(a, b))                #a=005 and b=00002
print("a={0:^3d} and b={1:^5d}".format(a, b))                #a= 5  and b=  2
print("a={:f}".format(123.4567898990))                       #a=123.456790
print("a={:8.3f}".format(123.4567898))                       #a= 123.457
```

```
a=5 and b=2
a=  5 and b=    2
a=  5 and b=    2
a=5   and b=2
a=5             and b=2
a=005 and b=00002
a= 5  and b=  2
a=123.456790
a= 123.457
```

In [36]:
```
for i in range (2, 6):
    # Using formatters to give 6
    # spaces to each set of values
    print("{:>6d} {:<6d} {:^6d} {:6d}"
    .format(i, i ** 2, i ** 3, i ** 4))
```

```
     2 4        8        16
     3 9        27       81
     4 16       64      256
     5 25      125      625
```

Find Data type of variable

In [13]:
```
x = "Hello World"
print(type(x))
x = 20
print(type(x))
x = 20.5
print(type(x))
x = 1j
print(type(x))
x = ["apple", "banana", "cherry"]
print(type(x))
x = ("apple", "banana", "cherry")
print(type(x))
x = {"name" : "John", "age" : 36}
print(type(x))
```

```
<class 'str'>
<class 'int'>
<class 'float'>
<class 'complex'>
<class 'list'>
<class 'tuple'>
<class 'dict'>
```

In [1]:
```
x = str("s1")
y = str(2)
z = str(3.0)
print(x)
print(y)
print(z)
print(type(z))
```

```
s1
2
3.0
<class 'str'>
```

In [3]:
```
str1 = 'Hello World!'
print (str1)          # Prints complete string
print (str1[0])       # Prints first character of the string
print (str1[2:5])     # Prints characters starting at index 2 to 4
                     # first index is inclusive while end is exclusive
print (str1[2:])     # Prints string starting from 3rd character
print (str1 * 2)      # Prints string two times
print (str1 + "TEST")
x=str1 * 2
print(x)
print(str1[-2])
str1='Nirma University'
b = "Hello, World!"
# H  e  l  l  o  ,    W o r l  d  !
# 0  1  2  3  4  5  6 7 8 9 10 11 12 positive in
# 13 12 11 10 9  8  7 6 5 4 3  2  1  negative in
print(b[-5:-2])
print(b[-5:])
print(b[-10:9])
print(b[1:-5])
print(b[-2:-5])
print(len(b))  #To get the length of a string, use the len() function.
```

```
Hello World!
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
Hello World!Hello World!
d
orl
orld!
lo, Wo
ello, W

13
```

In [7]:
```
a = " Hello, Wor,ld! "
print(a.strip()) # returns "Hello, World!"
#a=a.strip()
#print(a)
#The strip() method removes any whitespace from the
# beginning or the end:
print(a.lower())
print(a.upper())

print(a.replace("H", "J"))
print(a.replace("el", "dx"))
print(a.split(",")) # returns ['Hello', ' World!']
#The split() method splits the string into substrings if it finds instances of the
separator:
a = "Hello"
b = "World"
c = a + b
print(c)
```

```
Hello, Wor,ld!
 hello, wor,ld!
 HELLO, WOR,LD!
 Jello, Wor,ld!
 Hdxlo, Wor,ld!
[' Hello', ' Wor', 'ld! ']
HelloWorld
```

In [8]:
```
str1 = 'Hello World!'

print (str1[-1])
print (str1[-3:-1])
print (str1[-12:])
```

```
!
ld
Hello World!
```

In [28]:
```
a=10
print(type(a))
```

```
<class 'int'>
```

In [29]:
```
x = 35e3
y = 12E4
z = -87.7e100
a=1234354.123456789185677
print(x,y,z,a)
print(type(x))
print(type(y))
print(type(z))
print(type(a))
```

```
35000.0 120000.0 -8.77e+101 1234354.1234567892
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

In [8]:
```
x = 3+5j
y = 5j
z = -5j
print(x,y,z)
print(type(x))
print(type(y))
print(type(z))
```

```
(3+5j) 5j (-0-5j)
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

In [10]:
```
x = 1 # int
y = 2.8 # float
z = 1j # complex
#convert from int to float:
a = float(x)
print(a)
#convert from float to int:
b = int(y)
print(b)
#convert from int to complex:
a=2
temp=0
c = complex(x,temp)
print(a)
print(b)
print(c)
print(type(a))
print(type(b))
print(type(c))
```

```
1.0
2
2
2
(1+0j)
<class 'int'>
<class 'int'>
<class 'complex'>
```

In [5]:
```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

In [6]:
```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.'''
print(a)
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
```

In [11]:
```
# Python docstrings are the string literals that appear right
# after the definition of a function, method, class, or module.
# Let's take an example.
def square(n):
    '''Takes in a number n,    returns the square of n'''
    return n**2

print(square.__doc__)
```

```
Takes in a number n,    returns the square of n
```

In [12]:
```
a = "Hello, World!"
print(a[1])
```

```
e
```

Convertion of int,float and str

In [9]:
```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("39") # z will be 3
print(x,y,z)
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
print(x,y,z,w)
```

```
1 2 39
1.0 2.8 3.0 4.2
```

In [37]:
```
x = str("s1")
y = str(2)
z = str(3.0)
print(x)
print(y)
print(z)
print(type(x))
print(type(y))
print(type(z))
```

```
s1
2
3.0
<class 'str'>
<class 'str'>
<class 'str'>
```

In [39]:
```
x = 15
y = 2
print(x / y)
print(x // y)
```

```
7.5
7
```

Collecting inputs from user...

It will work ust like scanf() in C

In [7]:
```
a=input("Enter a:") # by dafault input value is string
print(type(a))
print(a)
a=int(input("Enter a:"))
print(type(a))
print(a)
```

```
Enter a:12
<class 'str'>
12
Enter a:14
<class 'int'>
14
```

In [1]:
```
input = eval(input("Enter any number of your choice:") )  # enter 10 + 10
print(input)
print(type(input))
```

```
Enter any number of your choice:10 + 20
30
<class 'int'>
```

In [2]:
```
# eval = eval(input("Enter any number of your choice"))
print(eval)
print(type(eval))
```

```
<built-in function eval>
<class 'builtin_function_or_method'>
```

In [1]:
```
evaluate = input("Enter what operation x has to perform: ")
# enter x + x + 100 - 35 + 5 * 80
print(evaluate)
print(type(evaluate))

x = int(input("x"))
print(type(x))

expression = eval(evaluate)
print(expression)
print(type(expression))
```

```
Enter what operation x has to perform: x + x + 100 - 35 + 5 * 80
x + x + 100 - 35 + 5 * 80
<class 'str'>
x10
<class 'int'>
485
<class 'int'>
```

In [ ]:
```

```