In [8]:

```
#problem 6


#Part a
s1 = "HELLO"

s2 = "hello"

if s1 == s2:
    print("They are equal")
else:
     print("They are not equal")


#part b
if s1.lower() == s2:
    print("They are equal")
else:
     print("They are not equal")

#part c
if s1 == s2.upper():
    print("They are equal")
else:
     print("They are not equal")
```

```
They are not equal
They are equal
They are equal
```

In [9]:

```
# problems 3-4

from math import *
x = 10
y = 3
u = x + y
v = x*y
w = x/y
z = sin(x)
r = 8*sin(x)
s = 5*sin(x*y)
p = x**y
%whos
```

```
Variable    Type                          Data/Info
---------------------------------------------------
acos        builtin_function_or_method    <built-in function acos>
acosh       builtin_function_or_method    <built-in function acosh>
asin        builtin_function_or_method    <built-in function asin>
asinh       builtin_function_or_method    <built-in function asinh>
atan        builtin_function_or_method    <built-in function atan>
atan2       builtin_function_or_method    <built-in function atan2>
atanh       builtin_function_or_method    <built-in function atanh>
ceil        builtin_function_or_method    <built-in function ceil>
copysign    builtin_function_or_method    <built-in function copysign>
cos         builtin_function_or_method    <built-in function cos>
cosh        builtin_function_or_method    <built-in function cosh>
degrees     builtin_function_or_method    <built-in function degrees>
e           float                         2.718281828459045
erf         builtin_function_or_method    <built-in function erf>
erfc        builtin_function_or_method    <built-in function erfc>
exp         builtin_function_or_method    <built-in function exp>
expm1       builtin_function_or_method    <built-in function expm1>
fabs        builtin_function_or_method    <built-in function fabs>
factorial   builtin_function_or_method    <built-in function factorial>
floor       builtin_function_or_method    <built-in function floor>
fmod        builtin_function_or_method    <built-in function fmod>
frexp       builtin_function_or_method    <built-in function frexp>
fsum        builtin_function_or_method    <built-in function fsum>
gamma       builtin_function_or_method    <built-in function gamma>
gcd         builtin_function_or_method    <built-in function gcd>
hypot       builtin_function_or_method    <built-in function hypot>
inf         float                         inf
isclose     builtin_function_or_method    <built-in function isclose>
isfinite    builtin_function_or_method    <built-in function isfinite>
isinf       builtin_function_or_method    <built-in function isinf>
isnan       builtin_function_or_method    <built-in function isnan>
ldexp       builtin_function_or_method    <built-in function ldexp>
lgamma      builtin_function_or_method    <built-in function lgamma>
log         builtin_function_or_method    <built-in function log>
log10       builtin_function_or_method    <built-in function log10>
log1p       builtin_function_or_method    <built-in function log1p>
log2        builtin_function_or_method    <built-in function log2>
modf        builtin_function_or_method    <built-in function modf>
nan         float                         nan
p           int                           1000
pi          float                         3.141592653589793
pow         builtin_function_or_method    <built-in function pow>
r           float                         -4.352168887114958
radians     builtin_function_or_method    <built-in function radians>
remainder   builtin_function_or_method    <built-in function remainder>
s           float                         -4.940158120464309
s1          str                           HELLO
s2          str                           hello
sin         builtin_function_or_method    <built-in function sin>
sinh        builtin_function_or_method    <built-in function sinh>
sqrt        builtin_function_or_method    <built-in function sqrt>
tan         builtin_function_or_method    <built-in function tan>
tanh        builtin_function_or_method    <built-in function tanh>
tau         float                         6.283185307179586
trunc       builtin_function_or_method    <built-in function trunc>
u           int                           13
v           int                           30
w           float                         3.3333333333333335
x           int                           10
y           int                           3
z           float                         -0.5440211108893698
```

In [10]:

```
#problem 5

S = '123'
print(f'S is a {type(S)}')
N = float(S)
print(f'N is a {type(N)}')
```

```
S is a <class 'str'>
N is a <class 'float'>
```

In [7]:

```
#problem 7

s1 = 'Engineering'

s2 = 'Book'

print(f'The word "Engineering" has {len(s1)} letters.')
print(f'The word "Book" has {len(s2)} letters.')
```

```
The word "Engineering" has 11 letters.
The word "Book" has 4 letters.
```

In [12]:

```
#problem 8

s1 = "Python is great!"

s2 = "Python"

if s2 in s1:
    print(f"The word {s2} exists in {s1}")
else:
    print(f"The word {s2}  does not exist in {s1}")
```

```
The word Python exists in Python is great!
```

In [19]:

```
#problem 9

s1 = "Python is great!"
word = s1.split()
print("The last word of the string is:", word[2])
```

```
The last word of the string is: great!
```

In [22]:

```
#problems 10-11

list_a = [1,8,9,15]
print("The Original list is", list_a)

list_a.insert(1,2)
list_a.append(4)
print("The Modified list is", list_a)

list_a.sort()
print("The sorted list is", list_a)
```

```
The Original list is [1, 8, 9, 15]
The Modified list is [1, 2, 8, 9, 15, 4]
The sorted list is [1, 2, 4, 8, 9, 15]
```

In [1]:

```
#problem 12

s1 = "Python is great!"
Lst = list(s1)
print(Lst)
```

```
['P', 'y', 't', 'h', 'o', 'n', ' ', 'i', 's', ' ', 'g', 'r', 'e', 'a', 't', '!']
```

In [3]:

```
#problem 13

Tuple_a = ("One",1)
print(Tuple_a)
```

```
('One', 1)
```

In [5]:

```
#problem 14

Tuple_a = ("One",1)
print(Tuple_a[1])
```

```
1
```

In [6]:

```
#problem 15

Tuple_a = (2, 3, 2, 3, 1, 2, 5)
print(set(Tuple_a))
```

```
{1, 2, 3, 5}
```

In [12]:

```
#problem 16

set_a = {2,3,2}

set_b = {1,2,3}

print(f"Set A is: {set_a}")
print(f"Set B is: {set_b}")
print(f"Union of A and B is: {set_a.union(set_b)}")
print(f"Intersection of A and B is: {set_a.intersection(set_b)}")
print(f"Difference of A and B is: {set_b.difference(set_a)}")
```

```
Set A is: {2, 3}
Set B is: {1, 2, 3}
Union of A and B is: {1, 2, 3}
Intersection of A and B is: {2, 3}
Difference of A and B is: {1}
```

In [13]:

```
#problem 17

Dict = {"A":"a", "B":"b", "C":"c"}
print("All of the keys are", Dict.keys())
```

```
All of the keys are dict_keys(['A', 'B', 'C'])
```

In [15]:

```
#problem 18
Dict = {"A":"a", "B":"b", "C":"c"}
if "B" in Dict:
    print("Key B is in the dictionary")
else:
    print(" Key B is not in the dictionary")
```

```
Key B is in the dictionary
```

In [4]:

```
#problem 19

import numpy as np
from math import *

x = np.array([1, 4, 3, 2, 9, 4])
print("x =", x)

y = np.array([2, 3, 4, 1, 2, 3])
print("y =", y)

u = x + y
print("x + y = ", u)

v = x*y
print("v =", v)

w = x/y
print("w =", w)

z = np.sin(x)
print("z =", z)

r = 8*np.sin(x)
print("r =", r)

s = 5*np.sin(x*y)
print("s =", s)

p = x**y
print("p = ", p)
```

```
x = [1 4 3 2 9 4]
y = [2 3 4 1 2 3]
x + y =  [ 3  7  7  3 11  7]
v = [ 2 12 12  2 18 12]
w = [0.5        1.33333333 0.75       2.         4.5        1.33333333]
z = [ 0.84147098 -0.7568025   0.14112001  0.90929743  0.41211849 -0.7568025 ]
r = [ 6.73176788 -6.05441996  1.12896006  7.27437941  3.29694788 -6.05441996]
s = [ 4.54648713 -2.68286459 -2.68286459  4.54648713 -3.75493623 -2.68286459]
p =  [ 1 64 81  2 81 64]
```

In [5]:

```
#problem 20

x = np.linspace(-10,10,100)
print(x)
```

```
[-10.          -9.7979798   -9.5959596   -9.39393939  -9.19191919
  -8.98989899  -8.78787879  -8.58585859  -8.38383838  -8.18181818
  -7.97979798  -7.77777778  -7.57575758  -7.37373737  -7.17171717
  -6.96969697  -6.76767677  -6.56565657  -6.36363636  -6.16161616
  -5.95959596  -5.75757576  -5.55555556  -5.35353535  -5.15151515
  -4.94949495  -4.74747475  -4.54545455  -4.34343434  -4.14141414
  -3.93939394  -3.73737374  -3.53535354  -3.33333333  -3.13131313
  -2.92929293  -2.72727273  -2.52525253  -2.32323232  -2.12121212
  -1.91919192  -1.71717172  -1.51515152  -1.31313131  -1.11111111
  -0.90909091  -0.70707071  -0.50505051  -0.3030303   -0.1010101
   0.1010101    0.3030303    0.50505051   0.70707071   0.90909091
   1.11111111   1.31313131   1.51515152   1.71717172   1.91919192
   2.12121212   2.32323232   2.52525253   2.72727273   2.92929293
   3.13131313   3.33333333   3.53535354   3.73737374   3.93939394
   4.14141414   4.34343434   4.54545455   4.74747475   4.94949495
   5.15151515   5.35353535   5.55555556   5.75757576   5.95959596
   6.16161616   6.36363636   6.56565657   6.76767677   6.96969697
   7.17171717   7.37373737   7.57575758   7.77777778   7.97979798
   8.18181818   8.38383838   8.58585859   8.78787879   8.98989899
   9.19191919   9.39393939   9.5959596    9.7979798   10.        ]
```

In [9]:

```
#problem 21

import numpy as np

array_a = np.array([-1, 0, 1, 2, 0, 3])
print("Indexed array a is =", array_a[array_a>0])
```

```
Indexed array a is = [1 2 3]
```

In [14]:

```
#problem 22

import numpy as np
from math import *

y = np.array([[3,5,3], [2,2,5], [3,8,9]])
print(f"Matrix y =\n{y}")
print(f"transpose of y =\n {np.transpose(y)}")
```

```
Matrix y =
[[3 5 3]
 [2 2 5]
 [3 8 9]]
```

In [4]:

```
#problem 23

import numpy as np

y = np.zeros((2,4))
print("The zero matrix =\n", y)
```

```
The zero matrix =
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

In [8]:

```
#problem 24

import numpy as np
y = np.zeros((2,4))
y[0,1] = 1
y[1,1] = 1
print("The zero matrix =\n", y)
```

```
The zero matrix =
 [[0. 1. 0. 0.]
 [0. 1. 0. 0.]]
```

In [10]:

```
#problem 25

%reset
```

In [ ]:

```

```