

```
In [1]: #Basic Data Types
x = 3
print(type(x)) # Prints "<class 'int'>"
print(x)       # Prints "3"
print(x + 1)    # Addition; prints "4"
print(x - 1)    # Subtraction; prints "2"
print(x * 2)    # Multiplication; prints "6"
print(x ** 2)   # Exponentiation; prints "9"
x += 1
print(x) # Prints "4"
x -= 2
print(x) # Prints "8"
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"

<class 'int'>
3
4
2
6
9
4
8
<class 'float'>
2.5 3.5 5.0 6.25
```

```
In [2]: #Logical Operator
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

```

In [4]:
#String Operations
hello = 'hello' # String literals can use single quotes
world = 'world' # or double quotes; it does not matter.
print(hello)    # Prints "hello"
print(len(hello)) # String length; prints "5"
hw = 'hello + ' + world # String concatenation

```

```
print(hw) # prints "hello world"
hw2 = "%s %s %s" % (hello, world, 12) # sprintf style string formatting
print(hw2) # prints "hello world 12"

s = "hello"
print(s.capitalize()) # Capitalize a string; prints "Hello"
print(s.upper()) # Convert a string to uppercase; prints "HELLO"
print(s.rstrip()) # Right-justify a string, padding with spaces; prints "hello "
print(s.center(7)) # Center a string, padding with spaces; prints "hello "
print(s.replace('l', 'ell')) # Replace all instances of one substring with another;
                             # prints "he(ell)l(ell)l"

print(' ' * 100) # Strip leading and trailing whitespace; prints "world"
```

```
hello
5
hello world
hello world 12
hello
HELLO
    hello
hello
he(ell)(ell)o
world
```

```
xs = [3, 1, 2]      # Create a list
print(xs, xs[2])    # Prints "[3, 1, 2] 2"
print(xs[-1])        # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'        # Lists can contain elements of different types
print(xs)            # Prints "[3, 1, 'foo']"
xs.append('bar')      # Add a new element to the end of the list
print(xs)            # Prints "[3, 1, 'foo', 'bar']"
x = xs.pop()         # Remove and return the last element of the list
print(x, xs)         # Prints "bar [3, 1, 'foo']"
```

```
[3, 1, 'foo', 'bar']
bar [3, 1, 'foo']

In [6]: #slicing
        nums = list(range(5))           # range is a built-in function that creates a list of integers
        print(nums)                    # Prints "[0, 1, 2, 3, 4]"
        print(nums[2:4])                # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
        print(nums[:2])                 # Get a slice from index 0 to 2 (exclusive); prints "[0, 1]"
        print(nums[2:])                 # Get a slice from the start to index 2 (exclusive); prints "[2, 3, 4]"
        print(nums[:1])                 # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
        print(nums[-1])                 # Slice indices can be negative; prints "[4]"
        nums[2:4] = [8, 9]              # Assign a new sublist to a slice
```

```
print(nums)           # Prints "[0, 1, 8, 9, 4]"

[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3, 4]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]

In [7]: #Loop
        #For Loop
animals = ['cat', 'dog', 'monkey']
```

```
for animal in animals:
    print(animal)
#2
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))

cat
dog
monkey
#1: cat
#2: dog
#3: monkey
```

```
[In: 9]: #List Comprehension
#Normal Method
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)    # Prints [0, 1, 4, 9, 16]

#Easy Shorten method
square = [x ** 2 for x in nums]
print(square)    # Prints [0, 1, 4, 9, 16]
```

```
#Another Method
nums = [0, 1, 2, 3, 4]
even_squares = [x**2 for x in nums if x % 2 == 0]
print(even_squares) # Prints "[0, 4, 16]"

[0, 1, 4, 9, 16]
[0, 1, 4, 9, 16]
[0, 4, 16]

In [10]: #Array using Numpy
#Basic
import numpy as np
```

```
a = np.array([1, 2, 3]) # Create a rank 1 array
print(type(a))          # Prints "<class 'numpy.ndarray'>"
print(a.shape)          # Prints "(3,)"
print(a[0], a[1], a[2]) # Prints "1 2 3"
a[0] = 5                # Change an element of the array
print(a)                # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]]) # Create a rank 2 array
print(b.shape)                # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0]) # Prints "1 2 4"

<class 'numpy.ndarray'>
```

```
[5]:
1 2 3
[5 2 3]
(2, 3)
1 2 4

In [11]: #Array with Built-in Functions

import numpy as np

a = np.zeros((2,2)) # Create an array of all zeros
print(a) # Prints "[[ 0.  0.]
                    [ 0.  0.]]"
```

```
b = np.ones((1,2)) # Create an array of all ones
print(b)           # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print(c)             # Prints "[[ 7.  7.]
                    #         [ 7.  7.]]"

d = np.eye(2)        # Create a 2x2 identity matrix
print(d)             # Prints "[[ 1.  0.]
                    #         [ 0.  1.]]"
```

```
e = np.random.random((2,2)) # Create an array filled with random values
print(e)                    # Right print "[[ 0.91584667  0.68143941]
                             #           [ 0.68744134  0.87236687]]"
```

```
[[0. 0.]
 [0. 0.]
 [[1., 1.]
 [7 7]
 [7 7]
 [[1., 0.]
 [0. 1.]
 [0.52096678 0.46687676]
 [0.21895956 0.77106362]]
```

```
In [12]: #Using MatLab
#Plotting
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)

# Plot the points using matplotlib
plt.plot(x, y)
```

```
plt.show() # You must call plt.show() to make graphics appear.
```

The plot displays two sine waves. The x-axis represents time from 0 to 10, and the y-axis represents a value from 0.25 to 1.00. The red wave starts at approximately 0.25 at time 0, reaches a peak of 1.00 at time 2.5, and returns to 0.25 at time 5. The blue wave starts at approximately 0.25 at time 0, reaches a peak of 1.00 at time 7.5, and returns to 0.25 at time 10.

```
In [20]: import numpy as np
import matplotlib.pyplot as plt

#populate the data (x and y)
x1=np.array([0,8,15,9,11,14,6,4,12,7,5])
x4=np.array([8,8,8,8,8,8,19,8,8])
y1=np.array([8.04,6.95,7.58,8.81,8.33,9.96,7.24,4.26,10.84,4.82,5.68])
y2=np.array([0.14,8.14,8.74,8.77,9.26,8.5,5.13,3.1,9.13,7.26,4.74])
y3=np.array([7.46,6.77,12.74,7.11,7.81,8.84,6.08,5.39,8.15,6.42,5.73])
```

```
y4=np.array([6.56,5.76,7.13,8.84,8.47,7.84,5.25,12.5,5.56,7.91,6.89])
np.random.seed(1)
plt.scatter(x1,y1,c="blue")
plt.show()
plt.scatter(x1,y2,c="green")
plt.show()
plt.scatter(x1,y3,c="orange")
plt.show()
plt.scatter(x4,y4,c="red")
plt.show()
array = np.array([[np.mean(x1),np.std(x1),np.mean(y1),np.std(y1)],
                  [np.mean(x1),np.std(x1),np.mean(y2),np.std(y2)],
                  [np.mean(x1),np.std(x1),np.mean(y3),np.std(y3)],
                  [np.mean(x4),np.std(x4),np.mean(y4),np.std(y4)]])
```

```
print(np.mean(x4), np.std(x4), np.mean(y4), np.std(y4)])

print(array([
    pcc1 = np.corrcoef(x1,y1)
    pcc2 = np.corrcoef(x1,y1)
    pcc3 = np.corrcoef(x1,y1)
    pcc4 = np.corrcoef(x1,y1)
    print(pcc1, pcc2, pcc3, pcc4)
])
```

Year of birth	Number of children per woman
1950	7.2
1955	7.0
1960	8.8
1965	8.1
1970	8.3
1975	7.6
1980	10.0

The top scatter plot shows a negative correlation between the number of children and the number of books read. The data points are blue dots at approximately (4, 4.2), (5, 5.8), and (7, 4.9).

The bottom scatter plot shows a positive correlation between the number of children and the number of books read. The data points are green dots at approximately (10, 0.1), (11, 0.3), and (12, 0.1).

The top scatter plot shows a positive correlation between the number of children and the number of books read. The x-axis represents the number of children (4 to 14) and the y-axis represents the number of books read (3 to 5). Two green dots are plotted at (4, 3) and (5, 4.5).

The bottom scatter plot shows a negative correlation between the number of children and the number of books read. The x-axis represents the number of children (4 to 14) and the y-axis represents the number of books read (13 to 14). One orange dot is plotted at (13, 13).

A scatter plot showing the relationship between the year of birth (X-axis) and the number of children per woman (Y-axis). The X-axis ranges from 4 to 14, and the Y-axis ranges from 6 to 8. The data points are as follows:

Year of Birth	Number of Children per Woman
4	5.5
5	5.8
6	6.1
7	6.5
8	6.8
9	7.1
10	7.5
11	7.8
12	8.2


[illegible]

```
[[0. 3.16227766 7.50090909 1.03762422]
 [0. 3.16227766 7.50090909 1.03710869]
 [0. 3.16227766 7.5 1.03593294]
 [0. 3.16227766 7.50090909 1.03608651]
 [1. 0.81642052]
 [0.81642052 1. ]] [[1. 0.81642052]
 [0.81642052 1. ]] [[1. 0.81642052]
 [0.81642052 1. ]] [[1. 0.81642052]
 [0.81642052 1. ]]


In [4]: import numpy as np
import matplotlib.pyplot as plt
```

```
# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
```



A plot showing a sine wave. The x-axis is labeled 'x axis label' and the y-axis is labeled 'y axis label'. The y-axis has tick marks at 0.25, 0.00, -0.25, -0.50, and -0.75. The plot contains a single blue line representing a sine wave. A legend in the bottom-left corner shows a blue line segment followed by the text 'Sine'.



The plot displays a cosine wave. The x-axis is labeled 'x axis label' and has major ticks at 0, 2, 4, 6, and 8. The y-axis has a major tick at -1.00. A legend in the top-left corner identifies the orange line as 'Cosine'.

```
In [2]: import numpy as np
import matplotlib.pyplot as plt

x=["Data Visualization","Database Systems","Computer Networks","Machine Vision"]
y=[50,55,60,65]
x1=["DV"]
```

```
x2=["db"]
x3=["CW"]
x4=["MV"]


y1=[50]
y2=[55]
y3=[60]
y4=[65]

plt.scatter(x1,y1,label="SET-1",c="red")
plt.scatter(x2,y2,label="SET-2",c="blue")
plt.scatter(x3,y3,label="SET-3",c="orange")
plt.scatter(x4,y4,label="SET-4",c="green")
plt.xlabel("Microbial diversity")
```

```
plt.scatter(x, y, c=category, s=area, alpha=0.5);
plt.xlabel("X-Axis");
plt.ylabel("Y-Axis");
plt.legend();
plt.show();
```

Anscombe's Quartet

SET	X	Y
SET-1	1	64.1
SET-1	5	63.5
SET-1	8	69.3
SET-1	8	68.7
SET-1	17	71.1
SET-1	17	72.6
SET-1	27	70.8
SET-1	27	73.4
SET-1	36	69.6
SET-1	36	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1	41	73.4
SET-1	41	69.6
SET-1	41	70.9
SET-1	41	71.1
SET-1	41	72.6
SET-1	41	70.8
SET-1		



X-Axis	Value
DV	50.0


```
In [3]: import numpy as np
import matplotlib.pyplot as plt
```

```
x=["Data Visualization", "Database Systems", "Computer Networks", "Machine Vision"]
y=[50,55,60,65]

plt.bar(x,y)
plt.show()
```

Category	Value
Data Visualization	50
Database Systems	55
Computer Networks	60
Machine Vision	65

Category	Percentage
The current government	48
The previous government	45
The international community	42
The economy itself	38



Topic	Count
Data Visualization	2
Database Systems	3
Computer Networks	4
Machine Vision	4

```
In [6]: #Slicing
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[1 2 3 4]
```

```
# [ 5 6 7 8]
# [ 9 10 11 12]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[1:3, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
```

```
print(a[0, 1]) # Prints
b[0, 0] = 77 # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1]) # Prints "77"

2
77

In [5]: #slicing2
#Slicing Examples
#nums = list(range(5)) # range is a built-in function that creates a list of integers
nums=[0,1,2,3,4,5,6]
print(nums)
# Get a slice from index 0 to 4 (exclusive); prints "[0, 1, 2, 3, 4]"
print(nums[0:4])
# Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
```

```
print(nums[2:]) # Get a slice from index 2 to the end; prints "[2, 3, 4, 5, 6]"
print(nums[:2]) # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print(nums[:]) # Get a slice of the whole list; prints "[0, 1, 2, 3, 4, 5, 6]"
print(nums[-1:]) # Slice indices can be negative; prints "[6, 1, 2, 3, 4, 5, 6]"
nums[2:4] = [8, 9] # Assign a new sublist to a slice
print(nums)

[0, 1, 2, 3, 4, 5, 6]
[2, 3]
[2, 3, 4, 5, 6]
[0, 1]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6]
```