IE416 : Robot Programming

Lab 1 : Introduction to Python

Submitted by

Group 20 - Robo Techs

Smeet Agrawal - 202101237

Priyesh Tandel - 202101222

6^{th} Semester, B.Tech

Submitted to

Professor Tapas Kumar Maiti

Dhirubhai Ambani Institute of Information and Communication Technology

Gandhinagar , Gujarat

Basic data types:

```
# -------------------- Code ----------------------------
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 6.2
5"
# ----------------------Output--------------------------

3 <class 'int'>
4
2
6
9
9
18
<class 'float'>
2.5 3.5 5.0 6.25
```

Booleans:

```
# -------------------- Code ---------------------------
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"

# ----------------------Output--------------------------
<class 'bool'>
False
True
False
True
```

Strings:

```
# -------------------- Code ----------------------------
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"
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style string formattin
print(hw12)  # prints "hello world 12"

# ----------------------Output--------------------------
hello
5
hello world
hello world 12


```

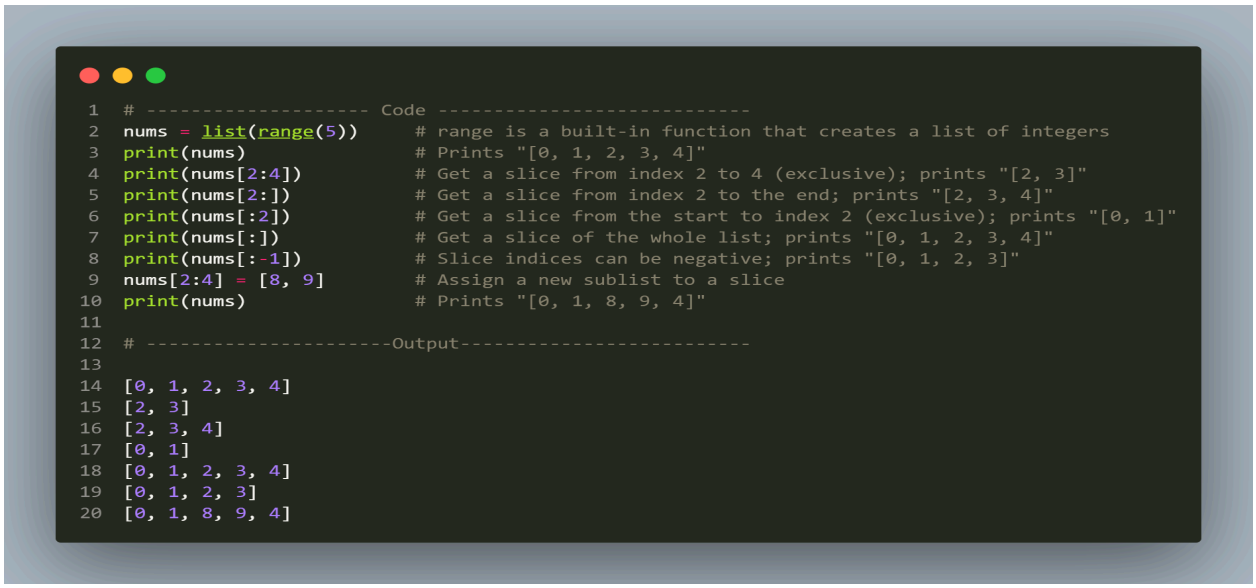
Operations on strings:

```
# -------------------- Code --------------------------
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # 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 anothe
r;                              # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace; prints "world"

# ---------------------Output-------------------------

Hello
HELLO
  hello
 hello
he(ell)(ell)o
world

```

Containers:

1) List :

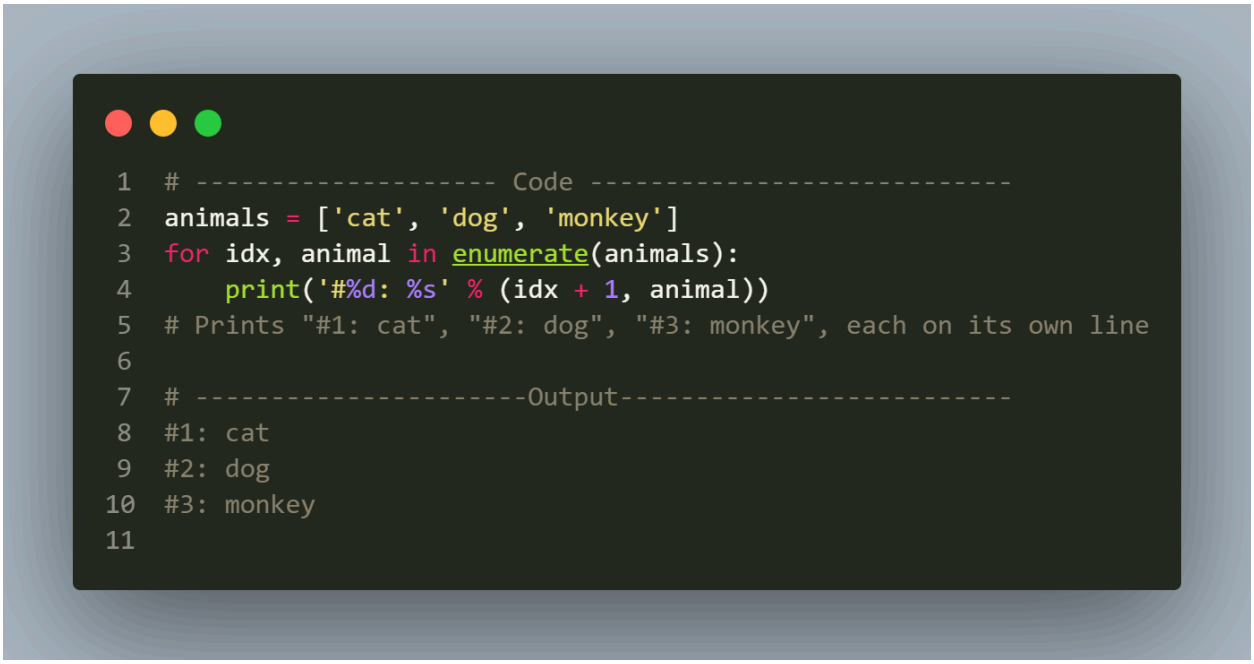
```python
# -------------------- Code ---------------------------
xs = [3, 1, 2]    # Create a list
print(xs, xs[2])  # Prints "[3, 1, 2] 2"
print(xs[-1])     # Negative indices count from the end of the list; prints "2"
xs[2] = 'foo'     # Lists can contain elements of different types
print(xs)         # 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']"

# ----------------------Output-------------------------

[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
bar [3, 1, 'foo']
```



```
# -------------------- Code --------------------------
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 2 to the end; prints "[2, 3, 4]"
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]"
print(nums[:-1])           # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]         # Assign a new sublist to a slice
print(nums)                # Prints "[0, 1, 8, 9, 4]"

# ----------------------Output-------------------------

[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]
```

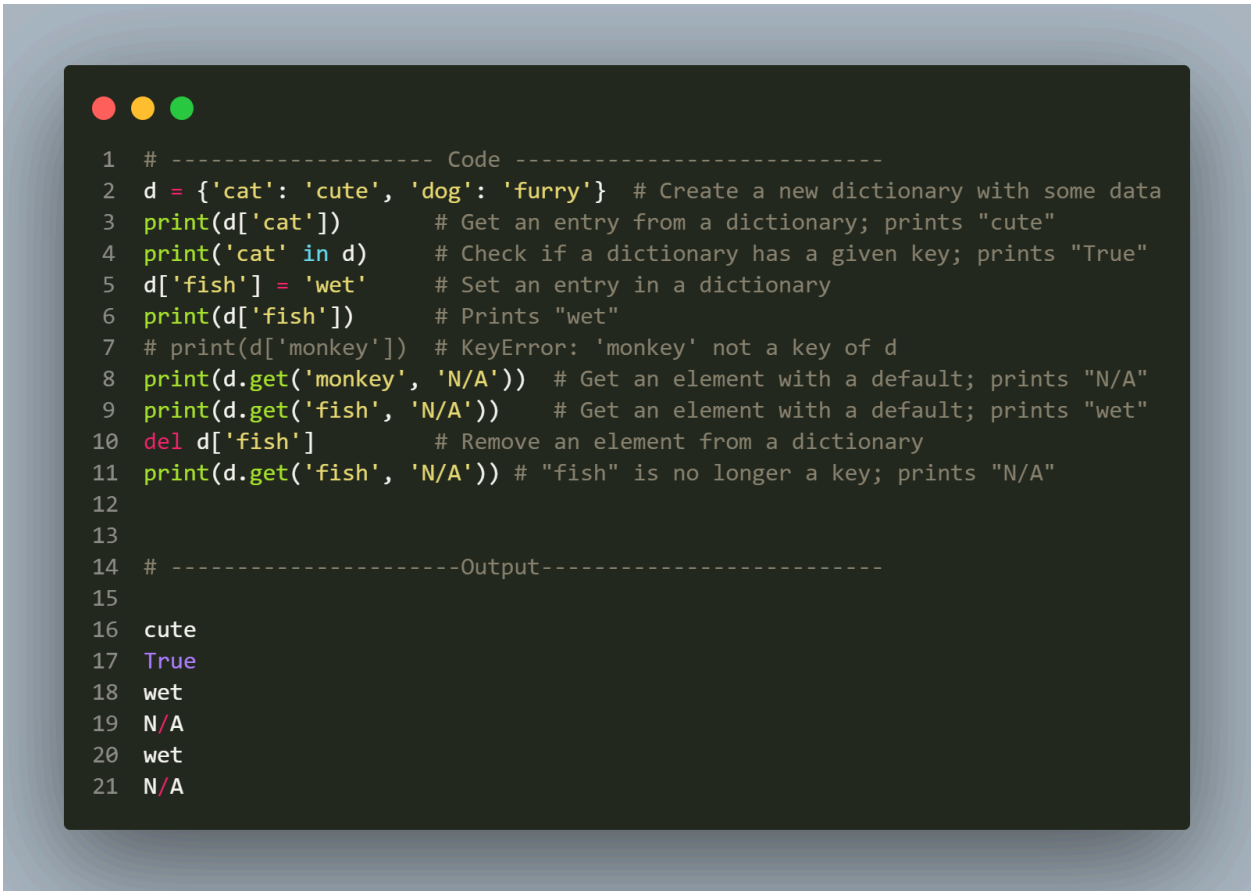
Loops:

```
# -------------------- Code ----------------------------
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
# Prints "cat", "dog", "monkey", each on its own line.

# ----------------------Output-------------------------
cat
dog
monkey

```



```
# -------------------- Code ----------------------------
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line

# ----------------------Output--------------------------
#1: cat
#2: dog
#3: monkey

```

List Comprehension:

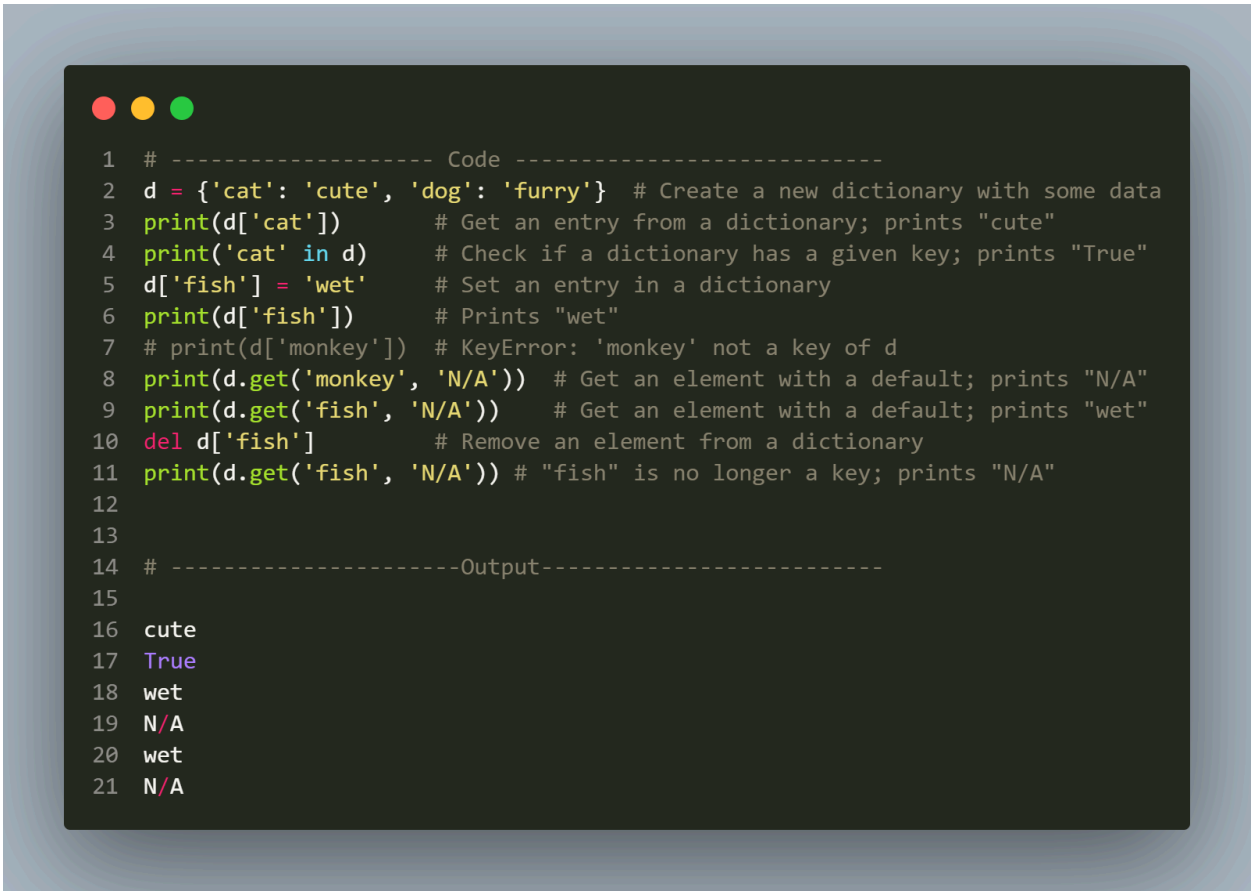


```
# -------------------- Code --------------------------
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
del d['fish']         # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"


# ---------------------Output-------------------------

cute
True
wet
N/A
wet
N/A
```

```python
# -------------------- Code ----------------------------
nums = [0, 1, 2, 3, 4]
squares = []
for x in nums:
    squares.append(x ** 2)
print(squares)   # Prints [0, 1, 4, 9, 16]


# ----------------------Output---------------------------

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

```python
# -------------------- Code ----------------------------
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]"


# ----------------------Output---------------------------

[0, 4, 16]
```

Dictionaries:



```
# -------------------- Code --------------------------
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
del d['fish']         # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"


# ----------------------Output-------------------------

cute
True
wet
N/A
wet
N/A
```

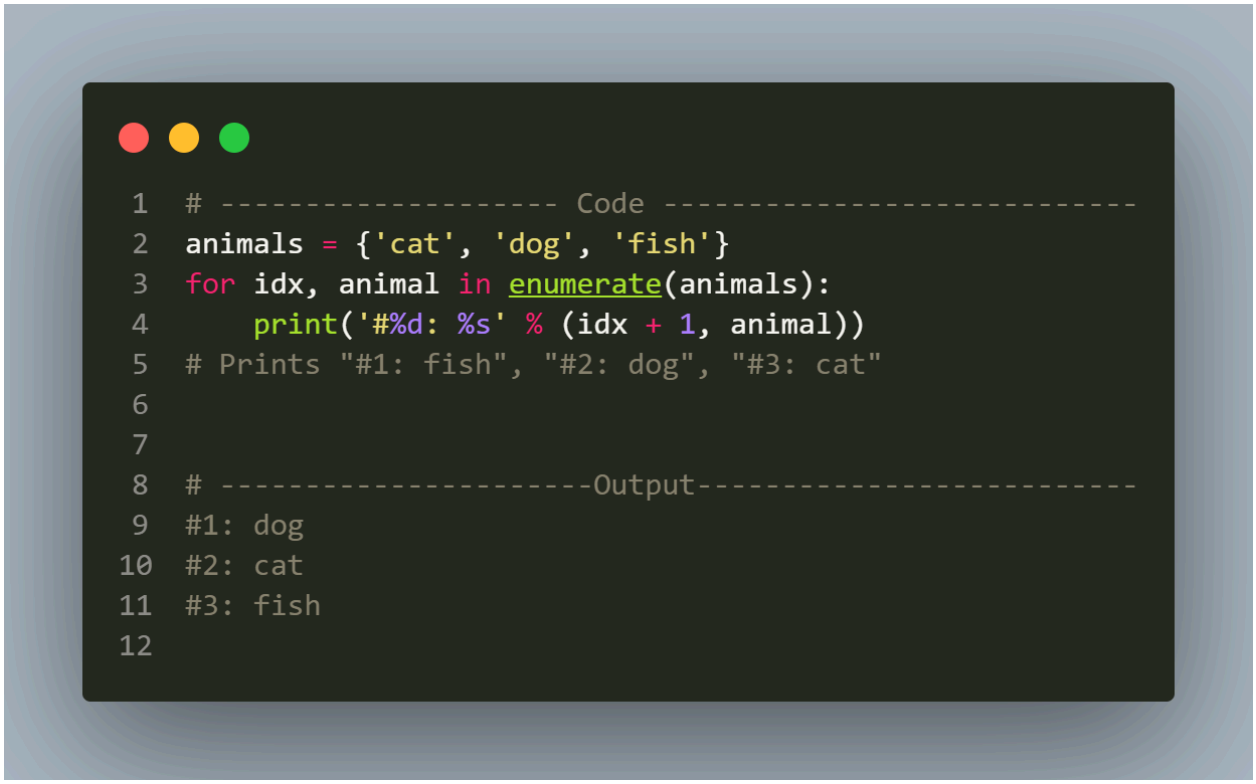
```
# -------------------- Code ----------------------------
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal in d:
    legs = d[animal]
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"


# ----------------------Output--------------------------

A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

```
# -------------------- Code ----------------------------
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"


# ----------------------Output--------------------------

A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

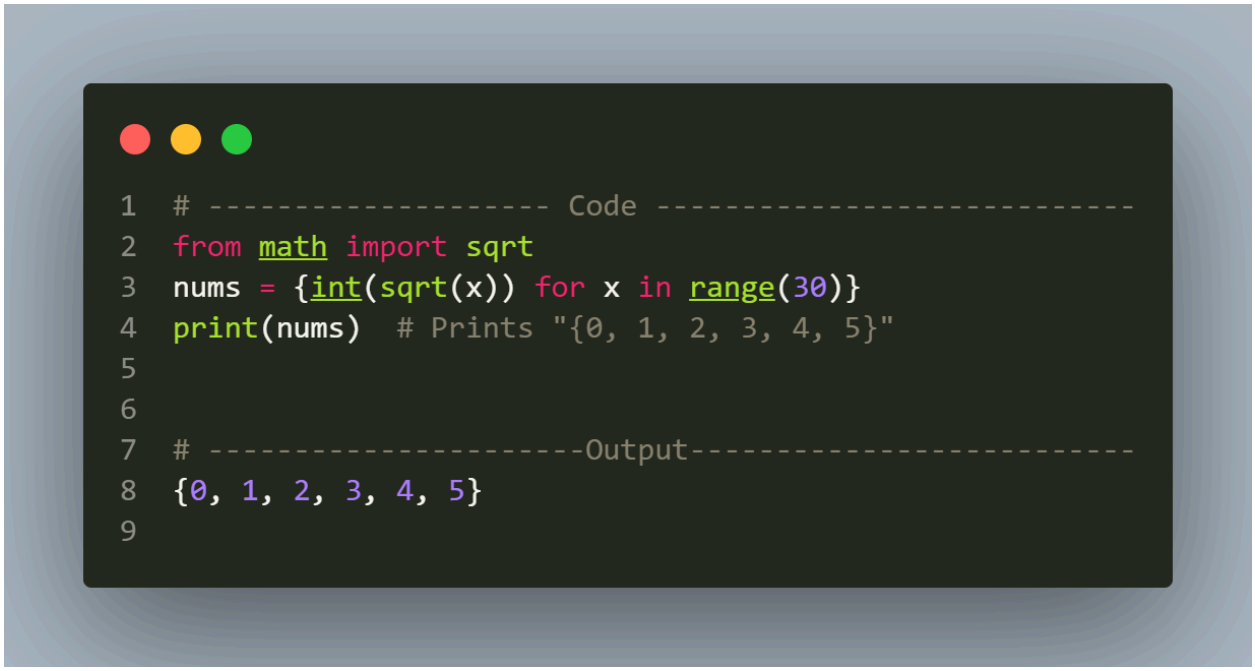
```python
# -------------------- Code -----------------------------
nums = [0, 1, 2, 3, 4]
even_num_to_square = {x: x ** 2 for x in nums if x % 2 == 0}
print(even_num_to_square)  # Prints "{0: 0, 2: 4, 4: 16}"


# ----------------------Output--------------------------

{0: 0, 2: 4, 4: 16}
```

Sets:

```python
# -------------------- Code --------------------------
animals = {'cat', 'dog'}
print('cat' in animals)   # Check if an element is in a set; prints "True"
print('fish' in animals)  # prints "False"
animals.add('fish')       # Add an element to a set
print('fish' in animals)  # Prints "True"
print(len(animals))       # Number of elements in a set; prints "3"
animals.add('cat')        # Adding an element that is already in the set does nothing
print(len(animals))       # Prints "3"
animals.remove('cat')     # Remove an element from a set
print(len(animals))       # Prints "2"


# ----------------------Output-------------------------

True
False
True
3
3
2
```

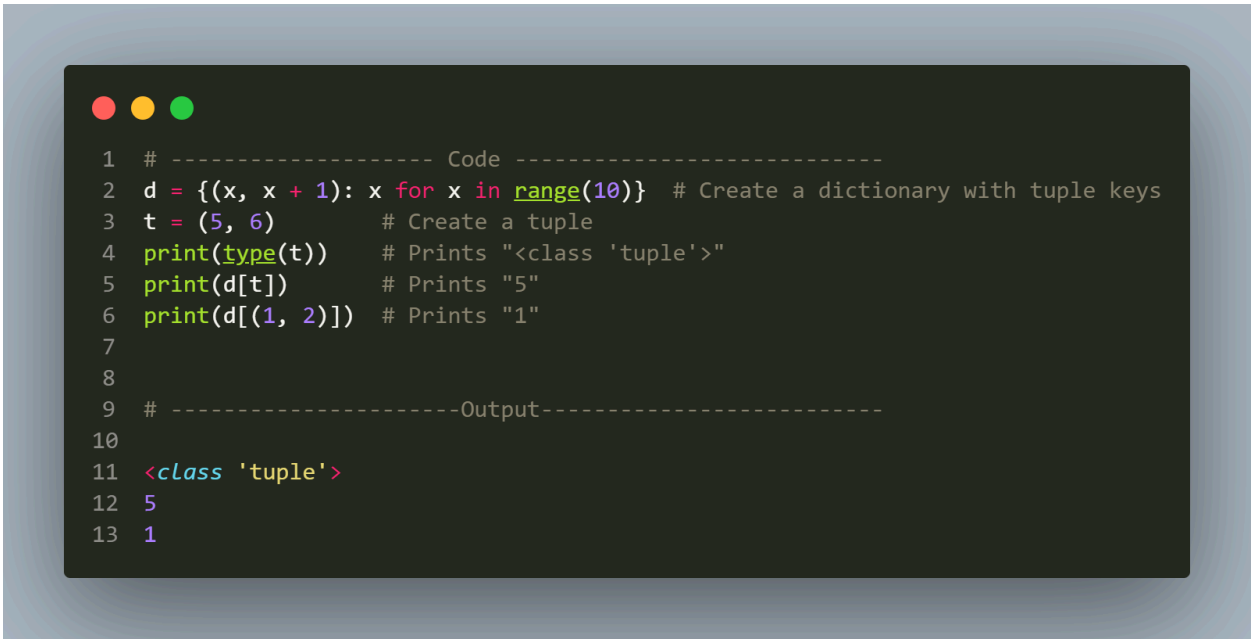


```
# -------------------- Code ----------------------------
animals = {'cat', 'dog', 'fish'}
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: fish", "#2: dog", "#3: cat"


# ----------------------Output--------------------------
#1: dog
#2: cat
#3: fish

```



```
# -------------------- Code ----------------------------
from math import sqrt
nums = {int(sqrt(x)) for x in range(30)}
print(nums)  # Prints "{0, 1, 2, 3, 4, 5}"


# ----------------------Output--------------------------
{0, 1, 2, 3, 4, 5}

```

Tuples



```
# -------------------- Code ---------------------------
d = {(x, x + 1): x for x in range(10)}  # Create a dictionary with tuple keys
t = (5, 6)        # Create a tuple
print(type(t))    # Prints "<class 'tuple'>"
print(d[t])       # Prints "5"
print(d[(1, 2)])  # Prints "1"


# ----------------------Output-------------------------

<class 'tuple'>
5
1
```

Functions

```
# -------------------- Code ---------------------------
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

for x in [-1, 0, 1]:
    print(sign(x))
# Prints "negative", "zero", "positive"

# ----------------------Output-------------------------

negative
zero
positive
```


# -------------------- Code ----------------------------
def hello(name, loud=False):
    if loud:
        print('HELLO, %s!' % name.upper())
    else:
        print('Hello, %s' % name)

hello('Bob') # Prints "Hello, Bob"
hello('Fred', loud=True) # Prints "HELLO, FRED!"
----------------------Output--------------------------

Hello, Bob
HELLO, FRED!

Classes:

```
# -------------------- Code ---------------------------
class Greeter(object):

    # Constructor
    def __init__(self, name):
        self.name = name  # Create an instance variable

    # Instance method
    def greet(self, loud=False):
        if loud:
            print('HELLO, %s!' % self.name.upper())
        else:
            print('Hello, %s' % self.name)

g = Greeter('Fred')  # Construct an instance of the Greeter class
g.greet()            # Call an instance method; prints "Hello, Fred"
g.greet(loud=True)   # Call an instance method; prints "HELLO, FRED!"
# ----------------------Output--------------------------

Hello, Fred
HELLO, FRED!
```

Numpy

Arrays :

```
# -------------------- Code --------------------

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"

''' ----------------------Output-------------------------

<class 'numpy.ndarray'>
(3,)
1 2 3
[5 2 3]
(2, 3)
1 2 4

'''

```

Array indexing - `Slicing`

```python
# -------------------- Code -------------------

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)                     # Might print "[[ 0.91940167  0.08143941]
                             #              [ 0.68744134  0.87236687]]"

''' ----------------------Output------------------------

[[0. 0.]
 [0. 0.]]
[[1. 1.]]
[[7 7]
 [7 7]]
[[1. 0.]
 [0. 1.]]
[[0.76904903 0.91551532]
 [0.03519157 0.04927133]]

'''
```

```python
# -------------------- Code --------------------

import numpy as np

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

b = a[:2, 1:3]

print(a[0, 1])
b[0, 0] = 77
print(a[0, 1])

''' ----------------------Output-------------------------

2
77

'''

```

```python
# -------------------- Code --------------------

import numpy as np

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


# original array:
row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
print(row_r1, row_r1.shape)  # Prints "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape)  # Prints "[[5 6 7 8]] (1, 4)"

# We can make the same distinction when accessing columns of an array:
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)  # Prints "[ 2  6 10] (3,)"
print(col_r2, col_r2.shape)  # Prints "[[ 2]


''' ----------------------Output-------------------------

[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)

'''

```

```python
# -------------------- Code --------------------

import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

# An example of integer array indexing.
# The returned array will have shape (3,) and
print(a[[0, 1, 2], [0, 1, 0]])  # Prints "[1 4 5]"

# The above example of integer array indexing is equivalent to this:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))  # Prints "[1 4 5]"

# When using integer array indexing, you can reuse the same
# element from the source array:
print(a[[0, 0], [1, 1]])  # Prints "[2 2]"

# Equivalent to the previous integer array indexing example
print(np.array([a[0, 1], a[0, 1]]))  # Prints "[2 2]"

''' ----------------------Output-------------------------

[1 4 5]
[1 4 5]
[2 2]
[2 2]

'''

```

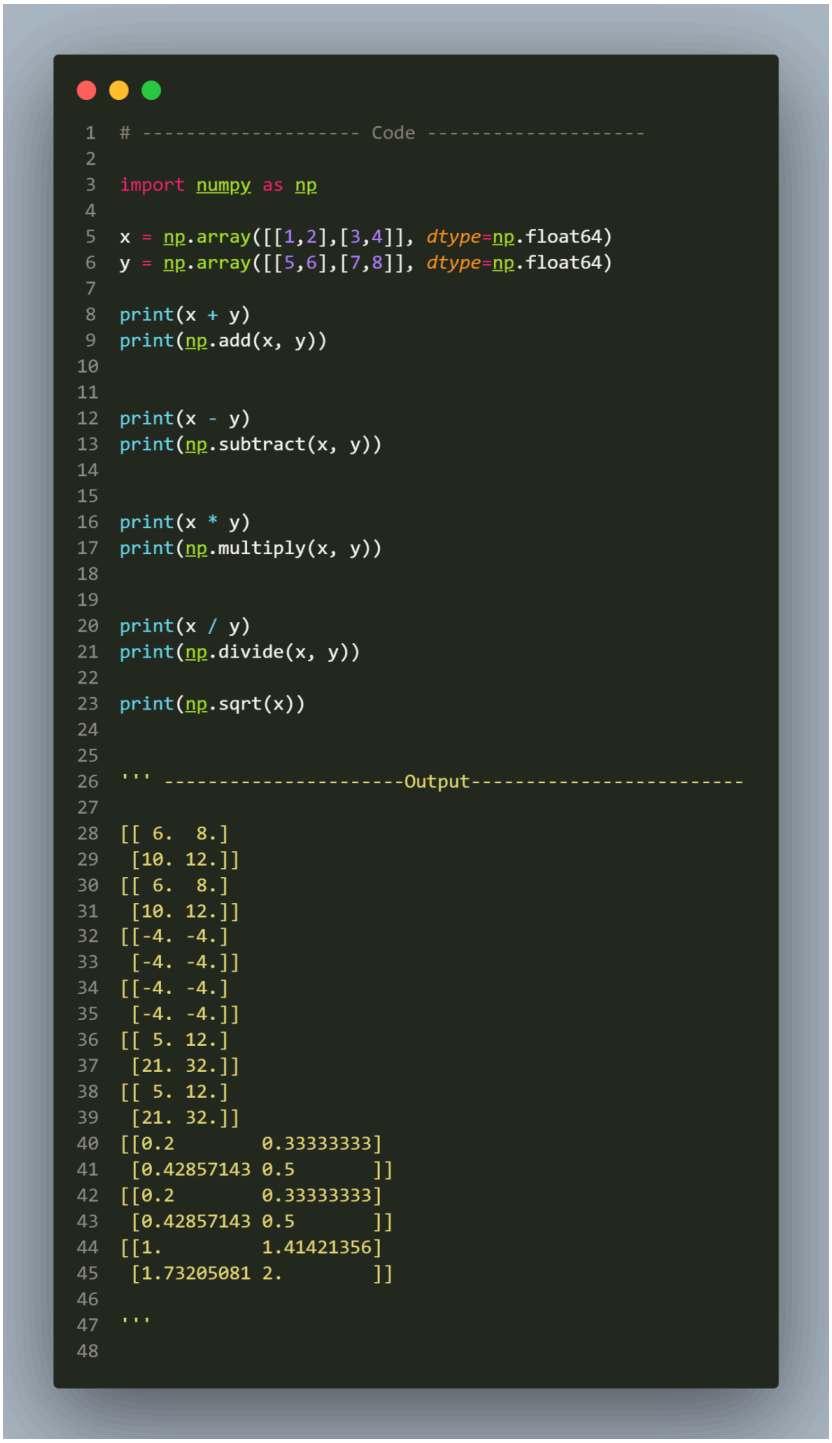
```python
# -------------------- Code --------------------

import numpy as np

import numpy as np

# Create a new array from which we will select elements
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])

print(a)

# Create an array of indices
b = np.array([0, 2, 0, 1])

# Select one element from each row of a using the indices in b
print(a[np.arange(4), b])  # Prints "[ 1  6  7 11]"

# Mutate one element from each row of a using the indices in b
a[np.arange(4), b] += 10

print(a)

''' ----------------------Output-------------------------

[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[ 1  6  7 11]
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]

'''

```

```python
# -------------------- Code --------------------

import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])

bool_idx = (a > 2)

print(bool_idx)

print(a[bool_idx])  # Prints "[3 4 5 6]"

print(a[a > 2])     # Prints "[3 4 5 6]"

''' ----------------------Output------------------------

[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
[3 4 5 6]

'''

```

Datatypes:

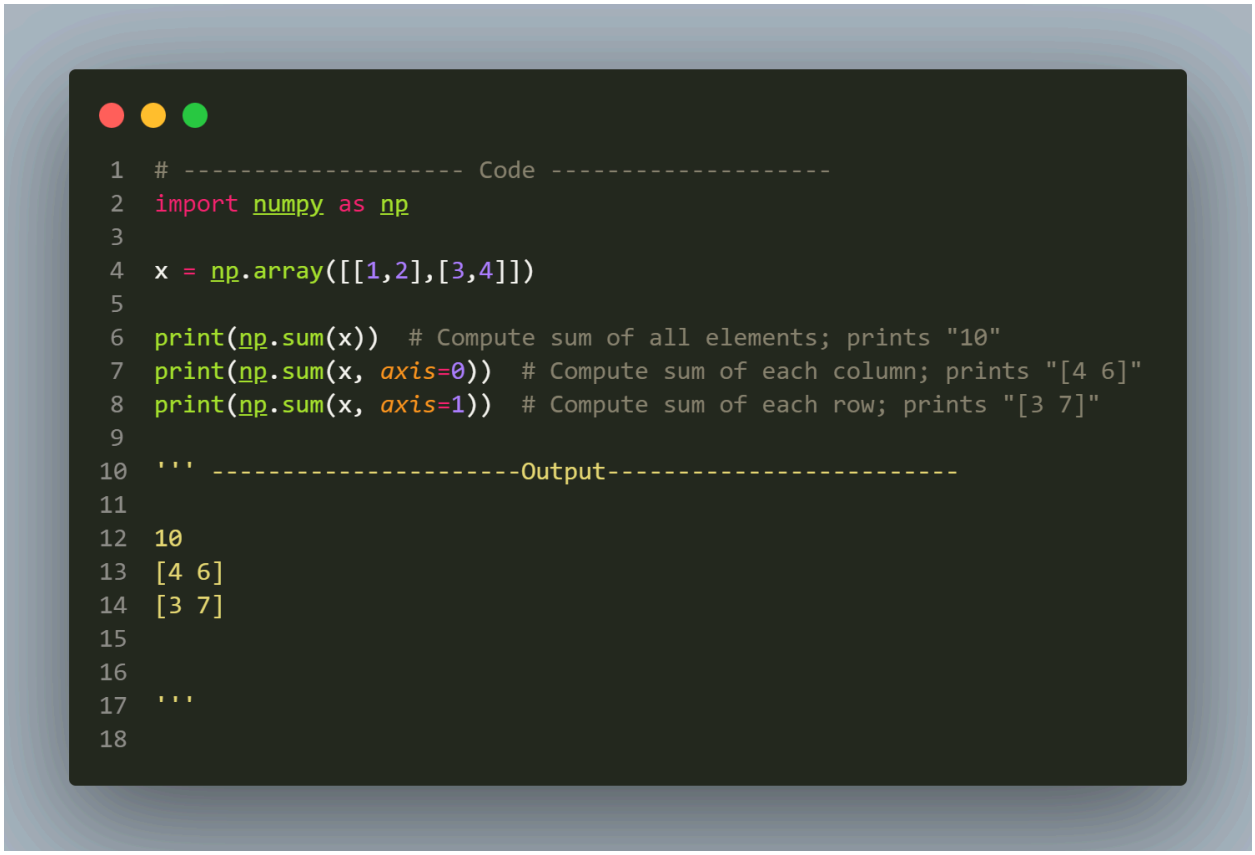
```python
# -------------------- Code --------------------

import numpy as np

x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
print(x.dtype)                         # Prints "int64"

''' ----------------------Output------------------------

int32
float64
int64

'''

```

Array math



```python
# -------------------- Code --------------------

import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

print(x + y)
print(np.add(x, y))


print(x - y)
print(np.subtract(x, y))


print(x * y)
print(np.multiply(x, y))


print(x / y)
print(np.divide(x, y))

print(np.sqrt(x))


''' ----------------------Output-------------------------

[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[0.2        0.33333333]
 [0.42857143 0.5       ]]
[[1.         1.41421356]
 [1.73205081 2.        ]]

'''
```

```python
# -------------------- Code --------------------

import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))

# Matrix / vector product; both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))


print(x.dot(y))
print(np.dot(x, y))


''' ----------------------Output-------------------------

219
219
[29 67]
[29 67]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]


'''

```



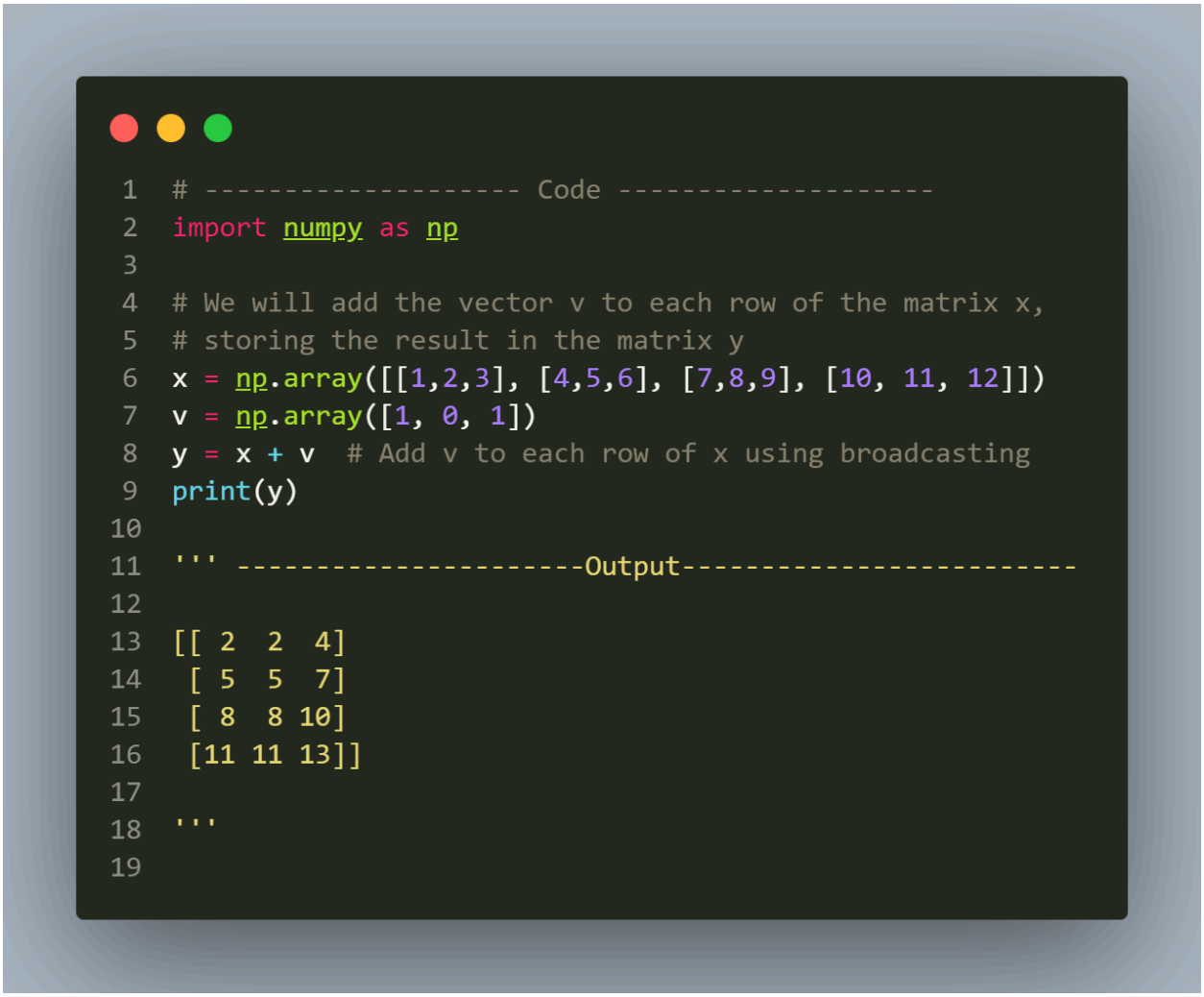
```
# -------------------- Code --------------------
import numpy as np

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

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"

''' ----------------------Output-------------------------

10
[4 6]
[3 7]


'''

```

```python
# -------------------- Code --------------------
import numpy as np

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

# Note that taking the transpose of a rank 1 array does nothing:
v = np.array([1,2,3])
print(v)    # Prints "[1 2 3]"
print(v.T)  # Prints "[1 2 3]"

''' ----------------------Output-------------------------

[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
[1 2 3]
[1 2 3]


'''

```

```python
# -------------------- Code --------------------
import numpy as np

x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = np.empty_like(x)   # Create an empty matrix with the same shape as x

# Add the vector v to each row of the matrix x with an explicit loop
for i in range(4):
    y[i, :] = x[i, :] + v


print(y)

''' ----------------------Output-------------------------

[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]


'''

```

Broadcasting

```python
# -------------------- Code --------------------
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
vv = np.tile(v, (4, 1))   # Stack 4 copies of v on top of each other
print(vv)
y = x + vv  # Add x and vv elementwise
print(y)

''' ----------------------Output-------------------------

[[1 0 1]
 [1 0 1]
 [1 0 1]
 [1 0 1]]
[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]


'''

```



```
# -------------------- Code --------------------
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v  # Add v to each row of x using broadcasting
print(y)

''' ----------------------Output-------------------------

[[ 2  2  4]
 [ 5  5  7]
 [ 8  8 10]
 [11 11 13]]

'''

```

```python
# -------------------- Code --------------------
import numpy as np

# Compute outer product of vectors
v = np.array([1,2,3])  # v has shape (3,)
w = np.array([4,5])    # w has shape (2,)
# To compute an outer product, we first reshape v to be a column
# vector of shape (3, 1); we can then broadcast it against w to yield
# an output of shape (3, 2), which is the outer product of v and w:
# [[ 4  5]
#  [ 8 10]
#  [12 15]]
print(np.reshape(v, (3, 1)) * w)

# Add a vector to each row of a matrix
x = np.array([[1,2,3], [4,5,6]])
# x has shape (2, 3) and v has shape (3,) so they broadcast to (2, 3),
# giving the following matrix:
# [[2 4 6]
#  [5 7 9]]
print(x + v)

# Add a vector to each column of a matrix
# x has shape (2, 3) and w has shape (2,).
# If we transpose x then it has shape (3, 2) and can be broadcast
# against w to yield a result of shape (3, 2); transposing this result
# yields the final result of shape (2, 3) which is the matrix x with
# the vector w added to each column. Gives the following matrix:
# [[ 5  6  7]
#  [ 9 10 11]]
print((x.T + w).T)

print(x + np.reshape(w, (2, 1)))


print(x * 2)

''' ----------------------Output-----------------------

[[ 4  5]
 [ 8 10]
 [12 15]]
[[2 4 6]
 [5 7 9]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 5  6  7]
 [ 9 10 11]]
[[ 2  4  6]
 [ 8 10 12]]

'''

```