multiverse

multiverse

Data Science Unit 1

Introduction to Python

Python

Python

pandas

SciPy

NumPy

matplotlib

multiverse

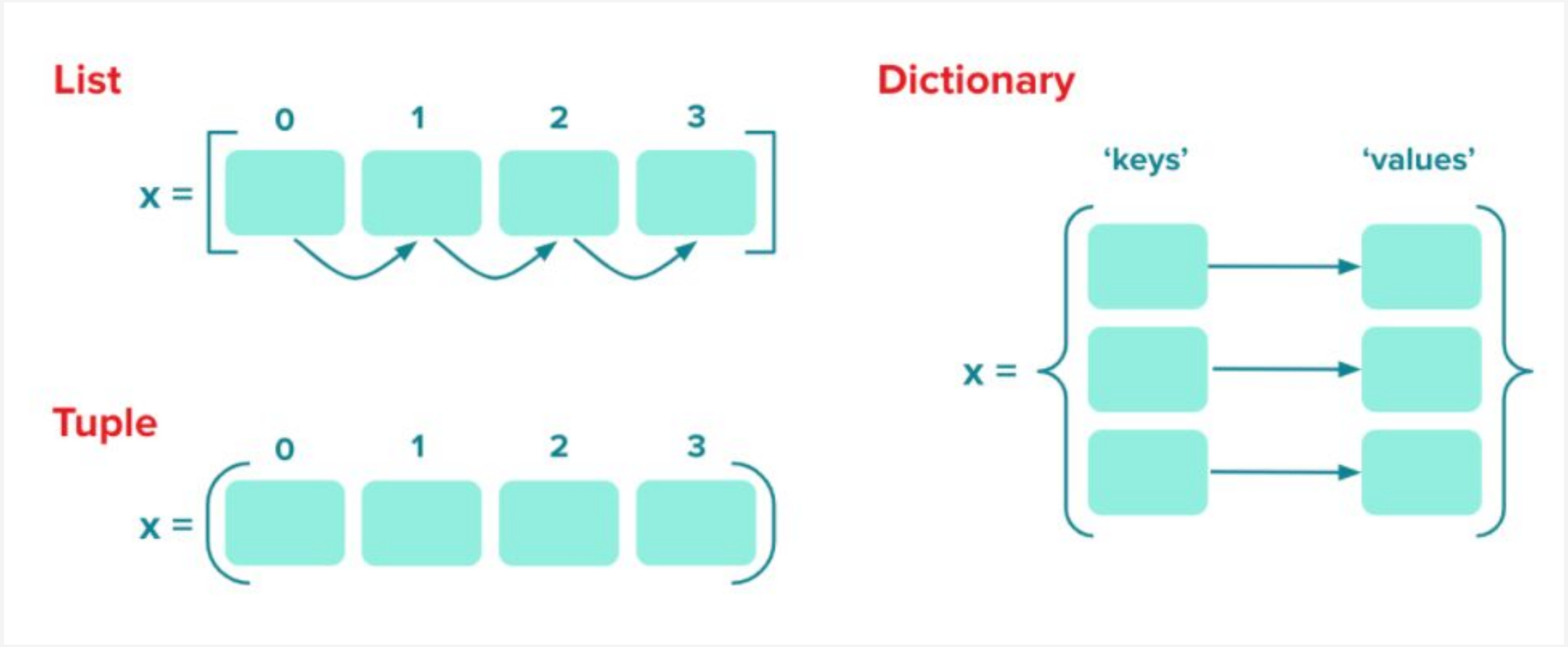
Data Science Unit 1

Data Types and Operators

Data Types

Data Type	Definition	Example
Integer	Whole numbers given from negative infinity to infinity	5, 3, -1, 1000
Float	‘Floating point number’ – has a decimal point in it	3.3, -2.4, 5.0
String	A set of letters, numbers or characters in general – surrounded by quotation marks	‘Data is Awesome’
Tuple	Ordered sequence with fixed number of elements- surrounded by parenthesis	(1,2), (‘Red’, ‘Green’, ‘Blue’)
List	Unordered sequence with no fixed number of elements- surrounded by square brackets	[1,2], [‘Red’, ‘Green’, ‘Blue]
Dictionary	Unordered collection of key value pairs. To access the value you need to use its key	{‘Blue’:5, ‘Red’ :2, ‘Green’:0}

Collections



Variables

Restrictions

- ➔ Variable names cannot contain spaces.
- ➔ A variable name must start with a letter or the underscore character.
- ➔ A variable name cannot start with a number.
- ➔ A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _)
- ➔ Variable names are case-sensitive (age, Age and AGE are three different variables)
- ➔ Variable names cannot be one of Python's Reserved Keywords

Best Practices

- ➔ Variable names should be lowercase.
- ➔ A variable's name should be representative of the value(s) it has been assigned, and informative of its content and use.
- ➔ If you must use multiple words in your variable name, use an underscore to separate them (snake_case)
- ➔ Variables should not be assigned the same name as a default or imported function or class (i.e., 'type', 'print', 'for'), or you risk overwriting these.

Operators

Operator	What it does	Example
+	Adds	1 + 1 = 2
-	Subtracts	3 - 2 = 1
*	Multiplies	4 * 4 = 16
/	Divides	5/2 = 2.5
//	Quotient (after division rounds down to whole number)	5//2 = 2
**	Exponent	3 ** 2 = 9
=	Assigns value	x = 2
%	Modulo (finds remainder)	5 % 2 = 1

Comparisons

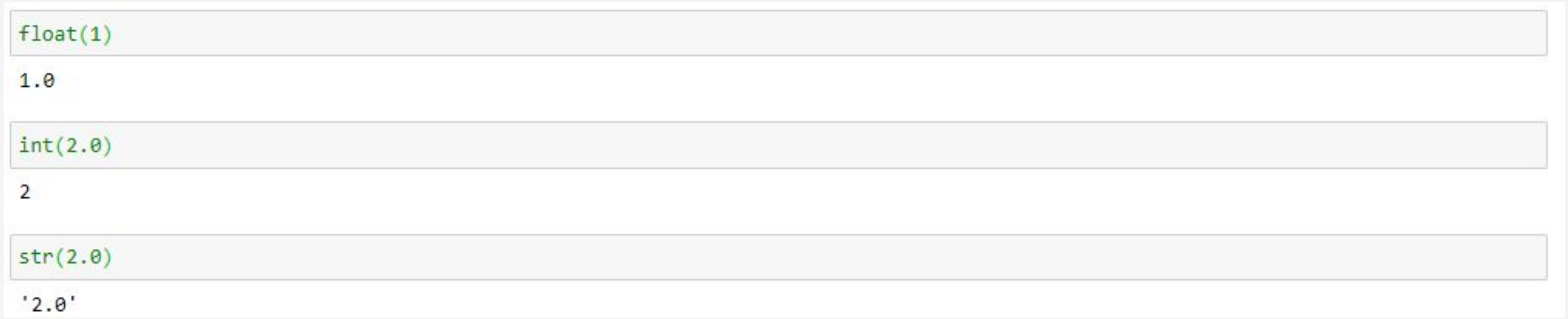
Operator	What it does
==	Equals to
!=	Not equals to
>	Greater than
>=	Greater than or equals to
<	Less than
<=	Less than or equals to

Booleans

$x = 2$

Boolean	Outcome
x == 2	True
x == 4	False
x == 2 and x == 4	False
x != 2	False
x == 2 or x == 4	True

Changing Types



```
float(1)
```

```
1.0
```

```
int(2.0)
```

```
2
```

```
str(2.0)
```

```
'2.0'
```

multiverse

Data Science Unit 1

Strings and Indexing

String

‘Hello World’

0 1 2 3 4 5 6 7 8 9 10

List

['Data', 1, 'London', 2.0]

multiverse

Data Science Unit 1

Collections

Tuples

(1,2,3)

Dictionary

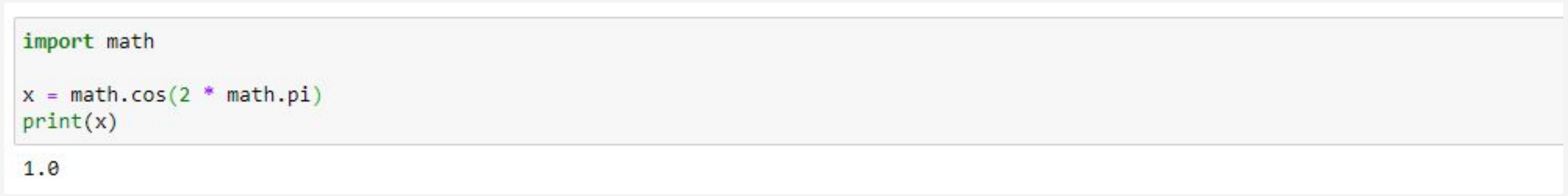
{'A':2, 'B':5, 'C':10}

multiverse

Data Science Unit 1

Importing Libraries

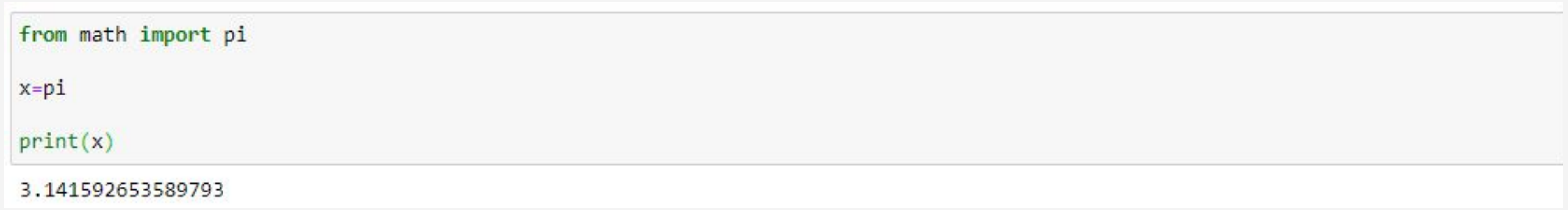
Import



```
import math

x = math.cos(2 * math.pi)
print(x)
```

```
1.0
```



```
from math import pi

x=pi

print(x)
```

```
3.141592653589793
```

multiverse

Data Science Unit 1

Practice

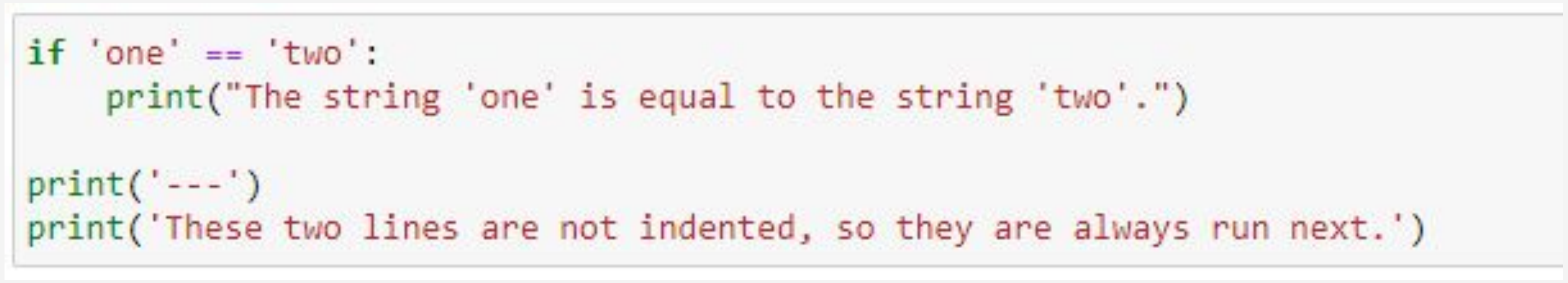
multiverse

Data Science Unit 1

Control Flow

Control Flow

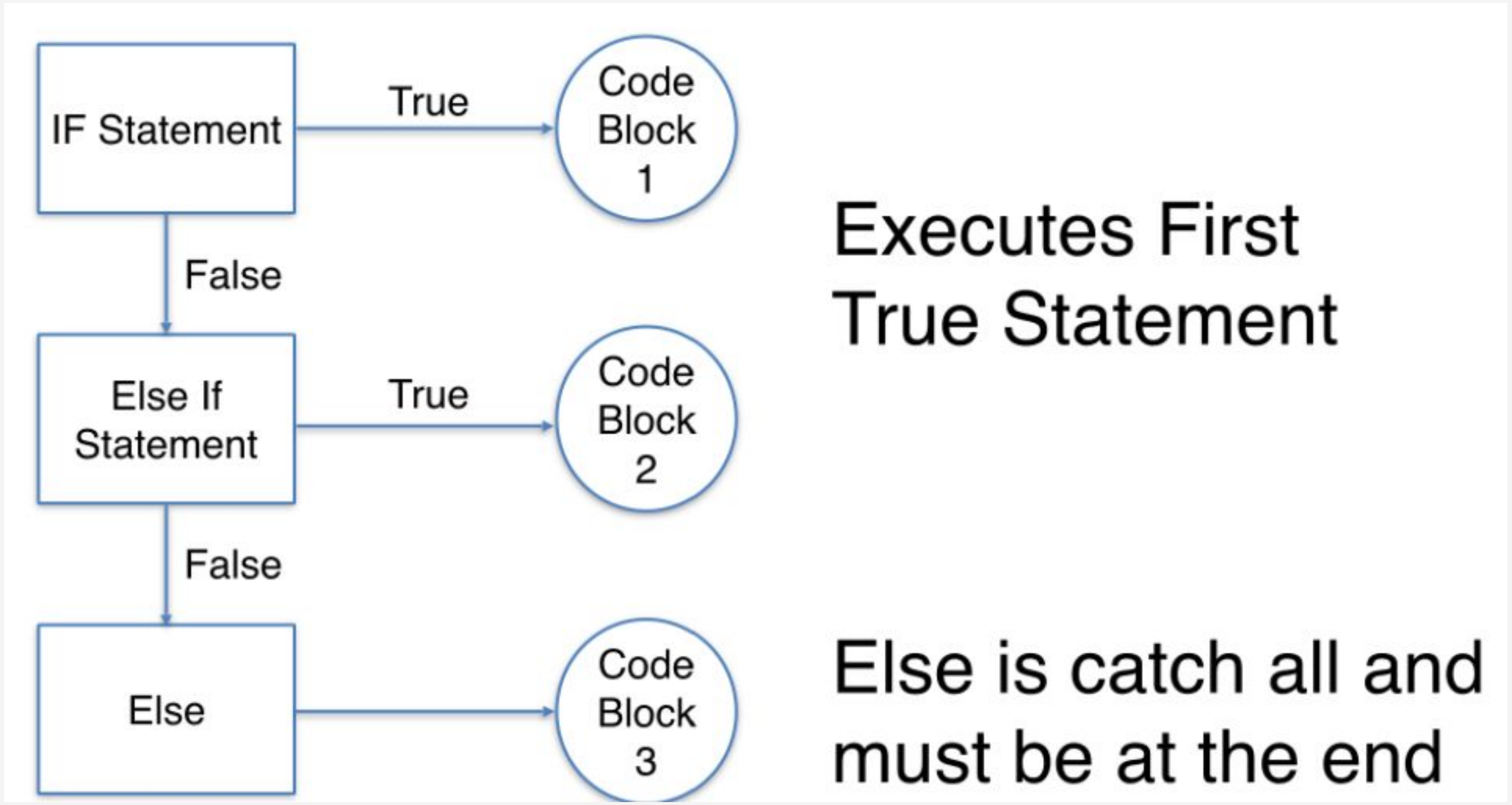
Indentation



```
if 'one' == 'two':
    print("The string 'one' is equal to the string 'two'.")

print('---')
print('These two lines are not indented, so they are always run next.')
```

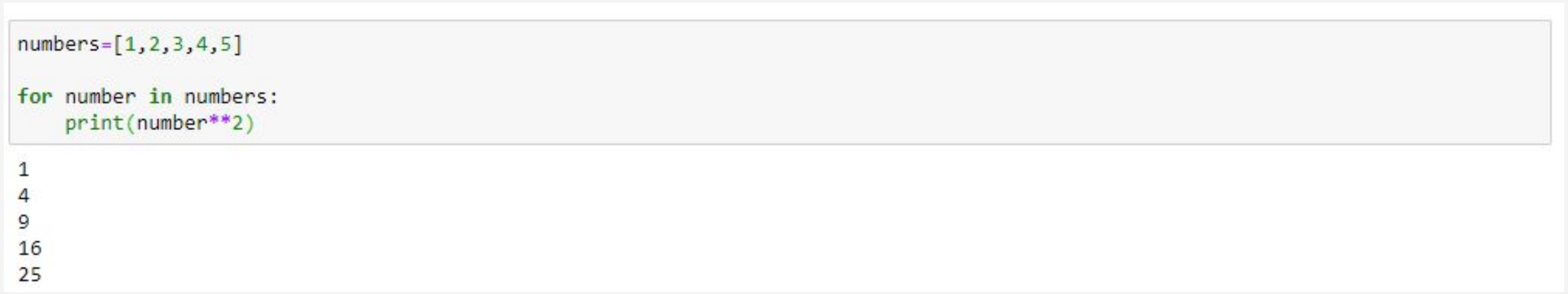
If/else



Executes First True Statement

Else is catch all and must be at the end

For

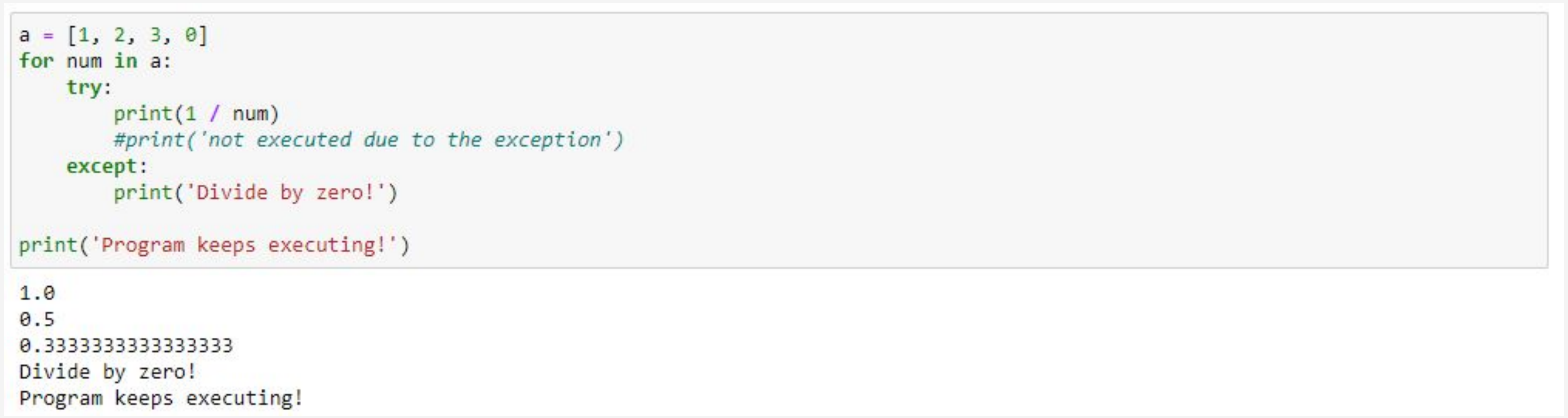


```
numbers=[1,2,3,4,5]

for number in numbers:
    print(number**2)
```

```
1
4
9
16
25
```

try-except

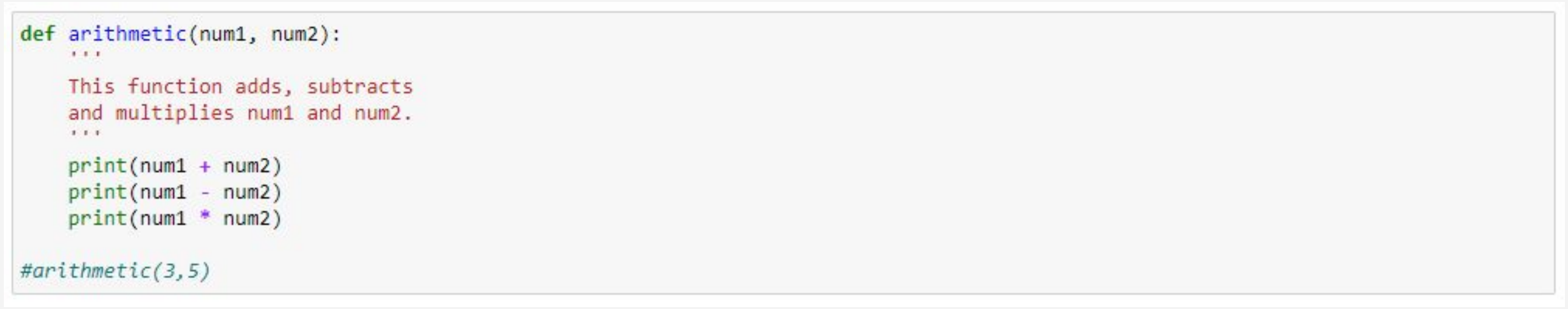


```python
a = [1, 2, 3, 0]
for num in a:
    try:
        print(1 / num)
        #print('not executed due to the exception')
    except:
        print('Divide by zero!')

print('Program keeps executing!')
```

```
1.0
0.5
0.3333333333333333
Divide by zero!
Program keeps executing!
```

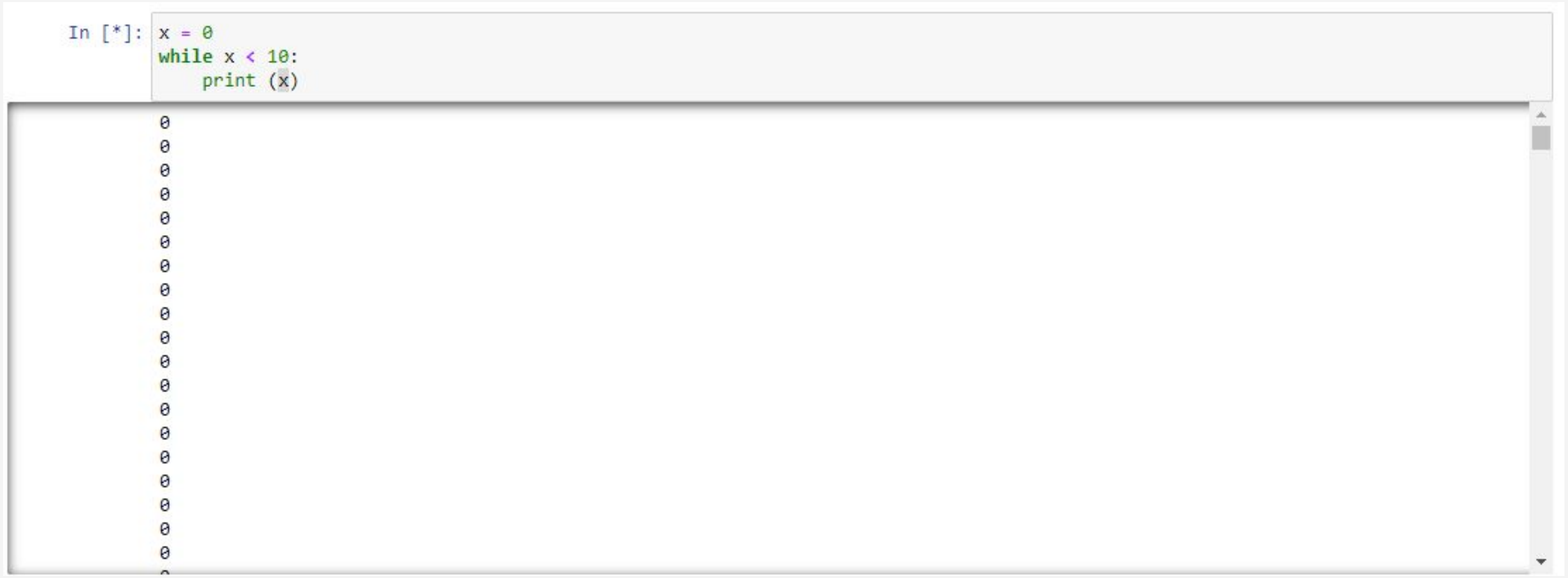
Functions



```
def arithmetic(num1, num2):
    '''
    This function adds, subtracts
    and multiplies num1 and num2.
    '''
    print(num1 + num2)
    print(num1 - num2)
    print(num1 * num2)

#arithmetic(3,5)
```

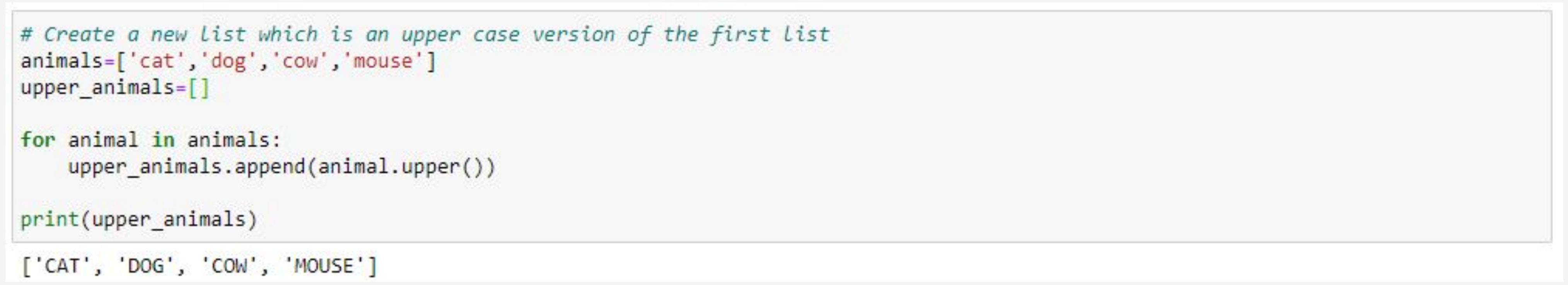
While



```
In [*]: x = 0
        while x < 10:
            print (x)
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

List and Dictionary Comprehensions

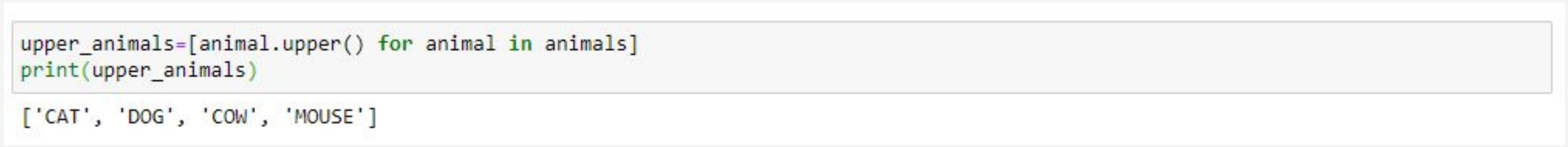


```
# Create a new list which is an upper case version of the first list
animals=['cat','dog','cow','mouse']
upper_animals=[]

for animal in animals:
    upper_animals.append(animal.upper())

print(upper_animals)
```

```
['CAT', 'DOG', 'COW', 'MOUSE']
```



```
upper_animals=[animal.upper() for animal in animals]
print(upper_animals)
```

```
['CAT', 'DOG', 'COW', 'MOUSE']
```

multiverse

Data Science Unit 1

Practice