Basics

```
In [2]: x=5
```

```
In [3]: x
Out[3]: 5
```

```
In [4]: print(x)
        5
```

```
In [5]: "Hello World!"
Out[5]: 'Hello World!'
```

```
In [6]: Hello World!

          Cell In[6], line 1
            Hello World!
                  ^
        SyntaxError: invalid syntax
```

```
In [ ]: 
```

```
In [7]: x=ajdkeenfkjfkhe

        ---------------------------------------------------------------------------
        NameError                                 Traceback (most recent call last)
        Cell In[7], line 1
        ----> 1 x=ajdkeenfkjfkhe

        NameError: name 'ajdkeenfkjfkhe' is not defined
```

```
In [8]: x='ajdkeenfkjfkhe'
```

```
In [9]: type(x)
Out[9]: str
```

```
In [ ]: 
```

```
In [10]: 'Hello World 5!'
Out[10]: 'Hello World 5!'
```

```
In [ ]:
```

We can write anything we like into markdown cells, and can comment our programming code

Hello World

```
In [ ]:
```

Data Types

```
In [11]: x="Hello World!"

         type(x)
Out[11]: str

In [12]: x=10
         type(x)
Out[12]: int

In [13]: x=3.78
         type(x)
Out[13]: float

In [14]: x=True
         y=False
         print(x)
         type(x)
         True
Out[14]: bool

In [15]: type(y)
Out[15]: bool

In [16]: z="True"

In [17]: type(z)
Out[17]: str
```

```
In [18]: p=true

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 p=true

NameError: name 'true' is not defined
```

```
In [ ]:
```

```
In [19]: x = [1, 2, 3, 4, 'a', "b"]
```

```
In [20]: type(x)

Out[20]: list
```

```
In [21]: x = (1, 2, 3, 4, 'a', "b")
```

```
In [22]: type(x)

Out[22]: tuple
```

```
In [23]: x = {1, 2, 3, 4, 'a', "b", 'b', 5, 5, 5}
```

```
In [24]: x

Out[24]: {1, 2, 3, 4, 5, 'a', 'b'}
```

```
In [25]: type(x)

Out[25]: set
```

```
In [26]: x = {"a" : 1, "b" : 2, "c" : 3}
```

```
In [27]: x

Out[27]: {'a': 1, 'b': 2, 'c': 3}
```

```
In [28]: type(x)

Out[28]: dict
```

```
In [29]: x={1 : 1, 2 : 2, 3 : 3}
```

```
In [30]: x

Out[30]: {1: 1, 2: 2, 3: 3}
```

```
In [31]: type(x)
Out[31]: dict
```

```
In [ ]:
```

Operators

```
In [32]: 2 + 3
Out[32]: 5
```

```
In [33]: x = 2+3
```

```
In [34]: print(x)

5
```

```
In [35]: x=3-2
         print(x)

1
```

```
In [36]: x=7*2
         print(x)

14
```

```
In [37]: x=8/2
         print(x)

4.0
```

```
In [38]: x=7//2 # Floor division
         print(x)

3
```

```
In [39]: x=7%2 # Modulus
         print(x)

1
```

```
In [40]: x=7**2
         print(x)

49
```

```
In [41]: print("Ha " * 10)

Ha Ha Ha Ha Ha Ha Ha Ha Ha Ha
```

```
In [ ]:
```

Comparisons

```
In [42]: 6 == 6
Out[42]: True
```

```
In [43]: 6!=6
Out[43]: False
```

```
In [44]: 6>7
Out[44]: False
```

```
In [45]: 6<7
Out[45]: True
```

```
In [ ]:
```

Logical Operators

```
In [46]: x=True
         y=False
```

```
In [47]: type(x)
Out[47]: bool
```

```
In [48]: x and y
Out[48]: False
```

```
In [49]: x or y
Out[49]: True
```

```
In [ ]:
```

Assignments

```
In [50]: m = 3
         n = 5
```

```
In [51]: m = m + 3
         print(m)

         6

In [52]: n += 3 # the same as n = n+3
         print(n)

         8

In [53]: n = n+3
         print(n)

         11

In [ ]:
```

Membership

You can search in text documents for values, words or word combinations (text data analysis/ Natural Language Processing):

```
In [54]: s = "Programming in Python is fun!"

In [55]: "y" in s

Out[55]: True

In [56]: "G" in s

Out[56]: False

In [57]: t = [1, 2, 'a', "b", 'c']

In [58]: 1 in t

Out[58]: True

In [59]: "a" in t

Out[59]: True

In [ ]:

In [ ]:

In [ ]:
```

```
In [ ]:
```

Markdown

Run the following code lines (one time as Code cell, and one time as Markdown cell):

Markdown cells

```
In [ ]: # Hello World
```

```
In [ ]: ### Hello World
```

```
In [ ]: **How are you?**
```

Hello World

Hello World

How are you?

Code cells:

```
In [65]: # Hello World
```

```
In [66]: ### Hello World
```

```
In [67]: **How are you?**
```

```
  Cell In[67], line 1
    **How are you?**
    ^
SyntaxError: invalid syntax
```

```
In [ ]:
```

```
In [ ]:
```

Define a variable "savings" and display it on the screen:

```
In [68]: savings = 1000
In [69]: savings
Out[69]: 1000
```

```
In [70]: print(savings)
         1000
```

```
In [ ]:
```

```
In [71]: # Counting letters:
```

```
In [72]: len("Hello")
Out[72]: 5
```

```
In [ ]:
```

Summary on string, operator, function, method:

String (S):

```
In [73]: S="Hello World"
         print(S)
         Hello World
```

Two objects (S and a string "!"), which are combined with an operator (+) :

```
In [74]: x = S + "!"
         print(x)
         Hello World!
```

Function (len() computes the number of elements); an object (x) is attributed to a function:

```
In [75]: len(x)
Out[75]: 12
```

Method. A method is assigned to an object, with a dot. upper() capitalises all letters:

```
In [76]: x.upper()
Out[76]: 'HELLO WORLD!'
```

```
In [ ]:
```

We need these simple programming steps for text data analyses.

Suppose we have the following text corpus x:

```
In [77]: x = "Inflation increased in the past 10 years."
         print(x)

         Inflation increased in the past 10 years.
```

```
In [ ]:
```

```
In [ ]:
```

Methods for strings:

```
In [80]: x.find("Inflation")

Out[80]: 0
```

```
In [81]: x.find("D")

Out[81]: -1
```

```
In [82]: x.find("d")

Out[82]: 18
```

```
In [83]: x = "Hello World"
```

```
In [84]: x.upper()

Out[84]: 'HELLO WORLD'
```

```
In [85]: x.lower()

Out[85]: 'hello world'
```

```
In [86]: x

Out[86]: 'Hello World'
```

```
In [87]: x.capitalize()

Out[87]: 'Hello world'
```

```
In [88]: x.find("H")
Out[88]: 0
```

```
In [89]: x.replace("d", "x")
Out[89]: 'Hello Worlx'
```

```
In [90]: x = "     Hel   lo   Wor   ld     "
```

```
In [91]: x.strip(" ")
Out[91]: 'Hel   lo   Wor   ld'
```

```
In [92]: x.split()
Out[92]: ['Hel', 'lo', 'Wor', 'ld']
```

```
In [ ]:
```

Methods for list:

```
In [93]: x = [1, 2, 3, 4, 5]
```

```
In [94]: x
Out[94]: [1, 2, 3, 4, 5]
```

```
In [95]: x.append(8)
```

```
In [96]: x
Out[96]: [1, 2, 3, 4, 5, 8]
```

```
In [97]: x.insert(5, 3)
```

```
In [98]: x
Out[98]: [1, 2, 3, 4, 5, 3, 8]
```

```
In [99]: x.remove(8)
```

```
In [100]: x
Out[100]: [1, 2, 3, 4, 5, 3]
```

```
In [101]: x.extend([3, 4, 5])
In [102]: x
Out[102]: [1, 2, 3, 4, 5, 3, 3, 4, 5]
```

```
In [103]: x.pop(0)
Out[103]: 1
```

```
In [104]: x
Out[104]: [2, 3, 4, 5, 3, 3, 4, 5]
```

```
In [105]: x.sort()
```

```
In [106]: x
Out[106]: [2, 3, 3, 3, 4, 4, 5, 5]
```

```
In [107]: x.reverse()
```

```
In [108]: x
Out[108]: [5, 5, 4, 4, 3, 3, 3, 2]
```

```
In [ ]:
```

Conditionals / If-else

```
In [109]: x=5

          if x > 0:
              print("Positive")

          Positive
```

```
In [110]: x=5

          if x > 0:
              print("Positive")
          elif x < 0:
              print("Negative")
          else:
              print("zero")

          Positive
```

```
In [ ]:
```

In [115]:
```
x=-1

if x < 0:
    print("Negative")
elif x == 2:
    print("two")
elif x > 2:
    print("Positive and greater than two")
else:
    print("Zero or less than two but positive")
```

```
Negative
```

In [ ]:

Loops:

In [116]:
```
x=0

while x < 5:
    x += 1
    print(x)
```

```
1
2
3
4
5
```

In [117]:
```

for i in [1, 2, 3, 4, 5]:
    print(i)
```

```
1
2
3
4
5
```

In [118]:
```
for i in range(1, 6):
    print(i)
```

```
1
2
3
4
5
```

In [ ]:

Functions

```
In [120]: def greater_function(x, y):
              if x > y:
                  return x
              else:
                  return y

          t = greater_function(8, 4)
          print(t)

          8
```

```
In [ ]:
```

Indexing

```
In [121]: list1 = [4, 2, 7.5, 3]
          list2 = [[4, 5, 6], "Python", 'Java', (2, 3, 4)]

In [122]: list1[0]

Out[122]: 4

In [123]: list1[2]

Out[123]: 7.5

In [124]: list2[0]

Out[124]: [4, 5, 6]

In [125]: list2[-1]

Out[125]: (2, 3, 4)

In [126]: list2[-2]

Out[126]: 'Java'

In [127]: list2[2][0]

Out[127]: 'J'

In [128]: list1[-2]

Out[128]: 7.5

In [129]: list2[-2][-2]

Out[129]: 'v'
```

```
In [130]: list2[0]="R"
          print(list2)

          ['R', 'Python', 'Java', (2, 3, 4)]
```

```
In [ ]:
```

Slicing

```
In [139]: list1[1:]

Out[139]: [2, 7.5, 3]
```

```
In [140]: Part1= list1[0]
          print(Part1)

          4
```

```
In [141]: Part2 = list1[1:]
          print(Part2)

          [2, 7.5, 3]
```

```
In [142]: Part3 = list2[:3]
          print(Part3)

          ['R', 'Python', 'Java']
```

```
In [143]: Part4 = list2[:4]
          print(Part4)

          ['R', 'Python', 'Java', (2, 3, 4)]
```

```
In [144]: Part5 = list2[:5]
          print(Part5)

          ['R', 'Python', 'Java', (2, 3, 4)]
```

```
In [145]: Part6 = list2[1:3]
          print(Part6)

          ['Python', 'Java']
```

```
In [ ]:
```

Change of data type

```
In [1]: name = "Daniela Schmidt"
In [2]: type(name)

Out[2]: str

In [3]: x= list(name)

In [4]: print(x)

        ['D', 'a', 'n', 'i', 'e', 'l', 'a', ' ', 'S', 'c', 'h', 'm', 'i', 'd', 't']

In [5]: x=3.574

In [6]: type(x)

Out[6]: float

In [7]: y=str(x)

In [8]: type(y)

Out[8]: str
```