Introduction to Jupyter

This is my first program of learning Python

In [1]:
```
msg = "john nash"
print(msg)
```
```
john nash
```

If we want to make capital letter or small letter of the above John Nash, then:

In [2]:
```
print(msg.upper())
print(msg.title())
```
```
JOHN NASH
John Nash
```

String Concatenation

In [4]:
```

line1 = "You thought it would be easy"
line2 = "You thought it wouldn't be strange"
line3 = "But then you started coding"
line4 = "Things never were the same"

print(line1)
print(line2)
print(line3)
print(line4)
```
```
You thought it would be easy
You thought it wouldn't be strange
But then you started coding
Things never were the same
```

In [5]:
```
print(line1+line2+line3+line4)
```
```
You thought it would be easyYou thought it wouldn't be strangeBut then you star
ted codingThings never were the same
```

In [6]:
```
concat_strings = line1+line2+line3+line4
print(concat_strings)
```
```
You thought it would be easyYou thought it wouldn't be strangeBut then you star
ted codingThings never were the same
```

In [8]:
```
print(line1,line2,line3,line4,sep = "\n")
```

```
You thought it would be easy
You thought it wouldn't be strange
But then you started coding
Things never were the same
```

Formula

By using the option, sep = "\n", each comma is interpretted as a new line. It is the same as adding "\n" to every line.

Distinction Double & Single Quotes

In [20]:
```
single_in_double = "We may use 'single quotes' within double quotes"
double_in_single = 'We  may use "double quotes" in double quotes'
double_in_double = "We may use \"double quotes\" in double quotes."
single_in_single = 'We may use \'single quotes\' in single quotes.'

print(single_in_double)
print(double_in_single)
print(double_in_double)
print(single_in_single)
```

```
We may use 'single quotes' within double quotes
We  may use "double quotes" in double quotes
We may use "double quotes" in double quotes.
We may use 'single quotes' in single quotes.
```

In [21]:
```
read_backslash = \
    "We may use two backslashes for a single backslash: \\"
new_line_and_tab = \
    "We may start a new line \n\tand use tab for a hanging indent"
print(read_backslash)
print(new_line_and_tab)
```

```
We may use two backslashes for a single backslash: \
We may start a new line
        and use tab for a hanging indent
```

.strip() and .replace() string methods

In [25]:
```
spaces = "    Look at the spaces in the text!    "
print("no spaces removed:", spaces, sep = "\n")

remove_left_spaces = spaces.lstrip()
remove_right_spaces = spaces.rstrip()
remove_left_and_right_spaces = spaces.strip()
remove_all_spaces = spaces.replace(" ","")

print("Remove left spaces:", remove_left_spaces)
print("Remove right spaces:", remove_right_spaces)
print("Remove left and right spaces:", remove_left_and_right_spaces)
print("Remove all spaces:", remove_all_spaces)

print("Capitalize all first letters:", remove_left_spaces.title())
```

```
no spaces removed:
    Look at the spaces in the text!
Remove left spaces: Look at the spaces in the text!
Remove right spaces:     Look at the spaces in the text!
Remove left and right spaces: Look at the spaces in the text!
Remove all spaces: Lookatthespacesinthetext!
Capitalize all first letters: Look At The Spaces In The Text!
```

Working with Values

In [26]:
```
num1 = 5 + 3
num1s = "5" + "3"

print("num1:", num1,"\nnum1s:", num1s)
```

```
num1: 8
num1s: 53
```

In [27]:
```
num1 = 5 / 3
num2 = 5 / 4
num3 = 4 / 3

print("num1:", num1)
print("num2:", num2)
print("num3:", num3)
```

```
num1: 1.6666666666666667
num2: 1.25
num3: 1.3333333333333333
```

In [28]:
```
type(num1)
```

Out[28]:
```
float
```

In [29]:
```
float(3)
```

Out[29]:
```
3.0
```

```
In [30]: 3 + 1.5

Out[30]: 4.5

In [31]: type(3 + 1.5)

Out[31]: float

In [32]: import sys
         sys.float_info

Out[32]: sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=
         2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, e
         psilon=2.220446049250313e-16, radix=2, rounds=1)

In [33]: 2. ** 1023

Out[33]: 8.98846567431158e+307

In [34]: 2. ** 1024

         ---------------------------------------------------------------------------
         OverflowError                             Traceback (most recent call last)
         <ipython-input-34-b5418d78437f> in <module>
         ----> 1 2. ** 1024

         OverflowError: (34, 'Result too large')
```