In [1]:

```
print ("Hello world")   # python 3 version
```

```
Hello world
```

In [2]:

```
my_text = "Hello world"
print(my_text)
```

```
Hello world
```

In [3]:

```
help(print)  # function that can access information and, for example,
              # see the documentation of a built-in
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

BASIC MATH

In [4]:

```
2+2
```

Out[4]:

```
4
```

In [5]:

```
2.0 + 2.0        # Floating point
```

Out[5]:

```
4.0
```

In [6]:

```
5/2              # la divisione tra due numeri interi restituisce un float
```

Out[6]:

```
2.5
```

In [7]:

```
5//2             # double slash to force integer division
```

Out[7]:

```
2
```

In [8]:

```
5 % 2            # 5 modulo 2
```

Out[8]:

```
1
```

In [10]:

```
3**2            # Exponentiation
```

Out[10]:

```
9
```

In [11]:

```
_ +2            # underscore "_" is the last returned value
```

Out[11]:

```
11
```

Output and formatting

Basic numerical types: int, float, complex

In [12]:

```
fl = 2/3
print(fl)
```

```
0.6666666666666666
```

In [13]:

```
print (type(fl), "\n") # Python's built-in float type has double precision. If you need more precision, get NumPy and use its numpy.float128.

print ("%f" %fl )
print ("%.2f" %fl)
print ("%.20f" %fl)
print ("%2f" %fl)
print ("%20f this is FL" %fl)
print ("%-20f this is FL" %fl)

# print ("\n %d" %(fl+3))  # as integer
```

```
<class 'float'>

0.666667
0.67
0.66666666666666662966
0.666667
            0.666667 this is FL
0.666667             this is FL
```

In [14]:

```
c = 3.4+2.3j
print(c)
print(c.real)
print(c.imag)

print ("%.5f" %c.imag)
```

```
(3.4+2.3j)
3.4
2.3
2.30000
```

Types and type conversions

- **Basic numerical types: int, float, complex**
- **Special types: bool, None**

- **Container types: str, tuple, list, dict, ...**

Additional numerical types and container are provided by NumPy

In [15]:

```
a = input('enter a number: ')
print(a, type(a))

a = float(a)
print(a, type(a))
```

```
enter a number: 7
7 <class 'str'>
7.0 <class 'float'>
```

In [16]:

```
print(int(4.255), float(7), complex(5.3))
```

```
4 7.0 (5.3+0j)
```

In [17]:

```
my_text = "Hello world"
print(type(my_text) ) # str is a string of character
print(len(my_text))

str(265)
```

```
<class 'str'>
11
```

Out[17]:

```
'265'
```

Relational and boolean operators

In [19]:

```
x = 7
y = 7.0
z = 10

print (x==y)
print (x>z)
print (x<=y)
print (x!=y)
```

```
True
False
True
False
```

In [20]:

```
A = not (x==y) # NOT True
print ('A = ', A)
B = (x==y) and (x>z) # True AND False
print ('B = ', B)
C = (x==y) or (x>z) # True OR False
print('C = ', C)
```

```
A =  False
B =  False
C =  True
```

Python modules

A module is a Python object with arbitrarily named attributes that you can bind and reference. Grouping related

code into a module makes the code easier to understand and use. \ Simply, a module is a file consisting of Python code where you can define functions, classes and variables (a module can also include runnable code).

In [22]:

```
from math import sqrt
x = 55.3
y = sqrt(x)
print(y)

# print(math.sin(x))
# print(pi)
```

```
7.436396977031283
```

In [21]:

```
import math
x = 55.3
print(math.sin(x))
```

```
-0.9485636937183082
```

In [24]:

```
# from math import pi
print(math.pi)
print (math.cos(x))
```

```
3.141592653589793
0.31658635308471483
```

In [ ]:

```
help(math.exp)
```

In [ ]:

```
help(math)
```

CONDITIONAL STATEMENT

In [31]:

```
# IF

from random import random
from random import randint
x = random()
y = randint(1, 200)
print(x,'\n', y)

if y%2==0:
  print ('y is even')
else:
  print ('y is odd')
```

```
0.7879884558771489
 27
y is odd
```

In [32]:

```
# WHILE

print('y =', y)
k = 1
while k<=y:
  print(k)
  k = k+1
```

```
print ('Last k', k) # WARNING!!
                    # Python uses indentation to indicate a block of code
```

```
y = 27
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
Last k 28
```

In [33]:

```
# FOR

primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)
```

```
2
3
5
7
```

In [34]:

```
# help (range)
sequence = [i for i in range(10)]
print(sequence)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

CONTAINER TYPES

Tuples: immutable sequences of objects of any type

In [ ]:

```
t = (1, 'hello', 3.14, True, (3.14, False)) # or () brackets!
print(t)
print(len(t))
print(t[0])
print(t[4])
print(t[-1])
print(type(t))
print(type(t[0]))
print(type(t[4]))
```

In [36]:

```
t[0:3]  # start:end
```

Out[36]:

```
(1, 'hello', 3.14)
```

In [38]:

```
3.14 in t  #test for inclusion
```

Out[38]:

```
True
```

In [39]:

```
t + (5, )  # concatenate tuples
# Not allowed: t+(5)
```

Out[39]:

```
(1, 'hello', 3.14, True, (3.14, False), 5)
```

In [ ]:

```
print(t[0])
t[0] = 2 #TypeError: 'tuple' object does not support item assignment
```

In [41]:

```
tt = ('a',1) * 3 # concatenate
print(tt)
```

```
('a', 1, 'a', 1, 'a', 1)
```

In [ ]:

```
tt[0:len(tt):2]
```

Lists: mutable sequences of objects of any type

In [42]:

```
l = [1, 'hello', 3.14, True, (3.14, False) ] # square brackets
print(l)
```

```
[1, 'hello', 3.14, True, (3.14, False)]
```

In [43]:

```
l[0] = 2 # now it is allowed
print(l)
```

```
[2, 'hello', 3.14, True, (3.14, False)]
```

In [44]:

```
# as for tuples:
print(len(l))
print(2.5 in l)
print(l[0:3])
```

```
5
False
[2, 'hello', 3.14]
```

In [45]:

```
v = []  # empty list
```

```
print(type(v), len(v))

v.append(5)
print(v)
v.append('hello')
print(v)
```

```
<class 'list'> 0
[5]
[5, 'hello']
```

FUNCTIONS

In [46]:

```
def print_sum(x,y):
  """PRINT_SUM prints and returns the sum of two input numbers
  """
  s = x+y
  print('Sum of two input numbers: ', s)
  return s

a = 7
b = 2
my_sum = print_sum(a,b)
print("The returned value is ", my_sum)
```

```
Sum of two input numbers:  9
The returned value is  9
```

In [47]:

```
s = print_sum( 3+5j , 7) # type compatibility
```

```
Sum of two input numbers:  (10+5j)
```

In [48]:

```
help(print_sum)
```

```
Help on function print_sum in module __main__:

print_sum(x, y)
    PRINT_SUM prints and returns the sum of two input numbers
```

In [49]:

```
def sum_and_diff(x,y):
  """SUM_AND_DIFF computes sum and difference
  sum is the first output
  """
  return x+y, x-y # return as a tupla
```

In [ ]:

```
help(sum_and_diff)
```

In [50]:

```
print(sum_and_diff(a,b))
S, D = sum_and_diff(a,b) # tupla
print(D)
```

```
(9, 5)
5
```

In [51]:

```
# Variabili locali immutabili - interi
```

```
def myrandom1(x):
  x = random()
  return x

def myrandom2():
  a = random()
  return a

a = 2
print('myrandom1: ', myrandom1(a))
print('a = ', a)
print('myrandom2:', myrandom2())
print('a = ', a)
```

```
myrandom1:  0.04713829152522497
a =  2
myrandom2: 0.8281644390999547
a =  2
```

In [52]:

```
# Variabili locali mutabili - liste

v = [i for i in range(2, 9)]
print ('Original v: \t', v)
print(len(v))

def redouble(x):
  for i in range(0, len(x)):
    x[i] = 2*x[i]
  return x

print ('Output: \t', redouble(v))    # NOT v = redouble(v)
print ('New v: \t\t', v)

def redouble2(x):
  d = []
  for i in range(0, len(x)):
    d.append(2*x[i])
  return d

v2 = redouble2(v)
print ('v input: \t', v)
print ('v2 output: \t', v2)
```

```
Original v:   [2, 3, 4, 5, 6, 7, 8]
7
Output:   [4, 6, 8, 10, 12, 14, 16]
New v:    [4, 6, 8, 10, 12, 14, 16]
v input:   [4, 6, 8, 10, 12, 14, 16]
v2 output:   [8, 12, 16, 20, 24, 28, 32]
```

Python 3 reference card

http://achievatek.com/wp-content/uploads/2017/03/AT-Python-3-%E2%80%94-Quick-Reference-Card.pdf