

* Abs(x,/)

**#abs(x,/) --> Positinal only**

print(abs(-70))

#70

print(abs(-70.12))

#70.12

print(abs(3+4j))

#5.0

#SQRT(a\*\*a + b\*\*b) = Hypotenuse

**#pow(base,exp,mod=None,/) --> Positinal only**

print(pow(2,3))

#8

print(pow(10,2))

#100

print(pow(10,2,3))

# (10\*2)%3 = 1

**#round**

**# round(number,ndigits=None)**

print(round(4.4))

#4

print(round(4.6))

#5

print(round(4.5))

#4

print(round(5.5))

#6 --> Close to even number

print(round(3.5421))

#4

print(round(3.5421,2))

#3.54

**#divmod**

**#divmod(a,b,/) --> Positinal only**

print(divmod(10,3))

#(3,1)

#10/3= 3.3333

#10%3 = 1

print(divmod(61,7))

#(8,5)

**######### min**

**#min(iterable, \*,key=None,default=None) --> Keyword arguments(\*)**

print(min(10,3,7,2,5))

#2

print(-10,3,7,-2,-5)

#-10

print(min(-10,3,7,-2,-5,key=abs))

#-2

print(min([],default="Empty List"))

#Empty List

**################### max**

**#max(iterable, \*,key=None,default=None) --> Keyword arguments(\*)**

words=["apple","banana","kiwi","grape"]

max(words,key=len)

#'banana'

**############################**

**############ sum**

#sum(iterable,start=0)

print(sum([1,2,3,4,5]))

#15

print(sum([1,2,3,4,5],start=20))

# #35

**################ EVAL**

**#eval(expression,globals=None)**

**#expression-string**

print(eval("10+20\*4-5"))

#85

global\_dict={"x":15,"y":20}

locals\_dict={"a":5}

eval("x + y + a",global\_dict,locals\_dict)

#35--> 15+20+5

A screen shot of a computer

AI-generated content may be incorrect.

**########Type**

**# type(object,bases=None,dict=None) -> a type object**

print(type(10))

#<class 'int'>

print(type(10.5))

#<class 'float'>

print(type("Hello"))

#<class 'str'>

print(type(True))

#<class 'bool'>

print(type([1,2,3]))

#<class 'list'>

print(type((1,2,3)))

#<class 'tuple'>

print(type({1,2,3}))

#<class 'set'>

print(type({"a":1,"b":2}))

#<class 'dict'>

print(type(None))

#<class 'NoneType'>

#

**############# isistance**

**#**

# isinstance(object,classinfo) -> bool

#

x=10

print(isinstance(x,int))

#True

print(isinstance(x,float))

#False

print(isinstance(x,(int,str)))

#True

**##hasattr**

**#hasattr(object,attribute) -> bool**

s="Hello World"

print(hasattr(s,"find"))

#True

print(hasattr(s,"islower"))

#True

print(hasattr(s,"search"))

#False

**#################getattr**

**#getattr(object,attribute,default=None) -> value**

import math

print(getattr(math,"pi"))

#3.141592653589793

#

print(getattr(math,"sqrt")(16))

#4.0

**#####################ID**

**#id(object) -> integer**

x=10

y=10

print(id(x),id(y))

#140732799907888 140732799907888

L1=[1,2,3]

L2=[4,5,6]

print(id(L1),id(L2))

#140732778823680 140732778824064

**############dir**

**#dir(object)**

print(dir(list))

# #['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter

print(dir(math))

# #['\_\_doc\_\_', '\_\_loader\_\_', '\_\_name\_\_', '\_\_package\_\_', '\_\_spec\_\_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

**#########repr**

**#repr(object) -> string**

text="Hello World"

print(repr(text))

#"Hello World"

A white paper with black text

AI-generated content may be incorrect.

Sorted:

# sorted(iterable,/,\*,key=None,reverse=False) --> Positinal and Keyword arguments(\*)

L1=[1,12,7,-3,8]

print(sorted(L1))

# [-3, 1, 7, 8, 12]

print(sorted(L1,reverse=True))

# [12, 8, 7, 1, -3]

print(sorted(L1,key=abs))

# [1, -3, 7, 8, 12]

**############### Reverse**

**# reverse(iterable) --> Positinal only**

L1=[1,12,7,-3,8]

print(reversed(L1))

# <list\_reverseiterator object at 0x000001D5518070B8>

rev=reversed(L1)

print(list(rev))

# [8, -3, 7, 12, 1]

###################slice

**#slice(start=None,stop=None,step=None) --> Positinal only**

**#stop compulsory**

**#s=slice(0,stop,1)**

L1=[10,20,30,40,50,60,70]

s=slice(5)

print(L1[s])

# [10, 20, 30, 40, 50]

**######## Iter/Next**

**############ iter(object,sentinel=None) --> Positinal only**

L1=[10,20]

it=iter(L1)

print(next(it))

#10

print(next(it))

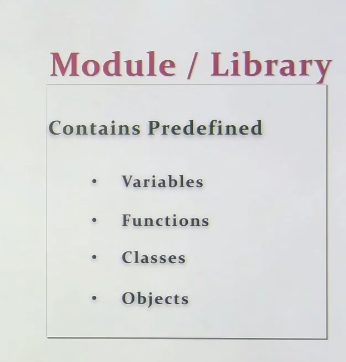
#2

print(next(it))

# StopIteration

Modules:

* Modules as library



Math

DateTime

OS

RegularExp

import re

import datetime as dt

from math import \*

Own Module:

MyModule.py

data=500

def add(a,b):

    return a+b

def sub(a,b):

    return a-b

if \_\_name\_\_=="\_\_main\_\_":

    print("sum is:",add(10,5))

    print("sub is:",sub(10,5))

#Result

# sum is: 15

# sub is: 5

MyProgram.py

import MyModule

print("Data from MyModule:", MyModule.data)

print("Addition from MyModule:", MyModule.add(20, 10))

print("Subtraction from MyModule:", MyModule.sub(20, 10))

#Result 3:

#Inclusion of \_\_name\_\_=="\_\_main\_\_" in module

# $ python MyProgram.py

# Data from MyModule: 500

# Addition from MyModule: 30

# Subtraction from MyModule: 10

A white card with red text and numbers

AI-generated content may be incorrect.