Mathematical functions

Built-in
Abdul

↳ abs(x, /) → positional only

abs(-70) #70
abs(-70.12) #70.12
abs(3+4j) = 5 (3^2+4^2)

↳ pow(2,3) // pow(base, exp, mod=None) // positional only.

pow(2,3) #8
pow(10,2) #100
pow(10,2,3) 100 % 3 = 1

↳ round round(number, ndigits=None)

round(4.4) #4
round(4.6) #5
round(4.5) #4
round(5.5) #6 } close to even number
round(3.5421) #4
round(3.5421,2) #3.54

↳ divmod divmod(a, b, /) → positional only

divmod(10,3)
(3,1)
10/3 = 3.3333
10%3 = 1
divmod(61,7)
(8,5)

↳ min
min(iterable, *, key=None, default=None)
* → keyword args

min(10,3,7,2,5)
# 2
min(-10,3,7,-2,-5)
# -10
min(-10,3,7,-2,-5, key=abs)
-2
min([], default="Empty List")
# Empty List

↳ Max
max(iterable, *, key=None, default=None)
words = ["apple", "banana", "kiwi", "grape"]
max(words, key=len)
# 'banana'

built-in
Asaul 2

↳ 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

globals_dict = {'x': 15, 'y': 20}
locals_dict = {'a': 5}
eval ('x+y+a', globals_dict, locals_dict)
#35 → 15+20+5

Object attributes

type (object, base=None, dict=None)
→ a type object

type (10) → int
type (10.5) → float
type ("Hello") → str
type (True) → bool
type ([1,2,3]) ⇒ list

type ((1,2,3)) → tuple
type ({1,2,3}) → set
type ({'a': 1, 'b': 2}) → dict
type (None) ⇒ NoneType

↳ instance

isistence(object, class_info) → bool

```
x = 10
isistence(x, int)   # True
isistence(x, float) # True
isistence(x, (int, str)) # True
```

↳ hasattr

hasattr(object, attribute) → bool

```
s = "hello world"
print(hasattr(s, "find"))
                    #True
print(hasattr(s, "islower"))
                    #True
hasattr(s, "search")
                    #False
```

↳ setattr

setattr(obj, attribute, default=None)

```
import math
setattr(math, "pi")
                    # 3.14
setattr(math, "set", 16)
                    #4.0
```

↳ ID

id(object) → integer

```
x = 10          id(x), id(y)
y = 10          888, 888
l1 = [1,2,3]
l2 = [4,5,6]
id(l1), id(l2)
887, 889
```

Wit in Abdul-3

↳ dir
dir(object)
dir(list)
dir(math)

↳ repr
repr(object) → String
text = "Hello world"
pt(repr(text))
# 'Hello world'

Iteration / Seq

↳ sorted(iterable, /, *, key=None, reverse=False)
L1 = [1, 12, 7, -3, 8]
sorted(L1) → [-3, 1, 7, 8, 12]
sorted(L1, key=abs) → [1, -3, 7, 8, 12]
sorted(L1, reverse=True)
= [12, 8, 7, 1, -3]

reverse(iterable)
→ pas only

↳ Reverse
L1 = [1, 12, 7, -3, 8]
reversed(L1) // iterator
rev = reversed(L1)
pt(list(rev))
[8, -3, 7, 12, 1]

#slice

#slice(start=None, stop=None, step=None)

stop Compulsory → positional only

```
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)

```
L1 = [10, 20]
it = iter(L1)
print(next(it)) #10
print(next(it)) #20
print(next(it)) # Stop Iteration
```

↳
Modules

```
import re
import datetime as dt
from math import *
```

Built-in Modules - 4

MyModule.PY

```
data = 500
def add(a,b):
    return a+b
def sub(a,b):
    return a-b
pt(__name__)
if __name__=="__main__":
    pt("sum", add(10,5))
    pt("sub", sub(10,5))
```

python MyModule.PY

```
__main__
15
5
```

MyProgram.py

```
import MyModule
pt(MyModule.data)
pt(MyModule.add(20,10))
pt(MyModule.sub(20,10))
```

O/P

```
MyModule
500
30
10
```