Lesson 10 Python Built-in Function

1.Built-in Function Definition

Python compiler is a program that provides users with common functions with unique names. And these common functions are built-in functions.

Built-in function is one part of compiler and takes effect as compiler starts. Standard library function is the external expansion of the compiler, and takes effect after the module is imported. In general, the execution efficiency of built-in function is higher than that of standard library function.

The amount of built-in functions must be strictly controlled, otherwise Python compiler will be “overstaffed”. Generally speaking, only the functions that are frequently used and are tightly bound to the language itself will be promoted as built-in function.

2. Built-in Function List

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slicea()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

2.1 Operation Type

1) abs(x)：abs function is used to return the absolute value of a value. The input x parameter can be the floating point number of the integer, and also can be complex number.

2) max(x，key=None)：The parameter x of the max function is an iterable object or multiple parameters, and max function will return the maximum element among them. max function can return the maximum value through designating the key parameter. If there are multiple maximum values, the first value will be returned.

3) min(x,key=None)：The usage of min function is the same as that of

max function. min function will return the minimum value of an iterable object or multiple parameters.

```
print(abs(-46))
print(max(2,3,4,5))
print(min(2,3,4,5))
```

```
46
5
2
```

4) pow(x,y, mod): When pow function only has x and y parameters, its function is to return x to the y power. When the third parameter mod exits, the pow function takes the remainder of mod on the basis of x raised to the y power.

5) round(number,ndigits)：Return the value of number rounded to ndigits of precision after the decimal point. If ndigits is omitted, the integer closest to the number is returned.

6) sum(x, start=0): sum function is to sum the values in x from left to right. Then add the value of start, sum and return the total value.

7) divmod(a, b): divmod function takes a and b as arguments, and return the quotient and remainder of a/b.

```
print(pow(2,3))
print(round(1.2345,3))
print(sum((1,2,3,4),2))
print(divmod(5,2))
```

```
8
1.234
12
(2, 1)
```

2.2 Convert the Type

1) int(x): If x is a number or string, the integer of x will be returned. If x is empty, 0 will be returned.

2) float(x): If x is a number or string, the floating point number of x will be returned. If x is empty, 0 will be returned.

3) complex(real, imag): It will return the complex form of **real + imag*1j**, or converts a string or number to the complex form. If the first parameter “real” is a string, the second parameter “**imag**” should be omitted.

```
print(int(2.3))
print(float(3))
print(complex(2,3))
```

```
2
3.0
(2+3j)
```

4) bool(x): Determine parameter x is true or false, and return **True** or **False**.

5) str(x): Convert the input x to string and return the result.

6) bytearray(x, encoding="utf-8"): bytearray converts the input x to ordered and mutable array composed of bytes, and return the array.

7) bytes(x, encoding="utf-8"): bytes convert the passed in parameter x to an immutable array of bytes, and return this array.

8) memoryview(x): it returns the memory view object of the parameter x, and the memory view object allows the data that supports the buffer protocol to be packaged, and the returned object is a list of tuples. Note: the incoming parameter x must be bytes.

```
# bool function
print(bool(3>4))
print(bool("2"=="2"))
# str function
print(str(3.45))
print(str([3,4]))
# bytearray function
print(bytearray("python", encoding="utf-8"))
# bytes function
print(bytes("python", encoding="utf-8"))
# memoryview function
print(memoryview(bytes(123)))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
False
True
3.45
[3, 4]
bytearray(b'python')
b'python'
<memory at 0x7f554c5eba00>
```

2.3 Base Conversion

1) bin(x): Convert the input integer x to a binary string prefixed with "0b"

2) oct(x): Convert the input integer x to an octal string prefixed with "0o"

3) hex(x): Convert the input integer x to a hexadecimal string prefixed with "0x".

4) ord(x): For the input single Unicode character, its corresponding Unicode integer will be returned.

5) chr(x): The chr function is the inverse of the ord function, and is used to return the corresponding single Unicode character for the input integer x.

```
# bin function
print(bin(3))
# oct function
print(oct(3))
# hex function
print(hex(3))
# ord function
print(ord('c'))
# chr function
print(chr(99))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
0b11
0o3
0x3
99
c
```

2.4 Numerical Operations

1) list(x): list function changes the incoming parameters to a new list and returns it. In addition, the list itself is also a mutable object.

2) dict(x): dict function changes the incoming parameter x into a new dictionary object and returns it. The dictionary object is immutable.

3) set(x): set function is used to change the input parameter x to a new collection object and return it. The set object is mutable, and its internal elements are not repeatable.

4) frozenset(x): frozenset function can also complete the function of the set function, but the frozenset object is immutable. Therefore it is not possible to insert values into the frozenset object.

```
# list function
print(list("1234"))
print(list())
# dict function
print(dict(a=2,b=4))
print(dict())
# set function
print(set())
set1 = set([1,2,3,4,1,2,3,4])
print(set1)
set1.add(5)
print(set1)
# frozenset function
frozenset1 = frozenset([1,2,3,4,1,2,3,41])
print(frozenset1)
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
['1', '2', '3', '4']
[]
{'a': 2, 'b': 4}
{}
set()
{1, 2, 3, 4}
{1, 2, 3, 4, 5}
frozenset({1, 2, 3, 4, 41})
```

5) tuple(x): tuple function changes the incoming parameter x into a new tuple object and returns it. The tuple object is immutable.

6) enumerate(x): It returns an enumeration object, and the input parameter x is an iterable object. Through **__next__()**, the returned enumeration object returns a tuple containing the count value and the value in x obtained by iteration.

7) range(x): The range function generates an immutable sequence of numbers from an input x, typically used to loop a specified number of times in a **for loop**

8) iter(x): The iter function generates an iterable object based on the input parameter x and returns the iterable object.

```
# tuple function
print(tuple([1,2,3,4]))
print(tuple())
# enumerate function
for index, value in enumerate("abed"):
    print(index, ":", value)
# range function
print(range(10))
print(range(0,10,2))
# iter function
iter1= iter([1,2,4,5])
for i in range(3):
    print(next(iter1))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
(1, 2, 3, 4)
()
0 : a
1 : b
2 : e
3 : d
range(0, 10)
range(0, 10, 2)
1
2
4

进程已结束,退出代码0
```

9) slice(x): The slice function is mainly used as flexible constructed slices, and it also returns a slice object.

10)object(): In Python, the object class is the base of all classes. The object function does not accept any parameters and returns an object without any characteristics.

11)super(): The super function is often used in Python class objects. In the inheritance of child classes from the parent class, the child class is used to refer to the objects and methods in the parent class without explicitly

specifying the name of the parent class.

```
# slice function
print(slice(5))
print(slice(0,5,2))
# object function
print(object())

# super function
class A:
    def __init__(self):
        self.fathername = "A"
class B(A):
    def __init__(self):
        super().__init__()
        self.sonmane = "B"
print(B().fathername)
print(B().sonmane)
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
slice(None, 5, None)
slice(0, 5, 2)
<object object at 0x7ff022f36d60>
A
B
```

2.5 Sorting Operation

1) sorted(x, key=None, reverse=False): Sort the iterable object x and return a new sorted object. The key parameter can specify how to compare, and when reverse is True, it indicates sorting in descending order.

2) reversed(seq): Reverse the input sequence seq to generate a new iterable object and return.

```
# sorted function
print(sorted([1,3,6,4,7,2]))
print(sorted([1,3,6,4,7,2], reverse=True))
print(sorted({"a":3,"b":6,"c":11}.items(), key=lambda x: x[1]))
# rever function
print(list(reversed([1,2,3,4,5,6])))
print(list(reversed("abcd")))
```

2.6. Sequence Operation

1) all(x): Determine whether each element of the iterable object x is true, and if one element is False (0), return False.

2) any(x): Same as the all function, the any function determines whether each element in x(iterable object) is True, and returns True as long as one element is True.

3) map(func, iter): The map function returns an iterator in which the func function will be applied to each element in the iterable object iter

4) filter(func, iter): Filter the element values in the iterable object iter through the func function, and return a new iterator composed of the filtered elements.

5) next(iter): The next function returns the next element in the iterable object.

```
# all function
print(all([i>2 for i in [1,2,3,4,5]]))
# any function
print(any([i>2 for i in [1,2,3,4,5]]))
# map function
def func(x):
    return x > 3
print(map(func,[1,2,3,4,5]))
for i in map(func,[1,2,3,4,5]):
    print(i,end=')t')
# filter function
print(filter(func,[1,2,3,4,5]))
for i in filter(func,[1,2,3,4,5]):
    print(i,end='It')
# next function
iterl =iter([1,2,3])
for i in range(3):
    print(next(iterl), end='It')
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
False
True
<map object at 0x7f0d38ddd6d0>
False )tFalse )tFalse )tTrue )tTrue )t<filter object at 0x7f0d38ddd6d0>
4 It5 It1 It2 It3 It
```

6) zip(*iter): According to multiple different iterators, the aggregation of elements in corresponding position is performed, and a new iterator is returned.

```
# zip function
for a,b in zip([1,2,3,4,5],"abcde"):
    print(a,b)
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.p
1 a
2 b
3 c
4 d
5 e
```

7) reduce(func,sequence[,initial]): Python3.x中需要从functools中导入reduce函数再使用。The two-argument function func is applied iteratively to each element in the sequence seq from left to right, and finally returns a single value as the result. It is a built-in function in Python2.x, and in Python3.x, you need to import the reduce function from functools first.

```
import functools
def add(x,y):
    #Add two numbers
    return x+y
suml = functools.reduce(add,[1,2,3,4,5])#Sum the lists:1+2+3+4+5
sum2 = functools.reduce(lambda x,y:x+y,[1,2,3,4,5])
print(suml)
print(sum2)
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
15
15
```

2.7. Object Element Operation

1) help(object): The help function can help users to query the information of different objects, including built-in methods, properties and other information.

2) id(object): Returns the identity value of the object, which is an integer and remains unique in the object's life cycle.

```
# help function
help(int)
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
Help on class int in module builtins:

class int(object)
 |  int([x]) -> integer
 |  int(x, base=10) -> integer
```

```
# id function
num = 10
print(id(num))
```

3) hash(object): If the object has a corresponding hash value, the corresponding hash value is returned.

4) type(object): The type function is used to return the type of the object.

5) dir(object): If there is no argument object, the dir function returns a list of names in the current local scope. If there is an argument object, the function attempts to return a list of valid properties for that object

6) len(object): Return the length of the object or the number of elements it contains.

```
# hash function
print(hash("python"))
# type function
str1 = "python"
list1 = [1,2]
print(type(str1))
print(type(list1))
# dir function
print(dir())
# len function
list2 = [1,2,3,4]
str2 = "ererdfff"
print(len(list2))
print(len(str2))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
-2607102823527459755
<class 'str'>
<class 'list'>
['__annotations__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'list1', 'str1']
4
8
```

7) repr(object): The repr function returns a string containing a printable representation of an object.

8) ascii(object): The ascii function is similar to the repr function, the ascii function returns a string containing a printable representation of an object. However unlike the repr function, ascii() escape non-ASCII encoded characters.

9) format(value ,format_spec): format function converts value into a "formatted" representation controlled by the format_spec parameter, and is mostly used in string formatting.

10)vars(object): The function returns objects with a **_dict__** attribute, such as modules, classes, instances, etc.

```
# repr function
print(repr([1,2,3,4]))
print(repr("python"))
# ascli function
print(ascii([1,2,3,4]))
print(ascii("python"))
print("python ()".format(123))
#  Output in octal 123
print(format(123,'o'))
#Output in hexadecimal
print(format(123,'x'))
# vars function
class A:
    def __init__(self):
        self.name = "python"
        self.age =10
print(vars(A()))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
[1, 2, 3, 4]
'python'
[1, 2, 3, 4]
'python'
python ()
173
7b
{}
```

2.8. Property Operation

1) isinstance(object, classinfo): The function is used to determine whether the object belongs to the type of classinfo, if it is, it returns True, otherwise it returns False

2) issubclass(class, classinfo): If class is a child class of classinfo class, the function returns True, otherwise it returns False.

```
# isinstance function
print(isinstance("python", str))
print(isinstance("python", int))
# issubclassa function
class A:
    def __init__(self):
        pass
class B:
    def __init__(self):
        pass
print(issubclass(B,A))
print(issubclass(int,object))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
True
False
True
True

进程已结束,退出代码0
```

3) hasattr(object, name): Return True if the name string is the name of one of the properties of the object object, False otherwise.

4) getattr(object, name): Return the value of the named property of the object. name must be a string. If the string name is one of the properties of the object, the value of that property is returned.

5) setattr(object, name, value): The string name refers to an existing property or a new property of the object. As long as the setattr operation is allowed by the object, the function will assign the value to the attribute.

6) delattr(object, name): The string name must be the name of an property of the object. If the deleting property is allowed by object, the delattr function will delete the specified name property.

7) **import(name)**: import modules dynamically

8) callable(object): If object can be called, True will be returned, otherwise False.

```
# hasattr,getattr,setattr delatr function
class A:
    def __init__(self):
        self.name ="python"
        self.age = 10
        self.sex = "male"
myclass= A()
print(hasattr(myclass,"name"))
print(getattr(myclass,"age"))
setattr(myclass,"num",12345)
print(vars(myclass))
delattr(myclass,"name")
print(vars(myclass))
# __import__ function
np = __import__("numpy")
print(np._version)
# callable function
print(callable(A))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
True
10
{'name': 'python', 'age': 10, 'sex': 'male', 'num': 12345}
{'age': 10, 'sex': 'male', 'num': 12345}
<module 'numpy._version' from '/home/lin/.local/lib/python3.8/site-packages/numpy/_version.py'>
True
```

2.9. Variable Operation

1) globals(): Return a dictionary composed of global variables and their values in the scope

2) locals(): Return a dictionary consisting of local variables and their values in the current scope.

```
# globals function
print(globals())
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <_frozen_importlib_external.SourceFil
```

```
# celocals function
def fune():
    local_varl =123
    ocal_var2 = "python"
    print(locals())
fune()
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
{'local_varl': 123, 'ocal_var2': 'python'}
```

2.10 Human-computer Operation

1) print(*objects, end='\n', file=sys.stdout): Print the contents of objects to the text stream specified by file, and end with end.

2) input(): read the value input by user

3) open(file, mode='r'): Open the file and return the corresponding file object, mode corresponds to the operation of reading or writing to the file. If the

file cannot be opened, the program will throw OSError.

```
# print function
print("python")
print(124)
# input function
input("Please enter a value:")
# open function
with open("a.txt",'w') as fw:
    fw.write("python")
    fw.write("in")
with open("a.txt",'r') as fr:
    print(fr.readlines())
```

```
python
124
Please enter a value:123
['pythonin']
```

2.11 Compilation Operation

1) compile(source,, mode): The function compiles the source into code which is then executed by the exec function or the eval function. The mode parameter indicates the mode adopted in compiling the code.

2) exec(object): The exec function supports dynamic execution of Python code, where object must be a string or a code object. If it is a string, the string will be parsed into a Python statement to execute, and if the two u fruit is coded, it will be executed directly.

3) eval(expression): The eval function evaluates the string expression as a valid expression and returns the result. The eval function can only evaluate a single expression, not complex code logic or assignment operation.

```
# complie,exec function
str_code = "print([i**2 for i in range (5) ])"
compilel = compile(str_code, '', mode="exec")
exec(compilel)
# eval function
print(eval("12*34+4"))
print(eval("float (200)"))
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
[0, 1, 4, 9, 16]
412
200.0
```

2.12 Decorator Function

1) classmethod(): It is used to to encapsulate a method into a class method, and the method modified by the decorator can be called without creating a class object.

2) staticmethod(): The staticmethod method is to convert the method in the class into a static method. The static method does not accept implicit parameters, and the static method can also be called without creating a class object.

3) property(): As a decorator, property can convert class methods into class properties for use.

```python
class MyClass(object):
    district = "BeiJing"

    def __init__(self, firstname="Joe", lastname="Harris"):
        self.firstname = firstname
        self.lastname = lastname
        self.age = 20

    # property,classmethod,staticmethod函数
    @property
    def fullName(self):
        print("ful1Name: ().  1)".format(self.firstname, self.lastname))

    @classmethod
    def showDistrict(cls):
        print("I an in ".format(cls.district))

    @staticmethod
    def showStaticmethod():
        print("this is showStaticmethod")


MyClass("hah", "lala").fullName
MyClass.showDistrict()
MyClass.showStaticmethod()
```

```
/usr/bin/python3.8 /home/lin/PycharmProjects/pythonProject/test.py
ful1Name: ().  1)
I an in
this is showStaticmethod
```