Information

Module 1 : Introduction to Python

1.1 Elementary Data Types

(1) Data Type

- Number :
  - Integer : x = 4
  - Long integer : x = 15L
  - Floating point : x = 3.142
  - Boolean : x = True
- Text :
  - Character : x = 'c'
  - String : x = "this"

```
# Data Type Example

x = 4 # integer
print(x,type(x))

y = True # boolean(True, False)
print(y, type(y))

z = 3.7  # floating point
print(z,type(z))

s = "This is a string" # string
print(s,type(s))
```

```
4 <class 'int'>
True <class 'bool'>
3.7 <class 'float'>
This is a string <class 'str'>
```

(2) Arithmetic Operations

```
# The arithmetic operations available
# for manipulating integers and floating point numbers

x = 4 # integer
```

```
x1 = x + 4 # addition
x2 = x * 3 # multiplication
x += 2 # equivalent to x = x + 2
x3 = x
x *= 3 # equivalent to x = x * 3
x4 = x
x5 = x % 4 # modulo (remainder) operator

z = 3.7 # floating point number
z1 = z - 2 # subtraction
z2 = z / 3 # division
z3 = z // 3 # integer division
z4 = z ** 2 # square of z
z5 = z4 ** 0.5 # square root
z6 = pow(z,2) # equivalent to square of z
z7 = round(z) # rounding z to its nearest integer
z8 = int(z) # type casting float to int

print(x,x1,x2,x3,x4,x5)
print(z,z1,z2,z3,z4)
print(z5,z6,z7,z8)
```

```
18 8 12 6 18 2
3.7 1.7000000000000002 1.2333333333333334 1.0 13.690000000000001
3.7 13.690000000000001 4 3
```

(3) Math Module

The following are some of the functions provided by the math module for integers and floating point numbers

Functions

- sqrt() : squre function
- pow() : power function , first parameter : base number, second parameter : n power
- exp(): the power of e = 2.71828
- log() : Returns the natural logarithm of first parameter. Second parameter is base, default base is e.
- fabs() : absolute value.
- factorial() : calculate the factorial.
- ceil() : ceiling function
- floor() : floor function
- trunc() : truncate function
- sin() : sin function
- tanh(): arctan function
- isnan(): check if nan type
- isinf():check if infinity type

Attribute

- pi : pi = 3.141592653...
- nan : not a number
- inf : infinity

```python
import math

x = 4
print(math.sqrt(x)) # sqrt(4) = 2
print(math.pow(x,2)) # 4**2 = 16
print(math.exp(x)) # exp(4) = 54.6
print(math.log(x,2)) # log based 2 (default is natural logarithm)
print(math.fabs(-4)) # absolute value
print(math.factorial(x)) # 4! = 4 x 3 x 2 x 1 = 24

z = 0.2
print(math.ceil(z)) # ceiling function
print(math.floor(z)) # floor function
print(math.trunc(z)) # truncate function

z = 3*math.pi # math.pi = 3.141592653589793
print(math.sin(z)) # sine function
print(math.tanh(z)) # arctan function

x = math.nan # not a number
print(math.isnan(x))

x = math.inf # infinity
print(math.isinf(x))
```

```
2.0
16.0
54.598150033144236
2.0
4.0
24
1
0
0
3.6739403974420594e-16
0.9999999869751758
True
True
```

(4) Logical Operations

- and : Logical AND
- or : Logical OR
- and not : Logical NOT

```
y1 = True
y2 = False

print(y1 and y2) # logical AND
print(y1 or y2) # logical OR
print(y1 and not y2) # logical NOT
```

```
False
True
True
```

(5) Manipulating Strings

Function

- len() : return the size of string
- uppder():convert to upper case
- lower():convert to lower case
- split():split the string, based on parameter
- replace():replace the 1st. parameter with 2nd. parameter
- find():find the position of parameter in string

Operation

- [:] : slice
- o : casting two string
- in : check if the 1st. string is a substring of 2nd. one
- == : equality comparison
- < : inequality comparison
- o : string concatenation
- o : replicate the string n times

```
s1 = "This"

print(s1[1:]) # print last three characters
print(len(s1)) # get the string length
print("Length of string is " + str(len(s1))) # type casting int to str
print(s1.upper()) # convert to upper case
print(s1.lower()) # convert to lower case

s2 = "This is a string"

words = s2.split(' ') # split the string into words
print(words[0])
print(s2.replace('a','another')) # replace "a" with "another"
print(s2.replace('is','at')) # replace "is" with "at"
print(s2.find("a")) # find the position of "a" in s2
print(s1 in s2) # check if s1 is a substring of s2

print(s1 == 'This') # equality comparison
```

```
print(s1 < 'That') # inequality comparison
print(s2 + " too") # string concatenation
print((s1 + " ")* 3) # replicate the string 3 times
```

```
his
4
Length of string is 4
THIS
this
This
This is another string
That at a string
8
True
True
False
This is a string too
This This This
```

1.2 Compound Data Types

The following examples show how to create and manipulate a list object.

(1) List

Function

- range(startvalue,endvalue,stepsize): get the numbers from startvalue to endvalue based on stepsize length.
- append():insert parameter to end of the list
- pop():remove last element of the list
- len() : get the size
- sum() : sums up elements of the list
- insert() : insert 2nd. parameter at index 1st.
- sort():sort elements in list; parameter : reverse = True - in descending order; False(default) - in upscending order.
- remove(): remove the parameter in the list.

Operation

- + :concatenate two list
- * : replicate the list with n times

```
intlist = [1, 3, 5, 7, 9]
print(type(intlist))
print(intlist)
intlist2 = list(range(0,10,2)) # range[startvalue, endvalue, stepsize]
print(intlist2)

```

```
print(intlist[2]) # get the third element of the list
print(intlist[:2]) # get the first two elements
print(intlist[2:]) # get the last three elements of the list
print(len(intlist)) # get the number of elements in the list
print(sum(intlist)) # sums up elements of the list

intlist.append(11) # insert 11 to end of the list
print(intlist)
print(intlist.pop()) # remove last element of the list
print(intlist)
print(intlist + [11,13,15]) # concatenate two lists
print(intlist * 3) # replicate the list
intlist.insert(2,4) # insert item 4 at index 2
print(intlist)
intlist.sort(reverse=True) # sort elements in descending order
print(intlist)
```

```
<class 'list'>
[1, 3, 5, 7, 9]
[0, 2, 4, 6, 8]
5
[1, 3]
[5, 7, 9]
5
25
[1, 3, 5, 7, 9, 11]
11
[1, 3, 5, 7, 9]
[1, 3, 5, 7, 9, 11, 13, 15]
[1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
[1, 3, 4, 5, 7, 9]
[9, 7, 5, 4, 3, 1]
```

(2) String

Function

- join():merge all elements of the list(string type) into a string

```
mylist = ['this', 'is', 'a', 'list']
print(mylist)
print(type(mylist))

print("list" in mylist) # check whether "list" is in mylist
print(mylist[2]) # show the 3rd element of the list
print(mylist[:2]) # show the first two elements of the list
print(mylist[2:]) # show the last two elements of the list
mylist.append("too") # insert element to end of the list

separator = " "
print(separator.join(mylist)) # merge all elements of the list into a string

```

```
mylist.remove("is") # remove element from list
print(mylist)
```

```
['this', 'is', 'a', 'list']
<class 'list'>
True
a
['this', 'is']
['a', 'list']
this is a list too
['this', 'a', 'list', 'too']
```

(3) Dictionary

Function

- keys():get the keys of the dictionary
- values():get the values of the dictionary
- len():get number of key-value pairs
- get():get the value of the parameter
- zip():contract two list inot a dictionary, 1st. list is key, 2nd. list is values
- sorted():
    - need import itemgetter module in operator package;
    - 1st. parameter:list.items();
    - 2nd. parameter: key = itemgetter(0) : sort by key of dictionary; itemgetter(1) : sort by value of dictionary

Operation

- in : check key if in dictionary

```
abbrev = {}
abbrev['MI'] = "Michigan"
abbrev['MN'] = "Minnesota"
abbrev['TX'] = "Texas"
abbrev['CA'] = "California"

print(abbrev)
print(abbrev.keys()) # get the keys of the dictionary
print(abbrev.values()) # get the values of the dictionary
print(len(abbrev)) # get number of key-value pairs

print(abbrev.get('MI'))
print("FL" in abbrev)
print("CA" in abbrev)

keys = ['apples', 'oranges', 'bananas', 'cherries']
values = [3, 4, 2, 10]
fruits = dict(zip(keys, values))
```

```
print(fruits)
print(sorted(fruits)) # sort keys of dictionary

from operator import itemgetter
print(sorted(fruits.items(), key=itemgetter(0))) # sort by key of dictionary
print(sorted(fruits.items(), key=itemgetter(1))) # sort by value of dictionary
```

```
{'MI': 'Michigan', 'MN': 'Minnesota', 'TX': 'Texas', 'CA': 'California'}
dict_keys(['MI', 'MN', 'TX', 'CA'])
dict_values(['Michigan', 'Minnesota', 'Texas', 'California'])
4
Michigan
False
True
{'apples': 3, 'oranges': 4, 'bananas': 2, 'cherries': 10}
['apples', 'bananas', 'cherries', 'oranges']
[('apples', 3), ('bananas', 2), ('cherries', 10), ('oranges', 4)]
[('bananas', 2), ('apples', 3), ('oranges', 4), ('cherries', 10)]
```

```
MItuple = ('MI', 'Michigan', 'Lansing')
CAtuple = ('CA', 'California', 'Sacramento')
TXtuple = ('TX', 'Texas', 'Austin')

print(MItuple)
print(MItuple[1:])

states = [MItuple, CAtuple, TXtuple] # this will create a list of tuples
print(states)
print(states[2])
print(states[2][:])
print(states[2][1:])

states.sort(key=lambda state: state[2]) # sort the states by their capital cities
print(states)
```

```
('MI', 'Michigan', 'Lansing')
('Michigan', 'Lansing')
[('MI', 'Michigan', 'Lansing'), ('CA', 'California', 'Sacramento'), ('TX',
'Texas', 'Austin')]
('TX', 'Texas', 'Austin')
('TX', 'Texas', 'Austin')
('Texas', 'Austin')
[('TX', 'Texas', 'Austin'), ('MI', 'Michigan', 'Lansing'), ('CA', 'California',
'Sacramento')]
```

1.3 Control Flow Statements

- if
- for
- while

(1) if

```python
# using if-else statement

x = 10

if x % 2 == 0:
    print("x =", x, "is even")
else:
    print("x =", x, "is odd")

if x > 0:
    print("x =", x, "is positive")
elif x < 0:
    print("x =", x, "is negative")
else:
    print("x =", x, "is neither positive nor negative")
```

```
x = 10 is even
x = 10 is positive
```

(2) loop

```python
# using for loop with a list

mylist = ['this', 'is', 'a', 'list']
for word in mylist:
    print(word.replace("is", "at"))

mylist2 = [len(word) for word in mylist] # number of characters in each word
print(mylist2)

# using for loop with list of tuples

states = [('MI', 'Michigan', 'Lansing'),('CA', 'California', 'Sacramento'),
 ('TX', 'Texas', 'Austin')]
sorted_capitals = [state[2] for state in states]
sorted_capitals.sort()
print(sorted_capitals)

# using for loop with dictionary

fruits = {'apples': 3, 'oranges': 4, 'bananas': 2, 'cherries': 10}
fruitnames = [k for (k,v) in fruits.items()]
print(fruitnames)
```

```
that
at
a
latt
[4, 2, 1, 4]
['Austin', 'Lansing', 'Sacramento']
['apples', 'oranges', 'bananas', 'cherries']
```

(3) while

```
# using while loop
mylist = list(range(-10,10))
print(mylist)

i = 0
while (mylist[i] < 0):
    i = i + 1

print("First non-negative number:", mylist[i])
```

```
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
First non-negative number: 0
```

1.4 User-Defined Functions

- Named : like other language,define the user function
- Unnamed : lambda keyword

Lambda Keyword

```
myfunc = lambda x: 3*x**2 - 2*x + 3 # example of an unnamed quadratic function
print(myfunc(2))
```

```
11
```

```
import math

# The following function will discard missing values from a list
def discard(inlist, sortFlag=False): # default value for sortFlag is False
    outlist = []
    for item in inlist:
        if not math.isnan(item):
            outlist.append(item)
    if sortFlag:
        outlist.sort()

    return outlist
```

```

mylist = [12, math.nan, 23, -11, 45, math.nan, 71]

print(discard(mylist,True))
```

```
[-11, 12, 23, 45, 71]
```

1.5 File I/O

write data from a list or other objects to a file.

```
states = [('MI', 'Michigan', 'Lansing'),('CA', 'California', 'Sacramento'),
 ('TX', 'Texas', 'Austin'), ('MN', 'Minnesota', 'St Paul')]

with open('states.txt', 'w') as f:
    f.write('\n'.join('%s,%s,%s' % state for state in states))

with open('states.txt', 'r') as f:
    for line in f:
        fields = line.split(sep=',') # split each line into its respective fields
        print('State=',fields[1],'(',fields[0],')','Capital:', fields[2])
```

```
State= Michigan ( MI ) Capital: Lansing

State= California ( CA ) Capital: Sacramento

State= Texas ( TX ) Capital: Austin

State= Minnesota ( MN ) Capital: St Paul
```