

# Information

- Title : 1.Introduction to Python 수행한 결과를 캡처한 파일
- ID : 12164484
- Name : 송해소
- Time : 2021.03.11

## 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"

```
1 # Data Type Example
2
3 x = 4 # integer
4 print(x,type(x))
5
6 y = True # boolean(True, False)
7 print(y, type(y))
8
9 z = 3.7 # floating point
10 print(z,type(z))
11
12 s = "This is a string" # string
13 print(s,type(s))
```

```
1 4 <class 'int'>
2 True <class 'bool'>
3 3.7 <class 'float'>
4 This is a string <class 'str'>
```

#### (2) Arithmetic Operations

```
1 # The arithmetic operations available
2 # for manipulating integers and floating point numbers
3
4 x = 4 # integer
```

```

5  x1 = x + 4 # addition
6  x2 = x * 3 # multiplication
7  x += 2 # equivalent to x = x + 2
8  x3 = x
9  x *= 3 # equivalent to x = x * 3
10 x4 = x
11 x5 = x % 4 # modulo (remainder) operator
12
13 z = 3.7 # floating point number
14 z1 = z - 2 # subtraction
15 z2 = z / 3 # division
16 z3 = z // 3 # integer division
17 z4 = z ** 2 # square of z
18 z5 = z4 ** 0.5 # square root
19 z6 = pow(z,2) # equivalent to square of z
20 z7 = round(z) # rounding z to its nearest integer
21 z8 = int(z) # type casting float to int
22
23 print(x,x1,x2,x3,x4,x5)
24 print(z,z1,z2,z3,z4)
25 print(z5,z6,z7,z8)

```

```

1  18 8 12 6 18 2
2  3.7 1.7000000000000002 1.2333333333333334 1.0 13.690000000000001
3  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()` : square 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 :  $\pi = 3.141592653\dots$
- nan : not a number
- inf : infinity

```
1 import math
2
3 x = 4
4 print(math.sqrt(x)) # sqrt(4) = 2
5 print(math.pow(x,2)) # 4**2 = 16
6 print(math.exp(x)) # exp(4) = 54.6
7 print(math.log(x,2)) # log based 2 (default is natural logarithm)
8 print(math.fabs(-4)) # absolute value
9 print(math.factorial(x)) # 4! = 4 x 3 x 2 x 1 = 24
10
11 z = 0.2
12 print(math.ceil(z)) # ceiling function
13 print(math.floor(z)) # floor function
14 print(math.trunc(z)) # truncate function
15
16 z = 3*math.pi # math.pi = 3.141592653589793
17 print(math.sin(z)) # sine function
18 print(math.tanh(z)) # arctan function
19
20 x = math.nan # not a number
21 print(math.isnan(x))
22
23 x = math.inf # infinity
24 print(math.isinf(x))
```

```
1 2.0
2 16.0
3 54.598150033144236
4 2.0
5 4.0
6 24
7 1
8 0
9 0
10 3.6739403974420594e-16
11 0.9999999869751758
12 True
13 True
```

## (4) Logical Operations

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

```

1 y1 = True
2 y2 = False
3
4 print(y1 and y2) # logical AND
5 print(y1 or y2) # logical OR
6 print(y1 and not y2) # logical NOT

```

```

1 False
2 True
3 True

```

## (5) Manipulating Strings

### Function

- `len()` : return the size of string
- `upper()`: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

```

1 s1 = "This"
2
3 print(s1[1:]) # print last three characters
4 print(len(s1)) # get the string length
5 print("Length of string is " + str(len(s1))) # type casting int to str
6 print(s1.upper()) # convert to upper case
7 print(s1.lower()) # convert to lower case
8
9 s2 = "This is a string"
10
11 words = s2.split(' ') # split the string into words
12 print(words[0])
13 print(s2.replace('a','another')) # replace "a" with "another"
14 print(s2.replace('is','at')) # replace "is" with "at"
15 print(s2.find("a")) # find the position of "a" in s2
16 print(s1 in s2) # check if s1 is a substring of s2
17
18 print(s1 == 'This') # equality comparison

```

```

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

```

```

1 his
2 4
3 Length of string is 4
4 THIS
5 this
6 This
7 This is another string
8 That at a string
9 8
10 True
11 True
12 False
13 This is a string too
14 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

```

1 intlist = [1, 3, 5, 7, 9]
2 print(type(intlist))
3 print(intlist)
4 intlist2 = list(range(0,10,2)) # range[startvalue, endvalue, stepsize]
5 print(intlist2)
6

```

```

7 print(intlist[2]) # get the third element of the list
8 print(intlist[:2]) # get the first two elements
9 print(intlist[2:]) # get the last three elements of the list
10 print(len(intlist)) # get the number of elements in the list
11 print(sum(intlist)) # sums up elements of the list
12
13 intlist.append(11) # insert 11 to end of the list
14 print(intlist)
15 print(intlist.pop()) # remove last element of the list
16 print(intlist)
17 print(intlist + [11,13,15]) # concatenate two lists
18 print(intlist * 3) # replicate the list
19 intlist.insert(2,4) # insert item 4 at index 2
20 print(intlist)
21 intlist.sort(reverse=True) # sort elements in descending order
22 print(intlist)

```

```

1 <class 'list'>
2 [1, 3, 5, 7, 9]
3 [0, 2, 4, 6, 8]
4 5
5 [1, 3]
6 [5, 7, 9]
7 5
8 25
9 [1, 3, 5, 7, 9, 11]
10 11
11 [1, 3, 5, 7, 9]
12 [1, 3, 5, 7, 9, 11, 13, 15]
13 [1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
14 [1, 3, 4, 5, 7, 9]
15 [9, 7, 5, 4, 3, 1]

```

## (2) String

### Function

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

```

1 mylist = ['this', 'is', 'a', 'list']
2 print(mylist)
3 print(type(mylist))
4
5 print("list" in mylist) # check whether "list" is in mylist
6 print(mylist[2]) # show the 3rd element of the list
7 print(mylist[:2]) # show the first two elements of the list
8 print(mylist[2:]) # show the last two elements of the list
9 mylist.append("too") # insert element to end of the list
10
11 separator = " "
12 print(separator.join(mylist)) # merge all elements of the list into a string
13

```

```

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

```

```

1 ['this', 'is', 'a', 'list']
2 <class 'list'>
3 True
4 a
5 ['this', 'is']
6 ['a', 'list']
7 this is a list too
8 ['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 into 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

```

1 abbrev = {}
2 abbrev['MI'] = "Michigan"
3 abbrev['MN'] = "Minnesota"
4 abbrev['TX'] = "Texas"
5 abbrev['CA'] = "California"
6
7 print(abbrev)
8 print(abbrev.keys()) # get the keys of the dictionary
9 print(abbrev.values()) # get the values of the dictionary
10 print(len(abbrev)) # get number of key-value pairs
11
12 print(abbrev.get('MI'))
13 print("FL" in abbrev)
14 print("CA" in abbrev)
15
16 keys = ['apples', 'oranges', 'bananas', 'cherries']
17 values = [3, 4, 2, 10]
18 fruits = dict(zip(keys, values))

```

```

19 print(fruits)
20 print(sorted(fruits)) # sort keys of dictionary
21
22 from operator import itemgetter
23 print(sorted(fruits.items(), key=itemgetter(0))) # sort by key of dictionary
24 print(sorted(fruits.items(), key=itemgetter(1))) # sort by value of dictionary

```

```

1  {'MI': 'Michigan', 'MN': 'Minnesota', 'TX': 'Texas', 'CA': 'California'}
2  dict_keys(['MI', 'MN', 'TX', 'CA'])
3  dict_values(['Michigan', 'Minnesota', 'Texas', 'California'])
4  4
5  Michigan
6  False
7  True
8  {'apples': 3, 'oranges': 4, 'bananas': 2, 'cherries': 10}
9  ['apples', 'bananas', 'cherries', 'oranges']
10 [('apples', 3), ('bananas', 2), ('cherries', 10), ('oranges', 4)]
11 [('bananas', 2), ('apples', 3), ('oranges', 4), ('cherries', 10)]

```

```

1  MItuple = ('MI', 'Michigan', 'Lansing')
2  CATuple = ('CA', 'California', 'Sacramento')
3  TXtuple = ('TX', 'Texas', 'Austin')
4
5  print(MItuple)
6  print(MItuple[1:])
7
8  states = [MItuple, CATuple, TXtuple] # this will create a list of tuples
9  print(states)
10 print(states[2])
11 print(states[2][:])
12 print(states[2][1:])
13
14 states.sort(key=lambda state: state[2]) # sort the states by their capital cities
15 print(states)

```

```

1  ('MI', 'Michigan', 'Lansing')
2  ('Michigan', 'Lansing')
3  [('MI', 'Michigan', 'Lansing'), ('CA', 'California', 'Sacramento'), ('TX',
   'Texas', 'Austin')]
4  ('TX', 'Texas', 'Austin')
5  ('TX', 'Texas', 'Austin')
6  ('Texas', 'Austin')
7  [('TX', 'Texas', 'Austin'), ('MI', 'Michigan', 'Lansing'), ('CA', 'California',
   'Sacramento')]

```

## 1.3 Control Flow Statements

---

- if
- for
- while



## (1) if

```
1 # using if-else statement
2
3 x = 10
4
5 if x % 2 == 0:
6     print("x =", x, "is even")
7 else:
8     print("x =", x, "is odd")
9
10 if x > 0:
11     print("x =", x, "is positive")
12 elif x < 0:
13     print("x =", x, "is negative")
14 else:
15     print("x =", x, "is neither positive nor negative")
```

```
1 x = 10 is even
2 x = 10 is positive
```

## (2) loop

```
1 # using for loop with a list
2
3 mylist = ['this', 'is', 'a', 'list']
4 for word in mylist:
5     print(word.replace("is", "at"))
6
7 mylist2 = [len(word) for word in mylist] # number of characters in each word
8 print(mylist2)
9
10 # using for loop with list of tuples
11
12 states = [('MI', 'Michigan', 'Lansing'), ('CA', 'California', 'Sacramento'),
13           ('TX', 'Texas', 'Austin')]
14 sorted_capitals = [state[2] for state in states]
15 sorted_capitals.sort()
16 print(sorted_capitals)
17
18 # using for loop with dictionary
19
20 fruits = {'apples': 3, 'oranges': 4, 'bananas': 2, 'cherries': 10}
21 fruitnames = [k for (k,v) in fruits.items()]
22 print(fruitnames)
```

```

1 that
2 at
3 a
4 latt
5 [4, 2, 1, 4]
6 ['Austin', 'Lansing', 'Sacramento']
7 ['apples', 'oranges', 'bananas', 'cherries']

```

### (3) while

```

1 # using while loop
2 mylist = list(range(-10,10))
3 print(mylist)
4
5 i = 0
6 while (mylist[i] < 0):
7     i = i + 1
8
9 print("First non-negative number:", mylist[i])

```

```

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

```

## 1.4 User-Defined Functions

---

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

### Lambda Keyword

```

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

```

```

1 11

```

```

1 import math
2
3 # The following function will discard missing values from a list
4 def discard(inlist, sortFlag=False): # default value for sortFlag is False
5     outlist = []
6     for item in inlist:
7         if not math.isnan(item):
8             outlist.append(item)
9     if sortFlag:
10         outlist.sort()
11
12     return outlist

```

```
13
14 mylist = [12, math.nan, 23, -11, 45, math.nan, 71]
15
16 print(discard(mylist,True))
```

```
1 [-11, 12, 23, 45, 71]
```

## 1.5 File I/O

---

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

```
1 states = [('MI', 'Michigan', 'Lansing'),('CA', 'California', 'Sacramento'),
2           ('TX', 'Texas', 'Austin'), ('MN', 'Minnesota', 'St Paul')]
3
4 with open('states.txt', 'w') as f:
5     f.write('\n'.join('%s,%s,%s' % state for state in states))
6
7 with open('states.txt', 'r') as f:
8     for line in f:
9         fields = line.split(sep=',') # split each line into its respective fields
10        print('State=',fields[1],('(',fields[0],'),'','Capital:', fields[2]))
```

```
1 State= Michigan ( MI ) Capital: Lansing
2
3 State= California ( CA ) Capital: Sacramento
4
5 State= Texas ( TX ) Capital: Austin
6
7 State= Minnesota ( MN ) Capital: St Paul
```