Lab.1 Python Programming Basics

1) Data Type: Number

You need to be familiar with the basic operations of numbers in Python. Note that Boolean operations in Python use the keyword and/or/not instead of the operator.

print(True+False) # The output is **1**. By default, **True** indicates 1, and **False** indicates 0.

print(True or False) # If **True** is displayed, enter **or** or perform the OR operation.

print(5//2) # The output is **2**, and // is the rounding operator.

print(5%2) # The output is **1**, and % is the modulo operator.

print(32)** # The output is **9**, and ** indicates the power operation.

print(5+1.6) # The output is **6.6**. By default, the sum of numbers of different precisions is the number of the highest precision type.

2) Data Type: Character String

S = 'python' # Assign value python to variable S.

len(obj): Return the object length.

print(len(S)) # Output: 6

print(S[0], S[1], S[-1]) # The output is pyn. Elements are obtained by index.

print (S+' 1', S*2) # The output is python1 pythonpython, which indicates mergence and duplication.

3) Data Type: List

- **Common operations on lists:**

```
animals = ['cat', 'dog', 'monkey']
animals.append('fish')                  # Append an element.
print(animals)                          # Output: ['cat', 'dog', 'monkey', 'fish']
animals.remove('fish')                  # Delete element fish.
print(animals)                          # Output: ['cat', 'dog', 'monkey']
animals.insert (1, 'fish')              # Insert element fish at subscript 1.
print(animals)                          # Output: ['cat', 'fish', 'dog', 'monkey']
                                        # list.pop([index=-1]): Remove the element (the last element by default) corresponding to the subscript in the list. The index indicates the subscript.

animals.pop(1)                          # Delete the element whose subscript is 1.
print(animals)                          # Output: ['cat', 'dog', 'monkey']


# Traverse and obtain the elements and indexes.

for i in enumerate(animals):
    print(i)            # Index consisting of the element subscript and element

Output: (0, cat)
        (1, dog)
        (2, monkey)

# List derivation.

list1 = [12,45,32,55]
list1.sort()                            # Sort the list.
print(list1)                            # Output: [12,32,45,55]

list1.reverse()                         # Reverse the list.
print(list1)                            # Output: [55,45,32,12]
```

4) Data Type: Tuple

- **Common operations on tuples:**

```
T=(1,2,3)                          # Create a tuple.
print(T+(4,5))                     # Combine tuples. The output is (1, 2, 3, 4, 5).
t=(42,)              # A tuple with only one element, which is different from a number.

tuple1 = (12,45,32,55,[1,0,3])             # Create a tuple.
tuple1[0] = "good"          # The program is abnormal, and the tuple is unchangeable.
tuple1[4][0] = 2            # Elements that can be changed in a tuple are changeable.
print(tuple1)               # (12,45,32,55,[2,0,3])
```

5) Data Type: Dictionary

- **Common operations on dictionaries:**

```
# Three value assignment operations on dictionaries.
x = {'food':'Spam' , 'quantity':4 , 'color':'pink'}

X =dict(food='Spam' , quantity=4, color='pink')

x = dict([("food", "Spam"),("quantity", "4"),("color","pink")])

d =x.copy()
d['color'] = 'red'
print(x)                                # {'food':'Spam','quantity':4,'color':'pink'}
print(d)                                # {'food':'Spam','quantity':4,'color':'red'}

# Element access.
print (d ['name'])                      # Obtain the error information.
print(d.get('name'))                    # Output: None

# Output: The key value does not exist.
print(d.get('name','The key value does not exist.'))

print(d.keys())                         # Output: dict_keys(['food', 'quantity', 'color'])
print(d.values())                       # Output: dict_values(['Spam', 4, 'red'])

# Output: dict_items([('food', 'Spam'), ('quantity', 4), ('color', 'red')])
print(d.items())
```

d.clear() # Clear all data in the dictionary.
print(d) # Output: {}
del(d) # Delete the dictionary.
print(d) # The program is abnormal, and a message is displayed, indicating that d is not defined.

6) Data Type: Set

- **Common operations on sets:**

sample_set = {'Prince', 'Techs'}

The output is False. in is used to check whether an element exists in the set.
print('Data' in sample_set)

sample_set.add('Data') # Add element Data to the set.
print(sample_set) # Output: {'Prince', 'Techs', 'Data'}
print(len(sample_set)) # Output: 3

sample_set.remove('Data') # Delete element Data.
print(sample_set) # {'Prince', 'Techs'}

list2 = [1,3,1,5,3]
print(list(set(list2))) **# The output is [1,3,5]. The uniqueness of the set elements is used to deduplicate the list.**

sample_set = frozenset(sample_set) # Unchangeable set.

7) Deep Copy and Shallow Copy

The copy module in Python is used to implement deep copy.

import copy
Dict1 = {'name':'lee', 'age':89, 'num':[1,2,8]} # Create a dictionary.
Dict_copy = Dict1.copy() # Shallow copy.

Dict_dcopy = copy.deepcopy(Dict1) # Deep copy.
Dict1['num'][1] = 6 # Change the value of the nested list in the raw data.
print('Dict1:'+str(Dict1)+"\n",' Dict_copy:'+ str(Dict_copy)+"\n",' Dict_dcopy:'+ str(Dict_dcopy))

Output:

Dict1:{‘name’:’lee’, ‘age’:89, ‘num’:[1,6,8]}

Dict_copy :{'name':'lee', 'age':89, 'num':[1,6,8]} **# The shallow copy data is modified.**

Dict_dcopy :{'name':'lee', 'age':89, 'num':[1,2,8]} **# The deep copy data is not modified.**

8) if Statement

You can use "if statement" in Python to determine the level of a score input by a user.

#The input function receives input, which is a character string.

```
score = input("Please enter your score.")
```

try:… except Exception:… is a Python statement used to capture exceptions. If an error occurs in the statement in the try statement, the except statement will be executed.

```
try:
    score = float(score)          # Convert the score to a number.
    if 100>=score>=90:  # Check whether the entered value is greater than the score of a level.
        print("Excellent")          # Generate the level when conditions are met.
    elif 90 > score >= 80:
        print("Good")
    elif 80>score>50:
        print("Medium")
    else:
        print("Bad")
except Exception:
    print("Enter a correct score.")
```

9) Loop Statement

9.1. for loop: use the for loop statement to generate a multiplication table.

```
for i in range(1,10):                    # Define the outer loop.
    for j in range(1,i+1):               # Define the inner loop.
        print("%d*%d=%2d"%(i,j,i*j), end=" ")
    print()
```

Output:

1*1= 1
2*1= 2 2*2= 4
3*1= 3 3*2= 6 3*3= 9
4*1= 4 4*2= 8 4*3=12 4*4=16
5*1= 5 5*2=10 5*3=15 5*4=20 5*5=25
6*1= 6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
7*1= 7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
8*1= 8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
9*1= 9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81

9.2. while loop:when the condition is met, the statement block is executed cyclically. To end the loop, use break or continue.

```
i = 0                    # Create variable i.
while i<9:               # Set a condition for the loop.
        i+=1             # The value of i increases by 1 in each loop.
        if i == 3:       # Check whether the conditions are met.
                print("Exit this loop.")
                continue         # Execute continue to exit the current loop.
        if i == 5:
                print("Exit the current big loop.")
                break# Exit the current big loop.
        print(i)
```

Output:

1
2
Exit the current loop.
4
Exit the current big loop.

10) Customizing a Function

```
def hello(greeting='hello',name='world'):          # Default parameters.
   print('%s, %s!' % (greeting, name))             # Format the output.

hello()                                  # hello, world    Default parameter.
hello ('Greetings')                      # The Greetings and world parameters are position parameters.
hello ('Greetings', 'universe')          # The Greetings and universe parameters are position parameters.
hello (name= 'Gumby')                    # The hello and Gumby parameters are keyword parameters.
```

Output:

hello, world!

Greetings, world!

Greetings, universe!

hello, Gumby!

11) Object-oriented Programming

```
class welcomeClass:
   a = 5
   def function1(self):
      print("Welcome to my first class")

W = welcomeClass()
w.function1()
```

12) Standard Library Usage

sys.exit([n]): This method can be used to exit the current program. If the value of n is 0, the program exits normally; if the value of n is not 0, the program exits abnormally.

```
import sys
for i in range(100):
   print(i)
   if i ==5:
      sys.exit(0)
```

Output:

0
1
2
3
4
5