Part 1 Python Introduction and Installation

1. Python Introduction

1.1 Object Oriented Introduction■ √

Python is a object oriented programming language. Object Oriented is a method of software development and a programming paradigm. Difference between Object Oriented and Procedure Oriented is as follow.

Procedure Oriented: Focus on procedure, analyze the steps to solve problems and implement the steps sequentially with functions.

Object Oriented: Focus on object, decompose things that make up the problem into several objects and describe the behavior of an object in the overall solution.

1.2 Python Introduction■ √

Python provides ample API (Application Programming Interface) and tools,which enables the programmer to write extension modules in C, C++ and Python. In addition, Python compiler can also be integrated into the program which requires scripting language, so it is frequently used to integrate and package the program written in other languages.

1.3 Python Feature■ √

2. Python Installation

1) Take installing python3.8 for example. Input command "**sudo apt-get install python3.8**" command, and then input the password and press Enter to install. ■ √

2) If the prompt about whether to continue execution, please input "**Y**" and press Enter. If no error is reported, it means that Python3.8 is installed successfully.■ √

3) Input "**python3.8 -V**" command and press Enter to check whether the version is Python3.8.■ √

3. pyCharm Installation

3.1 Download pyCharm

1) Input command "**sudo apt install snapd snapd-xdg-open**" to install snap installation package. ■ √

2) Input command "**snap refresh**" to refresh snap.■ √

3) Input command "**sudo snap install pycharm-community --classic**" to install pyCharm■ √

3.2 Open pyCharm

1) Open menu and find pyCharm.■ √

2) Create new pyCharm project and configure. Click "**New Project**".■ √

3) Select "**Previously configured interpreter**" and click ....■ √

4) Select "**System interpreter**".■ √

5) Click “**create**”.■ √

6) Lastly, you will enter this interface.■ √

3.3 Usage of pyCharm■ √

Part 2 First Program-“Hello World”

For example, we can print “**Hello World**” string on the screen.

1. Operation Steps

1) Start virtual machine, press “**Ctrl+Alt+T**” to open command line terminal.■ √

2) Input “**sudo apt install vim**” command to install vim editor. During installation, if the prompt about whether to continue execution occurs, just input “**Y**” and press Enter.■ √

3) Input “**mkdir test**” command and press Enter to build a folder named “**test**” under the current directory.■ √

4) Input command “**cd test/**” and press Enter to enter “**test**” folder.■ √

5) Input “**touch hello.py**” command and press Enter to create a program file named “**hello**”.■ √

6) Input “**vim hello.py**” command and press Enter to open program file.■ √

7) Press “**I**” key to enter editing mode and then input “**print("Hello World")**”.■ √

8) Press “**Esc**” and input “**:wq**” and press Enter to save and exit the editing.■ √

9) Input “**python3 hello.py**” command and press Enter to run the program file. Then the string will be printed on the terminal.■ √

2. Expansion Content

Besides the string, print() function can be also used to output the result of mathematical expression. Take adding print() function of mathematical expression to the program file for example.

1) Start virtual machine, press “**Ctrl+Alt+T**” to open command line terminal.■ √

2) Input command “**cd test/**” and press Enter to enter “**cd test/**” folder.■ √

3) Input “**vim hello.py**” command, and press Enter to open program file.■ √

4) Press “**I**” key to enter the editing mode and input “**print(100+100)**”.■ √

Note: there is no need to enclose the mathematical expression with double quotation mark.

5) Press “**Esc**” and input “**:wq**” and press Enter to save and exit the editing.■ √

6) Input “**python3 hello.py**” command and press Enter to run the program file. Then the result of the mathematical expression will be printed on the terminal.■ √

3. Function Explanation■ √

Part 3 Python Basic Syntax

1. Comment■ √

1) Single Line Comment: Comment starts with “#” and its format is as follow.

2) Multi Line Comments: Insert three single quotation marks “'''” or three double quotation marks “"""” at the beginning and the end of the comment to comment multiple lines, and the format is as follow.

2. Indentation Rules■ √

Python uses indentation and **colon symbol** (:) for showing where blocks of code begin and end

Note: block of code at the same level should be indented consistently. We can use “**Tab**” key or input **4 spaces** to indent.

3. Coding Standard■ √

Python adopts PEP8 as coding standard. “**PEP**” represents Python Enhancement Proposal, and “**8**” indicates style guide of Python code.

4. Identifier Naming Standard■ √

Identifier is the name of variable, function, class, module and other objects.
In Python, naming identifier should be consistent with the naming rules.

5. Keyword/ Reserved Word■ √

Reserved word, also called keyword, is a word with special meaning in Python. And they cannot be used as variable names, function names, class names, module names or any other object names.

6. Data Type

There are six types of data in Python3, including **Number**, **String**, **List**, **Tuple**, **Dictionary** and **Set**.

And Number, String and Tuple are immutable, while List, Dictionary and Set are mutable.

6.1 Number■ √

Number includes three numeric types to represent numbers or value.

1) **int**: Integers can be binary, octal, decimal and hexadecimal values.

2) **float**: floats are decimal values generally composed of integers and decimals. Each floating point number occupy 8 bytes, that is 64 bits.

3) **complex**: complex number is a number with real and imaginary components. Both real and imaginary components belong to floating type.

4) **bool**: Only has “**True**” and “**False**” values. “**True**” corresponds to “**1**” and “False” corresponds to “**0**”

6.2 String■ √

A string is a collection of multiple characters. Strings in Python are surrounded by either single quotation marks “'”, or double quotation marks “""” and triple quotation marks “'''” or “""""""”.

6.3 List■ √

List is a sequence structure in Python, which can store any types of data,including integer, decimal, string, list, tuple, etc. Its format is as follow.

Variable name = [element 1, element 2,..., element n]

The number of elements in List is unlimited and there can be different types of elements in one List. For better readability of program, it is recommended to use one type of data in one list.

Each element in the List corresponds to integer index value. The corresponding element values can be obtained though index values so as to change and delete them.

6.4 Tuple■ √

Tuple is another important sequence structure in Python, which is similar
to List. And its format is as follow.

Variable name = (element 1, element 2,..., element n)

Different from List, Tuple is immutable sequence, which means that its elements cannot be changed or deleted.

6.5 Dictionary■ √

A dictionary is an unordered, mutable sequence that is created in the following format.

Variable name = (key1:value1, key2:value2,…, keyn:valuen)

Dictionary is the only type of mapping in Python, which means that elements corresponds to each other.

The elements of dictionary can be List, Tuple, Dictionary and other types of data, but key value must be the immutable type. In addition, there should be only one key value in the same dictionary variable.

6.6 Set■ √

Set is used to store non-repetitive elements and its format is as follow.

variable name = {element 1, element 2,..., element n}

Set can only store immutable data type, including integer, float type, string and tuple, but cannot store mutable data type, including List, dictionary and set.

Part 4 Python Conditional Statement

1. Conditional Statement Introduction

Conditional statement controls the execution of different blocks of code through judging whether the condition holds and based on the result of condition expression.

1.1 Conditional Expression🟩 √

Conditional expression consists of **operator** and **operand.** Take "**a<4**" for example. "**a**" and "**4**" are operand， and "**<**" is operator.

Type	Operator	Mathematical Symbol	Meaning
Arithmetic operator	+	+	add
	-	-	subtraction
	*	×	multiplication
	/	÷	Division, and the result is floating point number
	//		Division, and the result is a rounded down integer
	%	%	Modules/ Surplus
	**	^	Exponentiation
Comparison Operators	==	=	Equal
	!=	≠	Not equal
	>	>	Greater than
	<	<	Less than
	>=	⩾	Greater than or equal to
	<=	⩽	Less than or equal to
Comparison Operators	in	∈	Exist/ belong to
	not in	∉	Not exist/ not belong to

Python is able to combine the judgement condition logically through the reserved words "**not**", "**and**" and "**or**".

1) **not**: it represents the "**not**" relationship for an individual condition. If the Boolean property of the "**condition**" is "**True**", the Boolean property of the "**not condition**" is "**False**". If Boolean property of the "**condition**" is "**False**", the Boolean property of the "**not condition**" is "**True**"

2) **and**: it represents "**and**" relationship between several conditions. Only when the Boolean properties of the conditions connected by "and" are **all "True"**, the Boolean property of the logic expression is "**True**", otherwise "**False**".

3) **or**: it represents "**or**" relationship between several conditions. Only when the Boolean properties of the conditions connected by "or" are **all "False"**, the Boolean property of the logic expression is "**False**", otherwise "**True**".

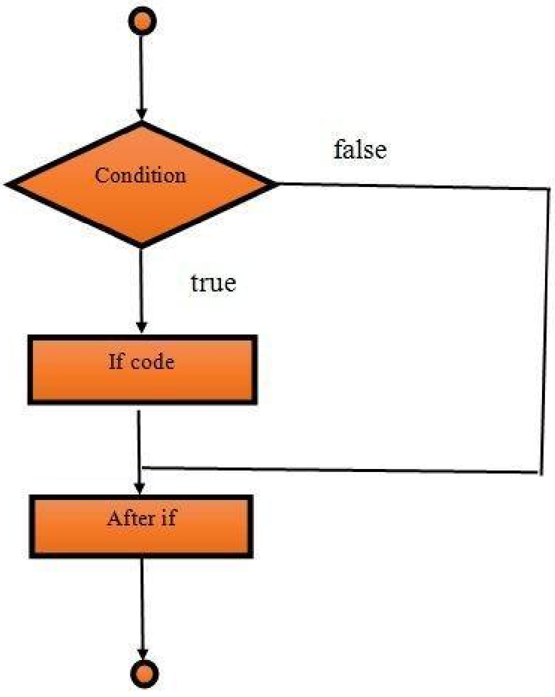
1.2 Selection Statement🟩 √

There are three types of selection statements, including **single-branch selection structure**, **double-branch selection structure** and **multi-branch selection structure**.

1) Single-branch selection structure

The syntax and execution process of single-branch selection structure are as follow.

Syntax	Execution Process

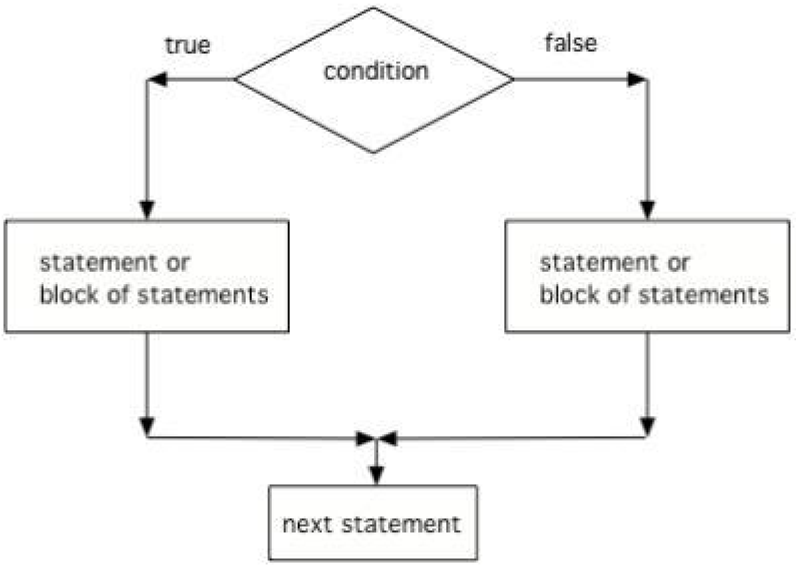
`if condition:`
`    code block`
`code block2`
	

If the condition of “if statement” is true, block of code 1 and 2 will be executed in sequence. Otherwise, skip block of code 1 and directly execute block of code 2.

2) Double-branch selection structure

The syntax and execution process of double-branch selection structure are as follow.

Syntax	Execution Process

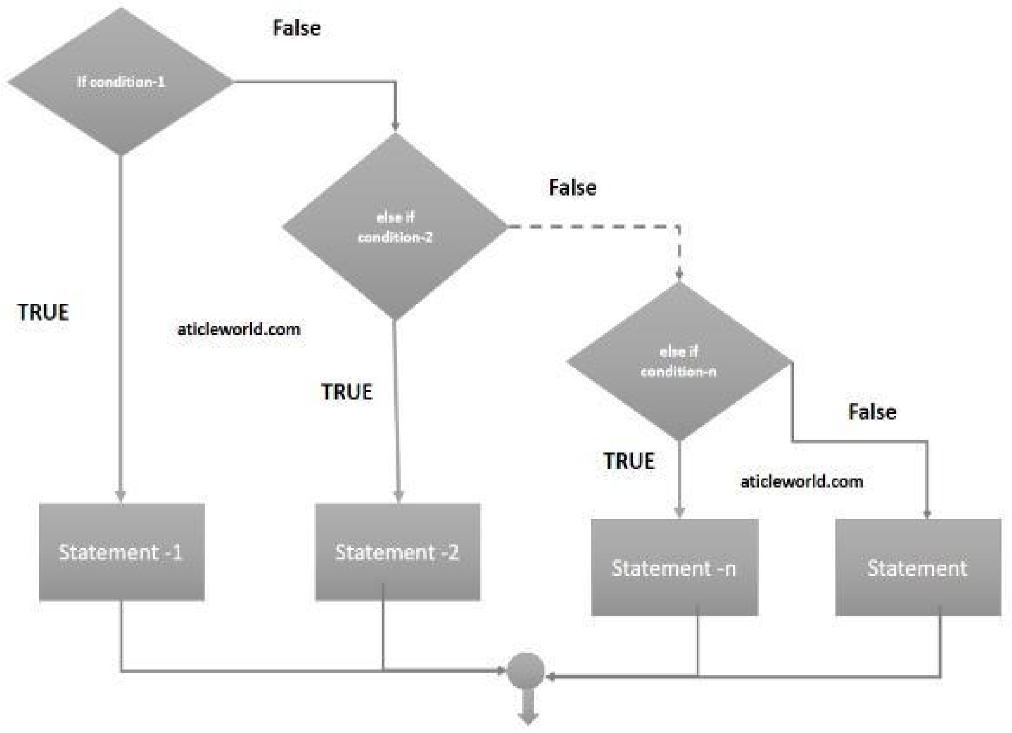
`if condition:`
`    code block1`
`else`
`     code block2`
	 true
	condition
	false
	statement or block of statements
	statement or block of statements
	next statement

If the condition of “if statement” is true, block of code 1 will be executed. Otherwise, block of code 2 will be executed.

3) **Multi-branch selection structure**

The syntax and execution process of multi-branch selection structure are as follow.

Syntax	Execution Process

`if condition:`
`   code block1`
`else condition2:`
`    code block2`
`.....`
`elif condition n:`
`    code block n`
`else:`
`   code block n+1`
	

If the condition of “if statement” is true, block of code 1 will be executed.
If the condition of “if statement” is false, conditions of elif statements will be judged in sequence. When the condition is true, the corresponding block of code will be executed.
If neither conditions of “if statement” nor elif statement are false, block of code n+1 will be executed.

2. Operation Steps■ √

In this routine, the program will calculate BMI value based on the input height and weight, and then perform health assessment.

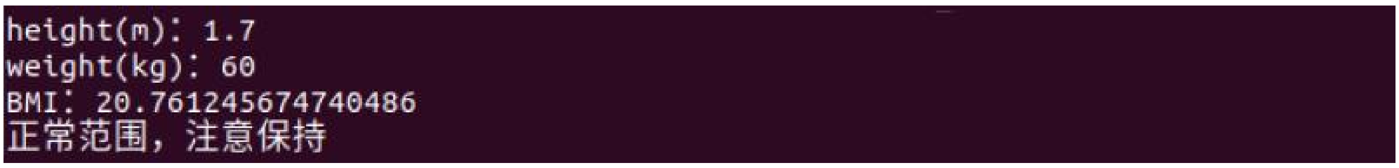
1) Press “**Ctrl+Alt+T**” to open command line terminal.

2) Input “**cd /mnt/hgfs/Share/**” command and press Enter to enter the shared folder.

3) Input “**python3 conditional_statement.py**” command and press Enter to run the routine.

3. Program Outcome■ √

Input height and weight in sequence, and press Enter. Then the terminal will print the corresponding BMI and assessment result.



```
height(m): 1.7
weight(kg): 60
BMI: 20.761245674740486
正常范围，注意保持
```

4. Program Analysis■√

```python
height = float(input("height(m)： "))
weight = float(input("weight(kg)： "))

bmi = weight / (height ** 2)   #计算 BMI 指数
print("BMI： "+str(bmi))

if bmi<18.5:
   print("体重过轻")
elif bmi>=18.5 and bmi<24.9:
   print("正常范围，注意保持")
elif bmi>=24.9 and bmi<29.9:
   print("体重过重")
else:
   print("肥胖")
```

1) Data input

Call input() function to receive the input data. The hints are inside the parenthesis.

```python
height = float(input("height(m)： "))
weight = float(input("weight(kg)： "))
```

2) Data calculation

Process the input data based on BMI calculation formula, and then print the result on the terminal through **print()** function.

```python
bmi = weight / (height ** 2)   #calculate BMI
print("BMI： "+str(bmi))
```

3) Range Judgement

As there are two or more judgement results, multi-branch structure should be adopted.

```python
if bmi<18.5:
   print("体重过轻")
elif bmi>=18.5 and bmi<24.9:
   print("正常范围，注意保持")
elif bmi>=24.9 and bmi<29.9:
   print("体重过重")
else:
   print("肥胖")
```

When BMI is less than 18.5, the corresponding BMI value and “underweight” will be printed.

When BMI is less than 24.9 and greater than or equal to 18.5, the corresponding BMI value and “normal, please keep it” will be printed.

When BMI is less than 29.9 and greater than or equal to 24.9, the corresponding BMI value and “overweight” will be printed.

When BMI is greater than 29.9, the corresponding BMI value and “obese” will be printed.

Part 5 Python Loop Statement

In the program, some steps of the algorithm will be executed repeatedly under certain condition, which is loop statement. In this Part, Python statement will be explained combining with the related routines.

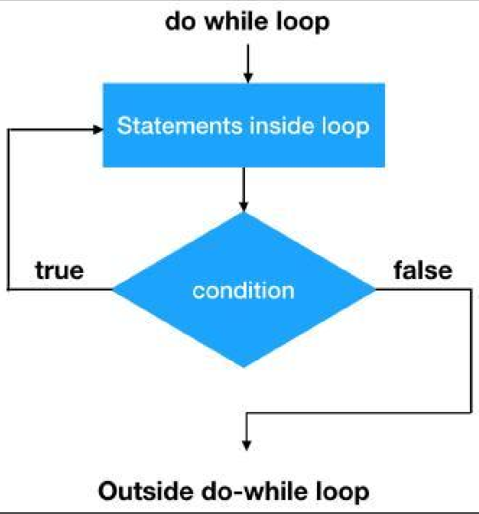
1. Loop Statement Introduction■ √

Loop statement includes "**while**" and "**for**", which is used to repeat some steps.

1.1 while loop

The syntax and execution process of while loop are as follow.

Syntax	Execution Process

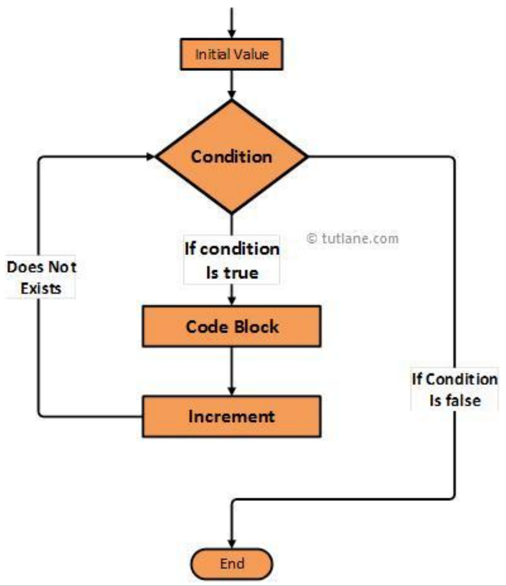
`while loop condition:`
`    code block`
	 do while loop
	Statements inside loop
	true
	condition
	false
	Outside do-while loop

When the loop condition is "**True**", the block of code in while loop will be executed till the loop condition is "**False**". When the loop condition is always "**True**", it will fall into "**dead loop**".

1.2 for loop

The syntax and execution process of for loop are as follow.

Syntax	Execution Process

`for loop variable in target:`
`    code block`
	 Initial Value
	Condition
	If condition Is true
	© tutlane.com
	Does Not Exists
	Code Block
	Increment
	If Condition Is false
	End

“**Target**”can be string, list, tuple, dictionary and other sequence type, while “**loop variable**” is used to store the elements read from the variable of the sequence type.

After entering **for loop**, traverse the elements within target and execute the block of code within **for loop** till the traversal ends.

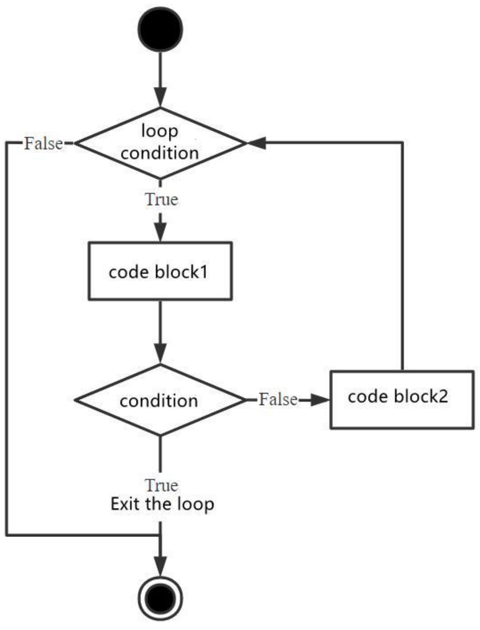
2. Loop Control Statement Introduction

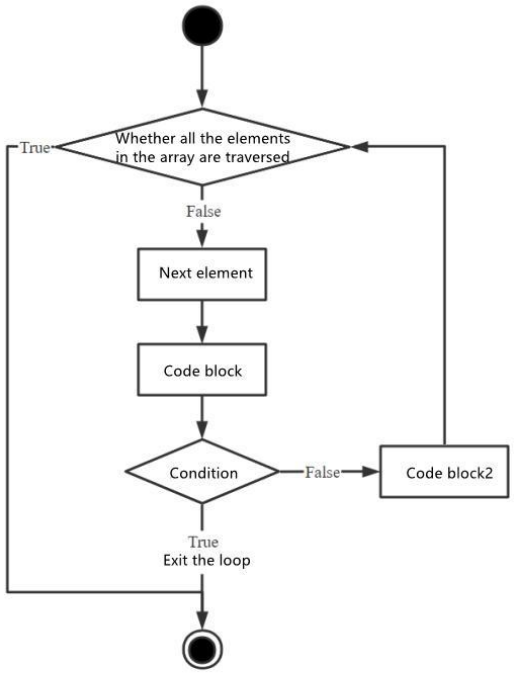
Loop control statement can be used to interrupt the loop, or skip the current loop to execute the next loop. Loop control statement contains “**break**”, “**continue**” and “**pass**”.

2.1 break Statement

break statement is used to get out of the whole loop. The syntax and execution process are as follow.

Syntax	Execution Process

`while loop condition:`
`    code block1`
`    if condition:`
`        break`
`      code block2`
	 loop condition
	False
	True
	code block1
	condition
	False
	code block2
	True
	Exit the loop

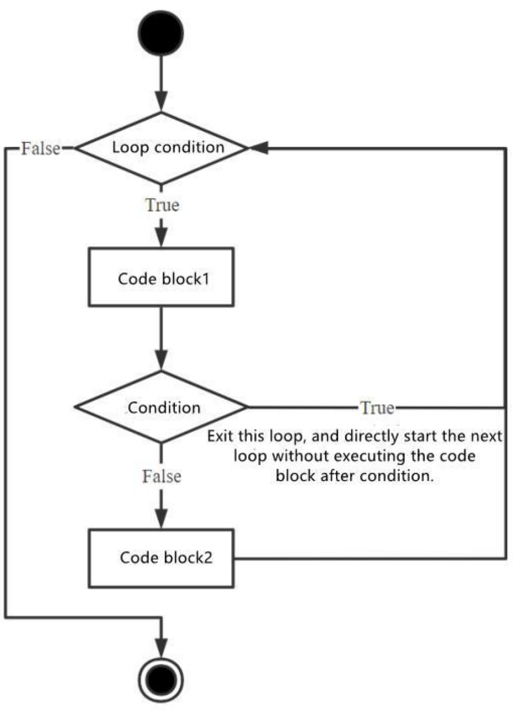
`for loop variable in target:`
`    code block1`
`    if condition:`
`        break`
`      code block2`
	 Whether all the elements in the array are traversed
	True
	False
	Next element
	Code block
	Condition
	False
	Code block2
	True
	Exit the loop

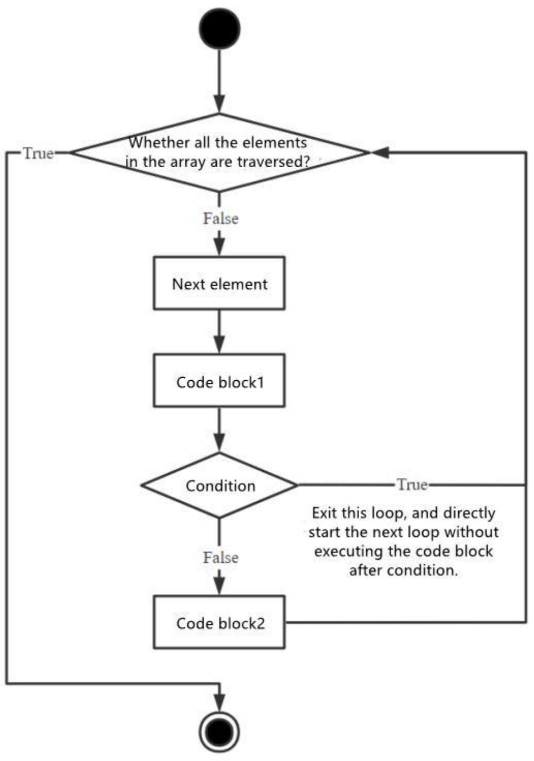
When the judgement condition is executed and is “**True**”, the loop will end. If the judgement condition is “False”, the block of code 2 will be executed and the loop will continue.

2.2 continue statement

continue statement is used to get out of this round of loop and carry out the next round of loop. The syntax and execution process are as follow.

Syntax	Execution Process

`while condition:`
`    code block1(0.001)`
`    if condition:`
`        continue`
`    code block2`
	 Loop condition
	False
	True
	Code block1
	Condition
	True
	Exit this loop, and directly start the next loop without executing the code block after condition.
	False
	Code block2

`for loop variable in target:`
`    code block1`
`    if condition:`
`        continue`
`    code block2`
	 Whether all the elements in the array are traversed?
	True
	False
	Next element
	Code block1
	Condition
	True
	Exit this loop, and directly start the next loop without executing the code block after condition.
	False
	Code block2

When the judgement statement is executed and is "**True**", this round of loop is complete and next round of loop starts. If the judgement condition is "False", the block of code 2 is executed and the second round of loop is done.

2.3 pass statement

pass statement is null statement and nothing will be executed. It is used to keep the integrity of the program structure.

3. Operation Steps🟩 √

This routine will calculate the factorial of the integer and print the elements of the string.

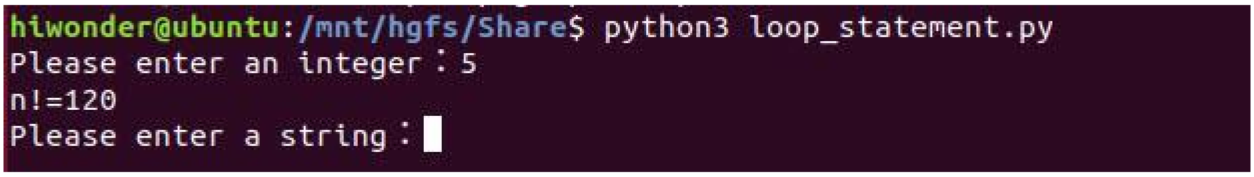
1) Press "**Ctrl+Alt+T**" to open command line terminal.

2) Input "**cd /mnt/hgfs/Share/**" command and press Enter to enter the shared folder.

3) Input "**python3 loop_statement.py**" command and press Enter to run the routine.

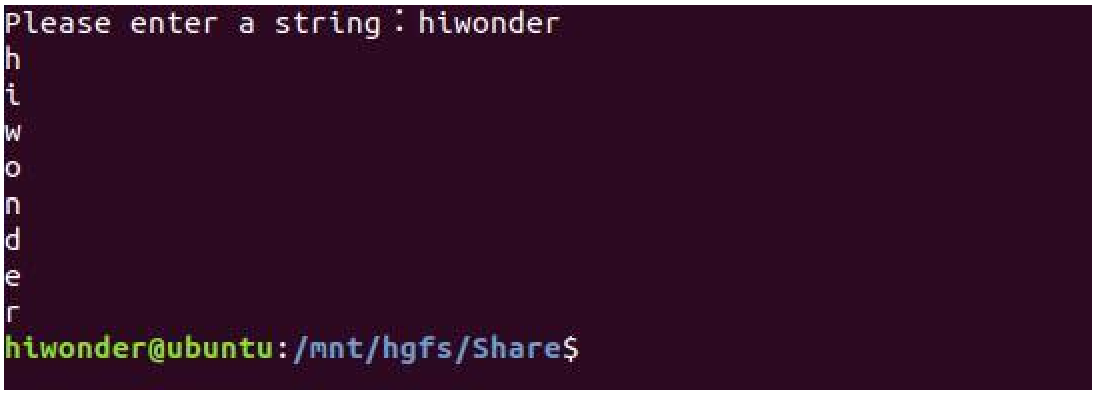
4. Program Outcome🟩 √

Input one integer and press Enter, and then the terminal will print its factorial.



```
hiwonder@ubuntu:/mnt/hgfs/Share$ python3 loop_statement.py
Please enter an integer：5
n!=120
Please enter a string：
```

Input a string and press Enter, and then the terminal will print the elements of this string.



```
Please enter a string：hiwonder
h
i
w
o
n
d
e
r
hiwonder@ubuntu:/mnt/hgfs/Share$
```

5. Program Analysis🟩 √

```
n = int(input("请输入一个整数: "))
fact = 1
i = 1
while i<= n:
    fact = fact*i
    i = i + 1
print("n!={}".format(fact))

string = input("请输入一个字符串: ")
for c in string:
    print(c)
```

5.1 Calculate Factorial of the Value🟩 √

1) Input data

Call input() function to receive the input data. The hints are inside the parenthesis.

```
n = int(input("请输入一个整数: "))
```

2) Create variable Input data

Create two variables for later calculation. “**fact**” is used to store the current factorial.

```
fact = 1
i = 1
```

3) while loop

Use while loop statement to calculate the factorial of the designated integer. During loop, “**i**” value will keep adding up. When this value is greater than the input integer, the loop will end. Then, call print() function to print the result on the terminal.

```
while i<= n:
    fact = fact*i
    i = i + 1
print("n!={}".format(fact))
```

5.2 Print String Elements🟩 √

1) Input data

Call input() function to receive the input data. The hints are inside the parenthesis.

```
string = input("请输入一个字符串：")
```

2) For loop

Use for loop to obtain the elements of the input string. Then call print() function in the loop structure to print the input string separately.

```
for c in string:
    print(c)
```