

Jupyter notebook common shortcuts



- Esc: exit edit mode, enable command mode

★ Enter	Enter edit mode	↑	Select cell above	↑ Shift + V	Paste cell above
★ ↑ Shift + Enter	Run cell, <u>select below</u>	or K		V	Paste cell below
Ctrl + Enter	Run cell	↓	Select cell below	★ Z	Undo last cell deletion
Alt + Enter	Run cell, <u>insert below</u>	or J		★ D then D	Delete selected cell
Y	Go to code	★ A	Insert cell <u>above</u>	↑ Shift + M	Merge cell below
M	Go to markdown	★ B	Insert cell <u>below</u>	Ctrl + S	Save and Checkpoint
R	Go to raw	X	Cut selected cell		
		C	Copy selected cell		

- <https://shortcutworld.com/Jupyter-Notebook/win/Jupyter->

Basic data types in Python



- 6 basic data types in Python 3:
 - ▶ Number
 - ▶ String
 - ▶ List
 - ▶ Tuple
 - ▶ Set
 - ▶ Dictionary
- **Fixed** data types: number, string, tuple;
- **Flexible** data types: list, dictionary, set.
- **Numeric** types in Python 3: int, float, bool, complex
(复数)



Variables

- Python does not require variables to be defined before use.
- It also does not require specifying the data type of a variable.
- One variable can be assigned with different type of values repeatedly.
 - ▶ This is part of the reason why Python is called a dynamic language.
- Python also allows to assign different types of values at the same time to multiple variables.



Operators & expressions

- Python supports standard operators

操作符	描述
+, -, *, /, %, <u>//</u> , <u>**</u>	算术运算：加、减、乘、除、取模、 <u>整除</u> 、 <u>幂</u>
<, <=, >, >=, !=, ==	关系运算符
and, or, not,	逻辑运算符



String

- In Python, string values are quoted with single quote ('), double quote (") or triple quote ("), these quotes must match.

- Escape symbol: \

back slash

转义序列	说明
\n	换行
\\	反斜杠
\"	双引号
\t	制表符

- We can use slice ([] or [:]) to select substrings.
- Python index starts with 0. Backward indexing starts with -1 as the last character and minus 1.

Common methods of Strings



Method	Operations
str.capitalize()	返回字符串的 <u>副本</u> ，其 <u>首字符大写</u> ，其余字符小写
str.count(sub [,start [,end]])	返回[start, end]范围内sub的非重叠出现次数，start和end可选
str.endswith(sub[,start[,end]])	返回布尔值，表示字符串是否以指定的sub结束，同类方法str.startswith()
str.find(sub [,start [,end]])	返回字符串中首次出现子串sub的索引位置，start和end可选，若未找到sub，返回-1
str.split(sep =None)	使用sep作为分隔符拆分字符串，返回字符串中单词的列表，分隔空字符串
str.strip([chars])	删除字符串前端和尾部chars指定的字符集，如果省略或None，则删除空白字符



Output in Python: String formatting

1. Use %

```
>>> name = "Eric"
>>> age = 74
>>> "Hello, %s. You are %s." % (name, age)
'Hello Eric. You are 74.'
```

- ▶ Verbose and error prone for longer strings and many variables

2. Use str.format() 容易出错

- ▶ Simple syntax

```
>>> "Hello, {1}. You are {0}.".format(age, name)
'Hello, Eric. You are 74.'
```

- ▶ Access values from dictionaries

```
person = {'name': 'Eric', 'age': 74}
print("Hello, {name}. You are {age}.".format(name=person['name'], age=person['age']))
Hello, Eric. You are 74.
```

```
>>> person = {'name': 'Eric', 'age': 74}
>>> "Hello, {name}. You are {age}.".format(**person)
'Hello, Eric. You are 74.'
```



Output in Python: String formatting

1. **Use f-strings:** Also called “formatted string literals,” f-strings are string literals that have an f at the beginning and curly braces containing expressions that will be replaced with their values.

```
name = "Eric"
age = 74
f"Hello, {name}. You are {age}."
'Hello, Eric. You are 74.'
```

2. Because f-strings are evaluated at runtime, you can put any and all valid Python expressions in them.

```
>>> f"{2 * 37}"
'74'
```

```
>>> f"{name.lower()} is funny."
'eric idle is funny.'
```

```
def to_lowercase(input):
    return input.lower()

name = "Eric Idle"
f"{to_lowercase(name)} is funny."
```

range() 左闭右开.



Built-in data structures

- When dealing with data records, the commonly used data structures are the four built-in data structures in Python
 - ▶ **List** *ordered, sequence of items, mutable.*
 - ▶ **Tuple**
 - ▶ **Dictionary**
 - ▶ **Set**

```
>>> print([1, 2, 3])           # <class 'list'>
[1, 2, 3]
>>> print((1, 2, 3))          # <class 'tuple'>
(1, 2, 3)
>>> print({'red', 'green', 'blue'}) # <class 'set'>
{'red', 'green', 'blue'}
>>> print({'name': 'Alice', 'age': 42}) # <class 'dict'>
{'name': 'Alice', 'age': 42}
```

List

- Features of list:
 - ▶ Flexible
 - ▶ Ordered
 - ▶ Members can have different types
 - ▶ Varied length
 - ▶ Allows nested lists
 - ▶ Allows in-place modification

- Definition of a list:

```
list = [2, 4, 6, 8, 10]
```

```
list = ['a', 'b', 'c', 'd', 'e']
```

- **print the lists; print the length of the lists.**

List



➤ Common methods:

- 1) L.append(v) : 把v添加到列表L的结尾
- 2) L.insert(i, v): 将值v插入到列表L的索引i处
- 3) L.index(v): 返回列表中第一个值为v的元素的索引
- 4) L.remove(v): 从列表L中移除第一次找到的值v
- 5) L.pop(i): 从列表的指定位置删除元素，并将其返回。如果没有指定索引，a.pop()返回最后一个元素。
- 6) L.reverse(): 倒排列表中的元素
- 7) L.count(v): 返回v在列表中出现的次数
- 8) L.sort(key=None, reverse=False): 对链表中的元素进行适当的排序。

List



➤ Iterate over the elements of a list

syntax	[<expr1> for k in L if <expr2>]
meaning	<pre>returnList=[] for k in L: if <expr2>: returnList.append(<expr1>) return returnList;</pre>

➤ Examples:

- ▶ For each element of a list, print its product with 3
- ▶ For each element of a list, if it's greater than 6, print its product with 3
- ▶ Given two lists, if the product of each element at the same position is greater than 0, print the product

Tuple

➤ Features:

- ▶ Immutable: fixed once defined
- ▶ Does not support item assignment
- ▶ Use parenthesis () instead of square brackets

Dictionary



➤ Consists of key-value pairs.

➤ Features:

- ▶ Uses brackets in definition.
- ▶ Key has to be unique.
- ▶ Only use immutable data type (e.g. string) as key.
- ▶ Unordered.
- ▶ Mutable: allow item assignment.

➤ Definition of a dictionary:

dict = {key1:value1, key2:value2}

➤ Common methods:

方法	描述
<u>get(key, default=None)</u>	返回指定键的值，若值不在字典中则返回default值，default的系统缺省值是None
<u>items()</u>	以列表返回可遍历的(键, 值) 元组数组
<u>keys()</u>	以列表返回一个字典所有的键
<u>values()</u>	以列表返回字典中的所有值
<u>update(dict)</u>	更新， <u>加入字典dict中的元素</u>
<u>clear()</u>	清空字典

Set

➤ Features:

- ▶ Unordered
- ▶ Unique elements

➤ Definition of a set:

$a = \{1, 1, 2, 3, 4\}$

$a = \text{set}([1, 1, 2, 3, 4])$ 去重.

➤ Note:

- ▶ `set()` method takes a list and turns it into a set.
- ▶ Must use the `set()` method to create an empty set.
- ▶ `{}` creates an empty dictionary.

Set



➤ Common methods:

<code>set_a.issubset(set_b)</code>	Whether the input set <code>set_b</code> is a subset of the current set <code>set_a</code>
<code>set_a.union(set_b)</code>	Computes the union of two sets
<code>set_a.difference(set_b)</code>	Computes <code>set_b - set_a</code>
<code>set_a.intersection(set_b)</code>	Computes the intersection of <code>set_a</code> and <code>set_b</code>

In:	<pre>set1 = set([0,1,2,3,4]) set2 = set([1,3,5,7,9]) print(set1.issubset(set2)) print(set1.union(set2)) print(set2.difference(set1)) print(set2.intersection(set1))</pre>
Out:	<pre>False {0, 1, 2, 3, 4, 5, 7, 9} {9, 5, 7}</pre>

lambda function



- Python使用lambda来创建匿名函数(anonymous function)
- 准确地说，lambda只是一个表达式，函数体比def定义的函数简单的多
- 在lambda表达式中只能封装有限的逻辑。
- lambda函数不能访问自有参数列表之外或全局命名空间里的参数。
- Syntax:

```
func_name=lambda [v1, v2,...]: expression  
func_name([v1, v2,...])
```

lambda function



- Example:

Write a lambda function to calculate $1+2x+y^2+zy$

```
polynomial = lambda x,y,z: 1+2*x+y**2+z*y  
polynomial(1,2,3)
```

Or the function can be anonymous:

```
lambda x,y,z: 1+2*x+y**2+z*y  
_(1,2,3)
```

The underscore references to the last evaluated expression.

Note: this only works in the interactive interpreter, and you could not write similar code in a Python module.

Create · insert... select · grant ·

File manipulation

DDL DML DQL DCL



➤ Common process:

1. Open file: `open()`;
2. Read/write file: `read()`, `readline()`, `readlines()`, `write()`;
3. Process data from file;
4. Close file: `close()`

➤ Documentation:

<https://docs.python.org/3/tutorial/inputoutput.html#reading-and->

Open a file



- First step is to open a file using the `open()` function. This will return a file object:

```
file_object = open(file_name [, access_mode = "r",  
                    buffering= -1])
```

➤ Note:

- ▶ Files opened with the `open()` function must be closed with the `close()` function when finish reading, so as to release the file.
- Alternatively, we can use `with open()` as `f`, which will automatically close the file when finish reading.

```
{ with open("student.txt", "r") as f:  
  processes...
```

- We can use `f.closed` afterwards to check whether file is closed automatically.

CSV files



- CSV (Comma Separated Values) is a very common file type of datasets in data mining.
- CSV is a plain text format:
 - ▶ Pure text, using certain character set, e.g. ASCII, Unicode or GB2312
 - ▶ Each line is a record.
 - ▶ Each record is separated into fields, the separator is usually comma or tab.
 - ▶ Field order is the same for each record.
- Python has a built-in csv module, which can be used to read and process csv files.
 - ▶ Load the module: `import csv`
 - ▶ Documentation: <https://docs.python.org/3/library/csv.html>