# Python Cheat Sheet

## Comments

| | |
|---|---|
| `#` | Single Line Comment |
| `"""`<br>`Text`<br>`More Text`<br>`"""` | Multi-line comment |

## Input/Output

| | |
|---|---|
| `format()` | Converts a value to a formatted representation |
| `input()` | Reads input from the console `text=input(“Enter: ")` |
| `open()` | Opens a file and returns a file object |
| `print()` | Prints to a text stream or the console |

## Generic Operations

| | |
|---|---|
| `len(a)` | Gives item count in a |
| `max()` | Returns the largest of the given arguments |
| `min()` | Returns the smallest of the given arguments |
| `sum()` | Sum of all arguments |
| `sorted(a)` | Sorted list copy of a |
| `str(x)` | Convert to string |

## List Operations

| | |
|---|---|
| `users = ['scott', 'bob', 'jim']` | A list stores a series of items in a particular order. Lists allow you to store sets of information in one place, whether you have just a few items or millions of items. |
| `list=[]` | Defines an empty list. |
| `list[0]='scott'` | Stores scott in the list. |
| `list.append('bob')` | Add a user to the end of the list. |
| `list.insert(2, 'jim')` | Insert a user at a particular position in the list. |
| `list`<br>`['scott', 'bob', 'jim']` | Shows the list. |
| `list[0]`<br>`scott` | Retrieves the character at the 0 position. |
| `list[0:2]`<br>`['scott', 'bob']` | Retrieves indexes starting with 0 and ending before 2. |
| `list.remove('bob')` | Removes bob from the list. |

## Dictionary Operations

| | |
|---|---|
| `user = {`<br>`    'first': 'scott',`<br>`    'last': 'stephenson',`<br>`    'location': 'texas',`<br>`}` | Python's dictionaries allow you to connect pieces of related information. Each piece of information in a<br>dictionary is stored as a key-value pair. When you provide a key, Python returns the value associated<br>with that key. You can loop through all the key-value pairs, all the keys, or all the values. |
| `dict={}` | Defines an empty dictionary |
| `dict[i]=a` | Stores “a” to the key “i” |
| `dict[i]` | Retrieves the item with the key “i” |
| `dict.key` | Gives all the key items |
| `dict.values` | Gives all the values |

## Datatypes

| | |
|---|---|
| `a=2` | Integer |
| `b=2.0` | Float |
| `c=1+2j` | Complex |
| `d=[1,2,3,’Word’]` | List |
| `e=(1,2,4)` | Tuple |
| `f=“New String”` | String |
| `g={3,4,5}` | Sets |
| `h= {'a': [1,2],'b': [3,4]}` | Dictionary |

# Python Cheat Sheet

## Numeric Operators

| | |
|---|---|
| `x + y` | Add |
| `x - y` | Subtract |
| `x * y` | Multiply |
| `x / y` | Divide |
| `x // y` | Floored quotient |
| `x % y` | Remainder |
| `-x` | x negated |
| `+x` | x unchanged |
| `abs(x)` | Absolute value |
| `int(x)` | Convert to integer |
| `float(x)` | Convert to float |
| `x ** y` | x to the power of y |

## Comparison Operators

| | |
|---|---|
| `<` | Strictly less than |
| `<=` | Less than or equal |
| `>` | Strictly greater than |
| `>=` | Greater than or equal |
| `==` | Equal |
| `!=` | Not equal |
| `is` | Object identity |
| `is not` | Negated object identity |

## Boolean Operators

| | |
|---|---|
| `x and y` | If both are True, then True, else False. |
| `x or y` | If either is True, then True, else False. |
| `not x` | If True, then False. If False, then True. |

## Class/Function/Method/Object

```
class Pen:
    """The functions/methods to define a pen"""
```

Classes represent things you want to model in your program. You use a class to make objects, which are specific instances of those things. A class defines the general behavior that a whole category of objects can have, and the information that can be associated with those objects.

```
class Pen:
    def color(self):
        self.color = 'Blue'
```

Within the Pen class, what information would you associate with this pen and what behavior would it have? The information is stored in variables called attributes, and the behavior is represented by functions. Functions that are part of a class are called methods.

```
obj=Pen()
```

Object

```
def new_function():
    name=input('What is your first name? ')
    print("Hello, " + name)
new_function()
Hello, Scott
```

Function can be independent from a class. They can be used to group commands together. Functions are named blocks of code designed to do one specific job. Functions allow you to write code once that can then be run whenever you need to accomplish the same task.

## Flow Control Method

```
if-else
if price>=700:
print(“Buy.”)
else:
print(“Don’t buy.”)
```

Conditional Statement

```
For loop
a=“New Text”
count=0
for i in a:
if i==‘e’:
count=count+1 print(count)
```

Iterative Loop Statement

```
While loop
a=0 i=1
while i <10:
a=a*2 i=i+1
print(a)
```

Conditional Loop Statement

```
Break, Pass and continue
```

Loop Controls

# Python Cheat Sheet

## Try, Except, Finally Block

| | |
|---|---|
| `try:`<br>`  print(x)`<br>`except:`<br>`  print("Something went wrong")`<br>`finally:`<br>`  print("The 'try except' is finished")` | Statement body block to raise Exception. |

## Netmiko

| | |
|---|---|
| `from netmiko import ConnectHandler`<br>`connection=ConnectHandler(device_type='cisco_ios', ip='x.x.x.x', username='admin', password='cisco')` | Import and use just the ConnectHandler from netmiko |
| `import netmiko`<br>`connection=netmiko.ConnectHandler(device_type= 'cisco_ios', ip='x.x.x.x', username='admin', password='cisco')` | Import and use all of netmiko |

## File Operations

| | |
|---|---|
| `file_object=open('File Name','opening mode')`<br>`file_object.write('text')` | Write information to a file (Opening modes: r: read, w: write, a: append, r+: both read and write)<br>write (single line of text) or writelines (multiple lines of text) |
| `file_object=open('File Name','opening mode')`<br>`print(file_object.read())` | Read the contents of a file |