PYTHON BASICS
Cheat Sheet

Data Structres

Numbers

```
# Integer (int)
>>> x = 2
# Float
>>> x = 2.3
# Complex
>>> x = 2j + 1
```

Operations

```
# Exponent
>>> 2**3
8

# Modulus / Remainder
>>> 22%8
6

# Integer Division
>>> 22//8
2

# Division
>>> 22/8
2.75

# Multiplication
>>> 3*3
9

# Subtraction
>>> 5-2
3

# Addition
>>> 2+2
4
```

Boolean

```
>>> a = 2
>>> b = 7

>>> a == b
False

>>> a != b
True

>>> a > b
False

>>> a >= b
False

>>> a < b
True

>>> a <= b
True

>>> (a > 1) & (b < 10)
True

>>> (a > 3) | (b > 5)
True

>>> a is b
False

>>> a is not b
True
```

Strings

```
>>> txt_1 = 'Hello World!'
'Hello World!'

>>> txt_2 = "HELLO WORLD!"
'HELLO WORLD!'

>>> long_txt = """
               Hello World,
               Welcome DSMLBC!
               """
'Hello World, \n Welcome DSMLBC!'
```

Indexing & Slicing

```
>>> txt_1[0]
'H'

>>> txt_1[-1]
'!'

>>> txt_1[1:4]
'ell'

>>> txt_1[:5]
'Hello'
```

String Methods

```
>>> len(txt_1)
12

>>> txt_1.upper()
'HELLO WORLD!'

>>> txt_2.lower()
'hello world!'

>>> txt_1.replace("World", "Era")
'Hello Era!'

>>> txt_1.split()
['Hello', 'World!']

>>> ' Hello World! '.strip()
'Hello World!'

>>> 'hello world!'.capitalize()
'Hello world!'
```

Dictionaries

```
>>> dict_1 = {"REG" : "Regression",
              "LOG" : "Logistic Regression",
              "CART": "Classification and Reg"}

>>> dict_2 = {"REG": ["RMSE", 10]
              "LOG": ["MSE", 20]
              "CART": ["SSE", 30]}
```

Key - Value Methods

```
>>> dict_1.keys()
dict_keys(['REG', 'LOG', 'CART'])

>>> dict_1.values()
dict_values(['Regression', 'Logistic Regression',
             'Classification and Reg'])

>>> dict_1.items()
dict_items([('REG', 'Regression'),
            ('LOG', 'Logistic Regression'),
            ('CART', 'Classification and Reg')])

>>> dict_1["REG"]
'Regression'

>>> dict_2["CART"][1]
30
```

Dictionary Methods

```
>>> dict_1 = {"REG": "Regression",
              "LOG": "Logistic Regression",
              "CART": "Classification and Reg"}

>>> dict_2 = {"REG": ["RMSE", 10]
              "LOG": ["MSE", 20]
              "CART":["SSE", 30]}
```

Lists

```
>>> list_1 = [1, 2, 3, "a", "b"]

>>> list_2 = [True, [1, 2, 3]]

>>> numbers = [4, 2, 1, 3]
```

Indexing & Slicing

```
>>> list_1[3]
'a'

>>>list_2[-1]
[1, 2, 3]

>>>list_1[0:4]
[2, 3, 'a']
```

List Methods

```
>>> list_1 + list_2
[1, 2, 3, 'a', 'b', True, [1, 2, 3]]

>>> list_1.append('c')
[1, 2, 3, 'a', 'b', 'c']

>>> list_2.remove('True')
[[1, 2, 3]]

>>> len(list_1)
5

>>> numbers.sort()
[1, 2, 3, 4]

>>> list_2.insert(1, False)
[True, False, [1, 2, 3]]

>>> numbers.pop()
[4, 2, 1]
```

Tuples

```
>>> tuple = ("john", "mark", 1, 2)

>>> tuple[0]
'john'

>>> tuple[1:3]
       ('mark', 1)

>> len(tuple)
4
```

Set

```
>>> set_1 = {1, 2, 2, 3, 3, 3}
{1, 2, 3}

>>> set_2 = set([3, 4, 4, 5, 5, 5, 6])
{3, 4, 5, 6}
```

Set Methods

```
>>> set_1.difference(set_2)
{1, 2}

>>> set_2.difference(set_1)
{4, 5, 6}

>>> set_1.symmetric_difference(set_2)
{1, 2, 4, 5, 6}

>>> set_1.intersection(set_2)
{3}

>>> set_1.union(set_2)
{1, 2, 3, 4, 5, 6}

>>> set_1.isdisjoint(set_2)
False

>>> set_1.issubset(set_2)
False

>>> set_1.issuperset(set_2)
False
```

Comprehensions

List Comprehensions

```
# Syntax
list = [expression for item in iterable if condition]
>>> squares = [x**2 for x in range(1, 11)]
>>> print(squares)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

>>> evens = [f"{x}: even" for x in range(1,10) if x % 2 == 0]
>>> print(evens)
['2: even', '4: even', '6: even', '8: even']

>>> even_odd = [f"{x}: even" if x%2 == 0 else f"{x}: odd"
                for x in range(1,5)]

>>> print(even_odd)
['1: odd', '2: even', '3: odd', '4: even']
```

Dictionary Comprehensions

```
# Syntax
dict = {key_exp: value_exp for item in iterable if condition}

>>> dictionary = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

>>> {k: v ** 2 for (k, v) in dictionary.items()}
{'a': 1, 'b': 4, 'c': 9, 'd': 16}

>>> {k.upper(): v for (k, v) in dictionary.items()}
{'A': 1, 'B': 2, 'C': 3, 'D': 4}

>>> {k.upper(): v*2 for (k, v) in dictionary.items()}
{'A': 2, 'B': 4, 'C': 6, 'D': 8}
```

Loops

For Loop

```
# Syntax
for <variable> in <list>:
   <code block>

>>> students = ["John", "Mark", "Venessa"]

>>> for student in students:
...     print(student)
John
Mark
Venessa

>>> for index, student in enumerate(students):
...     print(index, student)

0 John
1 Mark
2 Venessa
```

While Loop

```
# Syntax
while <condition>:
  <code to execute while the condition is true>

>>> i = 1
>>> while i < 5:
...     print(i)
...     if i==3:
...         break
...     i+=1
1
2
3

>>> while i < 5:
...     i+=1
...     if i==3:
...         continue
...     print(i)
2
4
5
```

Conditions

```
>>> number = 3.14
>>> if number > 0:
...     print(f"{number} is positive.")
... elif number < 0:
...     print(f"{number} is negative.")
... else:
...     print(f"{number} is zero!")
3.14 is positive.
```

Functions

```
# Function_1
>>> def say_hi(name):
...     print(f'Merhaba {name}')

# Calling function
>>> say_hi('Miuul')
Merhaba Miuul

# Function_2
>>> def summer(num_1, num_2):
...     """
...     Sum of two numbers
...     Args:
...         num_1: int, float
...         num_22: int, float
...     Returns:
...         int, float
...     """
...     Return num_1 + num_2

# Calling function
>>> summer(3, 4)
7

# Function_3
>>> def find_volume(length=1, width=1, depth=1):
...     print(f'Length = {length}')
...     print(f'Width = {width}')
...     print(f'Depth = {depth}')
...     volume = length * width * depth

# Calling function
>>> find_volume(1, 2, 3)
Length = 1
Width = 2
Depth = 3
6

>>> find_volume(2, depth=3, width=4)
Length = 2
Width = 4
Depth = 3
24
...     return volume
```

Local & Global Variables

```
>>> list_store = [1, 2] # Global variable

>>> def add_element(a, b):
... c=a*b
... list_store.append(c)
... print(list_store)

>>> add_element(1, 9)
[1, 2, 9]
```

Built-in Functions

```
# lambda
>>> summer = lambda a, b: a + b
>>> summer(3, 5)
8

# map
>>> salaries = [1000, 2000, 3000, 4000, 5000]
>>> list(map(lambda x: x * 20 / 100 + x, salaries))
[1200.0, 2400.0, 3600.0, 4800.0, 6000.0]

# filter
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(filter(lambda x: x % 2 == 0, numbers))
[2, 4, 6, 8, 10]

# reduce
>>> numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> reduce(lambda a, b: a + b, numbers)
55

# zip
>>> students = ["John", "Mark", "Venessa"]
>>> departments = ["mathematics", "statistics", "physics"]
>>> list(zip(students, departments))
[('John', 'mathematics'), ('Mark', 'statistics'), ('Venessa', 'physics')]
```