Python

<https://bic-berkeley.github.io/psych-214-fall-2016/string_literals.html>

<http://candcplusplus.com/python-numeric-literals-integer-floating-point-imaginary-literals>

what is PVM?

It is python virtual machine which is responsible for converting the byte code into machine code. So, the machine can execute those machine codes and give the output.

Difference between list and tuple?

The Key Difference between a List and a Tuple. The main difference between lists and tuples is the fact that **lists are mutable whereas tuples are immutable**. ... A mutable data type means that a python object of this type can be modified. An immutable object can't.

<https://www.programiz.com/python-programming/list-vs-tuples>

**>>> t = [('Swordfish', 'Dominic Sena', 2001), ('Snowden', ' Oliver Stone', 2016), ('Taxi Driver', 'Martin Scorsese', 1976)]**

**>>> t**

[('Swordfish', 'Dominic Sena', 2001), ('Snowden', ' Oliver Stone', 2016), ('Taxi Driver', 'Martin Scorsese', 1976)]

**>>> key\_val= {('alpha','bravo'):123}**

**>>> key\_val = {['alpha','bravo']:123}**

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'list'

>>>

Set: <https://www.programiz.com/python-programming/set>

>>> set(tuple)

{1, 2}

>>> a = set(1,2,3,1,4)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: set expected at most 1 arguments, got 5

>>> set(tuple)

{1, 2}

>>> my\_set = {1, 2, 3}

>>> print(my\_set)

{1, 2, 3}

>>> my\_set = {1.0, "Hello", (1, 2, 3)}

>>> my\_set

{1.0, 'Hello', (1, 2, 3)}

>>> a = {}

>>> type(a)

<class 'dict'>

>>> a = set()

>>> type(a)

<class 'set'>

>>> myset = {1,3}

>>> myset

{1, 3}

>>> myset.add(2)

>>> myset

{1, 2, 3}

>>> myset.add(4)

>>> myset

{1, 2, 3, 4}

>>> myset.update([5,6,7])

>>> myset

{1, 2, 3, 4, 5, 6, 7}

>>> myset.update({8,9},[10])

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

>>> myset.add("Hello")

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 'Hello'}

>>> myset[1]

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'set' object is not subscriptable

**Delete element in set**

>>> myset.discard(10)

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8, 9, 'Hello'}

>>> myset.discard("Hello")

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8, 9}

**Diffrence between discard and remove**

>>> myset.discard(0)

>>> myset.remove(0)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

KeyError: 0

>>> myset.remove(9)

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8}

**Note: there is no indexing in set**

**Pop in set: It pops random number**

>>> myset

{1, 2, 3, 4, 5, 6, 7, 8}

>>> myset.pop()

1

>>> myset

{2, 3, 4, 5, 6, 7, 8}

>>> myset.pop()

2

>>> myset

{3, 4, 5, 6, 7, 8}

>>> myset.pop()

3

>>>

**Clear all elements in set**

>>> myset.clear()

>>> myset

set()

**Can you count repeated element in set?**

No, you cannot read repeated element in set

**>>> a = {1,2,3,4,4}**

>>> **list(a)**

[1, 2, 3, 4]

Even if you convert the set to list you cannot count the duplicate element.

Can you reverse set ?

<https://stackoverflow.com/questions/67363312/reverse-the-elements-of-a-set>

As set is random it does not have indexing, it is not essential or needed but you can reverse the set.

>>> s = {1, 2, 3}

>>> for i in range(len(s)):

elem = max(s)

s.remove(elem)

print(elem, end=',' if s else '')

...

3,2,1>>>

It's just a variable name, and it's conventional in python to use \_ for throwaway variables. It just indicates that the loop variable isn't actually used.

>>> person.get('name')

'Vinee'

>>> person['name']

'Vinee'

>>> person.get('name', ['hello','hiiii'])

'Vinee'

>>> person.get('e', ['hello','hiiii'])

['hello', 'hiiii']

>>> person.get('e')

>>> person.keys()

dict\_keys(['name', 'age'])

>>> person.values()

dict\_values(['Vinee', 29])

>>> person.items()

dict\_items([('name', 'Vinee'), ('age', 29)])

add to a dictionary

>>> person

{'name': 'Vinee', 'age': 29}

>>> person['age'] = 30

>>> person

{'name': 'Vinee', 'age': 30}

You can use pop() to remove a particular key

>>> person.pop("name")

'Vinee'

>>> person

{'age': 30}

Can key be null?

it can be mentioned as None in python

can we use same key multiple times?

>>> person["name"] = 'yash'

>>> person

{'name': 'yash', 'age': 30, '': 20, None: 0}

>>> person["name"] = 'Vins'

>>> person

{'name': 'Vins', 'age': 30, '': 20, None: 0}