Value -> AnyValid data types in python, ont, float, complex, str, list, dict, set, tuple

Var-> Container which holds data in memory, ar is kust an identier which points an objecg in memory

```
Start coding or generate with AI.
```

Generate print hello world using rot13 Close

```
# dir defines all the fucntion in python as shown
print(dir(list))
print("\n")
print(dir(tuple))
print("\n")
print(dir(set))
print("\n")
print(dir(dict))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__forma

['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',

['__and__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',

['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge
```

"WHat is Class ? " class is blue print of an object object -> collection of attributes and methods which can replicate any real world entitiy into computer program

pyhon have 2 types of object

1) Mutable -> can change after assignemnet (list, dict, set)

2) Non Mutale -> can not be changes after assignment (tuple, forzenset, int ,float etc)

Some built in function in python

1. type(value) -> print class of that object
2. id(value) -> every object has a uniquw id on python,

```
        if two reference have same id means they are pointing to same object
```

3. dir(value) -> it list all method of given class/ object
4. help(value)-> it will print documentation of a particular

```
help(list) # example
```

```
 |  remove(self, value, /)
 |      Remove first occurrence of value.
 |
 |      Raises ValueError if the value is not present.
 |
 |  reverse(self, /)
 |      Reverse *IN PLACE*.
 |
 |  sort(self, /, *, key=None, reverse=False)
 |      Sort the list in ascending order and return None.
 |
 |      The sort is in-place (i.e. the list itself is modified) and stable (i.e. the
 |      order of two equal elements is maintained).
 |
 |      If a key function is given, apply it once to each list item and sort them,
 |      ascending or descending, according to their function values.
 |
 |      The reverse flag can be set to sort in descending order.
 |
 |  ----------------------------------------------------------------------
 |  Class methods defined here:
 |
 |  __class_getitem__(...) from builtins.type
 |      See PEP 585
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

```
class Student:
  def __init__(self):
    self.name="sachin"
    self.section="A"
    self.roll_no="1234"
    self.dob="01/01/2004"
    self.course="Science"
    #....
#static information
  def info(self):   #Getter
    print(f"Name= {self.name}")
    print(f"Section = {self.section}")
    print(f"Roll No= {self.roll_no}")
    #....

  # to change any given data
  def change_name(self, new_name, new_roll_no, new_section): #Setter
    self.name = new_name
    self.roll_no = new_roll_no
    self.section = new_section
    #....

# in above code, any 2 types of Method are available in a given class
# 1) Attribute ki value get karege -> Getter / get attr
# 2) Attribute ki value set karenge -> Setter / set attr

s1=Student()  #object
print(s1.name)
print(s1.section)
print(s1.roll_no)
print(s1.dob)
print(s1.course) #insted of writting like this, we can directly call the info()

print("\n")

s1.info()
print("\n")

s1.change_name("Aayush Gupta",2306165,"A15")
s1.info()
```

```
sachin
A
1234
01/01/2004
Science
```

```
Name= sachin
Section = A
Roll No= 1234


Name= Aayush Gupta
Section = A15
Roll No= 2306165
```

```
l=["Java","C","C++"]
#list-> pre defined object
# indexing
#print(lang.get__getItem__(2)) #__method__ -> magic methods
#magic method are called indirectely
print(l[2]) # this will print the 2nd index of the list
print("\n")
print(l) # print the complete index as it is
print("\n")
l[1]="Python" #overWrite the 1st index
print(l)
```

```
C++


['Java', 'C', 'C++']


['Java', 'Python', 'C++']
```