DISCOVER OOP TRICKS

```
# THE TRICK BEHIND PYTHON(OOP)
#BY AKANSASIRA ISAAC
```

```
 #In python we use classes to create objects.
#An object is made up of attributes and methods.
#Attributes represent data above the object eg
class Student:
def__init__(self,name,age):
self.name=name
self.age=age
#name and age areattributes.
#we create attributesby using self parameter,followed by the name of
attribute.e.g
self.name=isaac


#We can go ahead and call the Student class and assign it a variable name e.g
 BSIT_STUDENT=Student()

 #then, we can easily access the type of the attribute by typing the name of the
object(BSIT_Student), folllowed by the name of the attribute.eg
 print BSIT_Student.name
 # we can adjust these attributes as below;
 BSIT_STUDENT.Age="24"


 #when we code ;
 def__init__(self,name,age)
 #we have assigned names to attributes, then when we go ahead and code;
 BSIT_STUDENT=Student("AKANSASIRA","24")
```

```
 #AKANSASIRA AND 24 are are now arguments.

                  #THE USE METHODS IN PYTHON
#We can think of methods as functions related to objects.an eg can be ;
def__init__
# you MUST pass the self parametor to a function to make it a method as seen
below

class Student:
    def__init__(self,name,age):
    self.name=name
    self.age=age

    def details(self):
        print("HIS NAME IS" + self.name + "HE IS AGED" + self.age)

    BSIT_Student=Student("AKANSASIRA","24")
    BSIT_Student.details()

    # using the above code, have created a method called details.
    # the selfkey word ensures that all attributes are accessible by all the
methods by specifying thename of the object,followed by the name of the method.eg
    BSIT_Student.details()
class Student:
    def__init__(self,name,age):
    self.name=name
    self.age=age
    dateOfBirth="07/dec/1999"
    def details(self):
        print(dateOfBirth + "IS WHEN HE WAS BORN" )

        BSIT_Student=Student("AKANSASIRA", "24")
        BSIT_Student.details()
```

```
        # the above code brings an ERROR because dateOfBirth is a local variable
of init.
        #being a local variable ,not a grobal variable, detials knows nothing
about it.

        #WE CAN SOLVE THIS PROBLEM  either by just ADDING THE self parameter on
dateOfBirth or WE CAN just leave the variable and we initialize it inside
details, as solved below

        class Student:
            def__init__(self,name,age,dateOfBirth):
    self.name=name
    self.age=age
    self.dateOfBirth=dateOfBirth
    def details(self):
        print(dateOfBirth + "IS WHEN HE WAS BORN" )

        BSIT_Student=Student("AKANSASIRA", "24", "07/dec/1999" )
        BSIT_Student.details()

    #or we can solve it by leaving the dateOfBirth as  a variable, and we
initialize it inside details as seen below

    lass Student:
            def__init__(self,name,age,):
    self.name=name
    self.age=age
    def details(self):
        dateOfBirth="07/12/1999"
        print(dateOfBirth + "IS WHEN HE WAS BORN" )

        BSIT_Student=Student("AKANSASIRA", "24", "07/dec/1999" )
        BSIT_Student.details()
                                IMPORTANCE OF INIT

#we use init to initialize all our attributes.
    #we initialize coz init is normally executed with every new class instance.
```

```
    #we cant callinit, we just define it
    #init isso special, it hasareserved name.
    #init remains init, no matter what.
                        MORE ON SELF PARAMETER

 inside the class while attibutes are variables with "self" prefix.#the self
parameter always represents the current instance of the class.
#whenever we pass it from method to method
#whever we create a new instance of class, a new object, the self key word will
have a different meaning


class Student:
    def__init__(self,name)
    self.name=name
    BSIT = Student("AKANSASIRA")
    BSIT = Student("MARTIN")


    #Lastly, classes are constructors(OPP)
                                END
```