Syntax for creating class:

```
        ==> class <Class_name>:  *class name should start in
    capital*
```

In [ ]:
```
class User:
    pass     # pass to next line
# to initialize an  object from the class use ()

user1 = User()
```

Working with attributes, class constructors and **init()** function

In [ ]:
```
# example
class User:
    pass     # pass to next line
# to initialize an  object from the class use ()

user1 = User()
user1.name = 'aswin'  #name is an attribute which we are creating
user1.id = 204        #age is an another attribute

#creating another one

user2 = User()
user2.name = 'hritick'
user2.id = 203

#creating another one

user3 = User()
user3.name = 'karthi'
user3.age = 207

print(user1.name)
print(user2.name)
print(user3.name)
```

```
aswin
hritick
karthi
```

it may be somewhat difficult and time consuming.

here the constructor comes into action. also known as initializing the class.

syntax:

```
    class <ClassName>:

        def __init__(self):
        # initializing attributes
```

In [ ]:
```
class User:
    def __init__(self):
        print("new user is being created...")

user1 = User()        #whenever we call this class it will initialized and the p
user1.name = 'aswin'  #name is an attribute which we are creating
user1.id = 204        #age is an another attribute
print(user1.name)

#creating another one

user2 = User()
user2.name = 'hritick'
user2.id = 203
print(user2.name)

#creating another one

user3 = User()
user3.name = 'karthi'
user3.age = 207
print(user3.name)
```

```
new user is being created...
aswin
new user is being created...
hritick
new user is being created...
karthi
```

Example: lets take a car with 8 seats, toyoto model and grey color.. I can create a class like this below. Here seats, color and model are the parameters.

In [ ]:
```
class Car:
    def __init__(self, seats,model, color):
        self.seats = seats
        self.model = model
        self.color = color
        self.sold = 0            # we can also assign value here so that it is not
        self.price = 3500000    # if we are having a static value we can only add

# self is the actual object which is being created or initialized
# we can add n number of parameters inside that class like seats, airbag, color

# now i can able to give the number of seats in the car by using below code

my_car = Car(8, "toyoto", "grey")

# Car() is the class, we are settingthe parameters inside the paranthesis, 8 for

'''
we can also use:
-------------------------------------
my_car = Car()
my_car.seats = 8
my_car.model = toyoto
my_car.color = grey
-------------------------------------
'''
```

```
# lets checke whether the parameter is assigned or not

print(my_car.model)
print(my_car.seats)
print(my_car.color)
print(my_car.price)
```

```
toyoto
8
grey
3500000
```

Adding method to the class:

When a function is attached to class, then it is called as Method.

Let's practice with a example of sample instagram code.

In [ ]:
```
class User :
    def __init__(self, user_id, username):
        self.id = user_id
        self.name = username
        self.following = 0
        self.followers = 0

    # imagine self == yourself
    # now we can add a method which describes that, if you follow anyone say a u
    # then the user would have +1 followers and yourself will have +1 following.

    def follow(self,user):
        user.followers += 1
        self.following += 1

    def accept_request(self,user):
        user.following += 1
        self.followers += 1

user_1 = User("mr_smart_solver.official","aswin")
user_2 = User("h_r_i_t_i_c_k","hritick")

user_1.follow(user_2)  # user_1 is following user_2
user_2.follow(user_1)  # user_2 is following user_1

print(user_1.followers)
print(user_1.following)
print(user_2.followers)
print(user_2.following)  # each prints the updated value of the user_1 and user_
```

```
1
1
1
1
```

Final Project of the day

Create a quiz by using OOP concept

data.py

In [ ]:
```
'''
question_data = [
{"text": "A slug's blood is green.", "answer": "True"},
{"text": "The loudest animal is the African Elephant.", "answer": "False"},
{"text": "Approximately one quarter of human bones are in the feet.", "answer":
{"text": "The total surface area of a human lungs is the size of a football pitc
{"text": "In West Virginia, USA, if you accidentally hit an animal with your car
{"text": "In London, UK, if you happen to die in the House of Parliament, you ar
{"text": "It is illegal to pee in the Ocean in Portugal.", "answer": "True"},
{"text": "Google was originally called 'Backrub'.", "answer": "True"},
{"text": "Buzz Aldrin's mother's maiden name was 'Moon'.", "answer": "True"},
{"text": "No piece of square dry paper can be folded in half more than 7 times."
{"text": "A few ounces of chocolate can to kill a small dog.", "answer": "True"}
]
'''
```

question_model.py

In [ ]:
```
'''
class Question :
    def __init__(self, text, answer):
        self.text = text
        self.answer = answer
'''
```

quiz_brain.py

In [ ]:
```
'''
class QuizBrain:
    def __init__(self, question_list):
        self.question_number = 0
        self.score = 0
        self.question_list = question_list

    def still_has_question(self):
        return self.question_number < len(self.question_list)

    def next_question(self):
        # print(question_list[question_number])
        current_question = self.question_list[self.question_number]
        self.question_number += 1
        answer = input(f"Q {self.question_number}: {current_question.text} (True
        self.check_answer(answer,current_question.answer)

    def check_answer(self, answer, correct_answer):
        if answer.lower() == correct_answer.lower():
            self.score += 1
            print('You got it right.!')
        else:
            print("That's wrong")
        print(f"The correct answer is {correct_answer}.")
        print(f"Your current score is {self.score}/{self.question_number}")
```

```
'''
```

main.py

In [ ]:
```
from data import question_data
from question_model import Question
from quiz_brain import QuizBrain

question_bank = []

for question in question_data:
    q_text = question["text"]
    q_answer = question["answer"]
    new_question = Question(text = q_text, answer = q_answer)
    question_bank.append(new_question)

# print(question_bank[0].text)

quiz = QuizBrain(question_bank)
while quiz.still_has_question:
    quiz.next_question()

print("You've completed the quiz")
print(f"Your final score was: {quiz.score}/{quiz.question_number}")
```