Day - 17

**Quiz Game: Class apply**

Create our own classes and create quiz game by applying objects of those classes

**17.1 Creating our own Class**

|  |  |
| --- | --- |
| * "**pass**" key word: In python, **functions**, **ctrl-stmnt** and **classes** needs indentation, if " indentation " is not maintained, the EXCEPTION is thrown. To resolve this, we use "**pass**" to any of the block of **functions**, **ctrl-stmnt** and **classes**, so that we can use those EMPTY-blocks afterward. Consider the following example. | **class** User:  **pass**  user\_1 = **User**()  **def** **funcTn**():  **pass**  **print**("Hello")  #*python class\_demo.py* |

**17.2 Variable naming conventions**

1. camelCase: toString, getJson
2. PascalCase: TomcatServer, SpringBoot, FileNotFoundException

* When the first letter of a camel-cased variable is uppercase, it is also known as Pascal case or upper camel case. When it is not, it is often referred to as lower camel case.

1. snake\_case: this\_is\_snake\_case, build\_docker\_image etc.
2. kebab-case: Used in URLs. For example, user-login-count or www.blog.com/cool-article-1

* Java / Kotlin / C / C++: Languages such as **Java** and **Kotlin**, which have a **C** and **C++** heritage, use
* *lower*C*amelCase* for variables and methods, and
* *PascalCase /upper camel case* for reference types such as enums, classes and interfaces.
* *snake*\_*case* is used for variables.
* Python: In Python as scripting-language style guide, recommend *snake*\_*case* in the instances (where *C-based languages* use *camelCase*). *PascalCase* for classes and *snake\_case* for variables/functions.
* In JavaScript: With HTML and JavaScript when *object-oriented derivations* of **JavaScript**, such as **TypeScript**, **AngularJS** and **Node.JS**, gained popularity, *camelCase* has become the standard for variable and method. (First, *snake*\_*case* used)

**17.3 Non-Initialized class: Inset properties after declaring :** Declared class

|  |  |
| --- | --- |
| * Following used default constructor. ***\_\_init\_\_()*** is invoked behind the scene. * Different Object of ***same*** ***class*** has different properties. | **class** User:  **pass**  user\_1 = **User**()  user\_1**.**nAm = "Beluga"  user\_1**.**id = "001"  **print**(user\_1**.**nAm)  **print**(user\_1**.**id)  #*python class\_demo.py* |

**17.3 Initialized class using** \_\_init\_\_():Defined class

Here inheritance occurs. All object have the common initialize properties/methods of the same class.

***Self:*** This keyword discussed in Day 16.

***\_\_init\_\_(self, p1, p2, p3, ...)***

p1, p2, p3, . . . parameters must be used when an object of that class is created. ***self*** just refers to the class itself during defination.

**class** InitClasDemo:

**def** **\_\_init\_\_**(self, v1, v2, v3):

**self.**ver1 = v1

**self.**mult = v2\*v3

test\_obj = **InitClasDemo**(1, 2, 3)

**print**("multiple is :", test\_obj**.**mult)

* All ***methods*** and ***\_\_init\_\_()*** in a class definition must use the "***self***" keyword to refer the class itself.
* Exercise 17.1: Quiz project part 1. Creating the question model.

|  |  |
| --- | --- |
| * Include the data; * Create a ***Question*** class with an ***\_\_init( )\_\_*** method with two attributes: ***text*** and ***answer*** attribute. | **class** Question:  **def** **\_\_init\_\_**(self, txt, ans):  **self.**text = txt  **self.**answer = ans  new\_1\_q = **Question**("Asdfsf", "True")  **print**(new\_1\_q**.**text)  #*python quiz\_pt\_1\_q\_model.py* |

* Exercise 17.2: Quiz project part 2. Creating a question bank. List of objects.
* Creating the **List** of *Question* *Objects* from the given Data.

1. Write a *for* loop to iterate over the *question\_data*.
2. Create a *Question object* from each entry in *question\_data*.
3. *Append* each *Question object* to the *question\_bank*.

Practice version

**from** quiz\_angela\_data **import** question\_data

**import** quiz\_pt\_1\_q\_model

question\_bank = []

given\_data\_size = **len**(question\_data)

**for** i **in** **range**(0, given\_data\_size):

    ques\_obj = quiz\_pt\_1\_q\_model**.Question**(question\_data[i]["text"], question\_data[i]["answer"])

    question\_bank**.append**(ques\_obj)

**print**(question\_bank)

#*python quiz\_main.py*

Instructor version

**from** quiz\_angela\_data **import** question\_data

**import** quiz\_pt\_1\_q\_model

question\_bank = []

**for** qes **in** question\_data:

    qes\_text = qes["text"]

    qes\_ans = qes["answer"]

    new\_ques\_obj = quiz\_pt\_1\_q\_model**.Question**(qes\_text, qes\_ans)

    question\_bank**.append**(new\_ques\_obj)

**print**(question\_bank)

#*python quiz\_main.py*

* Exercise 17.3: The ***QuizBrain*** and the ***next\_question()*** Method.

|  |  |
| --- | --- |
| Create a class called ***QuizBrain***.  Write an ***\_\_init\_\_( )*** method.  Initialise the ***question\_number*** to ***0***. Initialize the ***question\_list*** to an ***input***. | #TODO: asking the questions  #TODO: checking if the answer was correct  #TODO: checking if we're the end of the quiz |

**from** quiz\_angela\_data **import** question\_data

**import** quiz\_pt\_1\_q\_model

**import** quiz\_quiz\_brain

question\_bank = []

**for** qes **in** question\_data:

    qes\_text = qes["text"]

    qes\_ans = qes["answer"]

    new\_ques\_obj = quiz\_pt\_1\_q\_model**.Question**(qes\_text, qes\_ans)

    question\_bank**.append**(new\_ques\_obj)

quiz = quiz\_quiz\_brain**.QuizBrain**(question\_bank)

quiz**.next\_question**()

#*python quiz\_main.py*

* Exercise 17.4: How to continue showing new Questions

Practice solution:

**while** quiz**.**question\_no **!=** **len**(question\_bank):

    quiz**.next\_question**()

Instructor version

|  |  |
| --- | --- |
| **class** QuizBrain:  **def** **\_\_init\_\_**(self, q\_list):  **self.**question\_no = 0  **self.**question\_list = q\_list  **def** **still\_has\_questions**(self):          #*Return "True" or "False" directly*  **return** **self.**question\_no **<** **len**(**self.**question\_list)                  #*Equivalent to following::*          #*if self.question\_no < len(self.question\_list):*          #*return True*          #*else:*          #*return False*  **def** **next\_question**(self):          current\_question = **self.**question\_list[**self.**question\_no]  **self.**question\_no += 1          ans = **input**(f"Q. {**self.**question\_no}: {current\_question**.**text}, True/False")          #*not using a dictionary so dont use self.question\_list[self.question\_no]['text']. Use "." for object*  #*python quiz\_quiz-brain.py* | **from** quiz\_angela\_data **import** question\_data  **import** quiz\_pt\_1\_q\_model  **import** quiz\_quiz\_brain  question\_bank = []  **for** qes **in** question\_data:      qes\_text = qes["text"]      qes\_ans = qes["answer"]        new\_ques\_obj = quiz\_pt\_1\_q\_model**.Question**(qes\_text, qes\_ans)      question\_bank**.append**(new\_ques\_obj)  quiz = quiz\_quiz\_brain**.QuizBrain**(question\_bank)  **while** quiz**.still\_has\_questions**():      quiz**.next\_question**()  #*python quiz\_main.py* |

* Exercise 17.5: Checking Answers and Keeping Score.

quiz\_quiz-brain.py

**class** QuizBrain:

**def** **\_\_init\_\_**(self, q\_list):

**self.**score = 0

**self.**question\_no = 0

**self.**question\_list = q\_list

**def** **still\_has\_questions**(self):

        #*Return "True" or "False" directly*

**return** **self.**question\_no **<** **len**(**self.**question\_list)

                #*Equivalent to following::*

        #*if self.question\_no < len(self.question\_list):*

        #*return True*

        #*else:*

        #*return False*

**def** **next\_question**(self):

        current\_question = **self.**question\_list[**self.**question\_no]

**self.**question\_no += 1

        ans = **input**(f"Q. {**self.**question\_no}: {current\_question**.**text}, True/False: ")

         #*not using a dictionary so dont use self.question\_list[self.question\_no]['text']. Use "." for object*

**self.is\_correct**(ans, current\_question**.**answer) #*always use "self" to recocnise the current object's method*

**def** **is\_correct**(self, u\_ans, actual\_ans):

**if** u\_ans**.lower**() **==** actual\_ans**.lower**():

**print**("You are right")

**self.**score += 1

**else**:

**print**("Soory you are wrong")

**print**(f"yor score is {**self.**score}/{**self.**question\_no}")

**print**("\n")

#*python quiz\_quiz-brain.py*

quiz\_main.py

**from** quiz\_angela\_data **import** question\_data

**import** quiz\_pt\_1\_q\_model

**import** quiz\_quiz\_brain

question\_bank = []

**for** qes **in** question\_data:

    qes\_text = qes["text"]

    qes\_ans = qes["answer"]

    new\_ques\_obj = quiz\_pt\_1\_q\_model**.Question**(qes\_text, qes\_ans)

    question\_bank**.append**(new\_ques\_obj)

quiz = quiz\_quiz\_brain**.QuizBrain**(question\_bank)

**while** quiz**.still\_has\_questions**():

    quiz**.next\_question**()

**print**("The Quiz if finished !!!!")

**print**(f"Your final score is: {quiz**.**score}/{quiz**.**question\_no}")

#*python quiz\_main.py*

quiz\_pt\_1\_q\_model.py

**class** Question:

**def** **\_\_init\_\_**(self, txt, ans):

**self.**text = txt

**self.**answer = ans

#*new\_1\_q = Question("Asdfsf", "True")*

#*print(new\_1\_q.text)*

#*python quiz\_pt\_1\_q\_model.py*

data.py

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": "True"},

    {"text": "The total surface area of a human lungs is the size of a football pitch.", "answer": "True"},

    {"text": "In West Virginia, USA, if you accidentally hit an animal with your car, you are free to take it home to eat.", "answer": "True"},

    {"text": "In London, UK, if you happen to die in the House of Parliament, you are entitled to a state funeral.", "answer": "False"},

    {"text": "It is illegal to pee in the Ocean in Portugal.", "answer": "True"},

    {"text": "You can lead a cow down stairs but not up stairs.", "answer": "False"},

    {"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.", "answer": "False"},

    {"text": "A few ounces of chocolate can to kill a small dog.", "answer": "True"}

]

**17.4 The BENIFIT of MODULARITY : Can work with other data from other database**

|  |  |
| --- | --- |
| The Benefits of OOP Use ***Open Trivia DB*** to Get New Questions. |  |

|  |  |
| --- | --- |
| question\_data = [      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "medium",          "question": "The HTML5 standard was published in 2014.",          "correct\_answer": "True",          "incorrect\_answers": [              "False"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "medium",          "question": "The first computer bug was formed by faulty wires.",          "correct\_answer": "False",          "incorrect\_answers": [              "True"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "medium",          "question": "FLAC stands for 'Free Lossless Audio Condenser'.",          "correct\_answer": "False",          "incorrect\_answers": [              "True"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "medium",          "question": "All program codes have to be compiled into an executable file in order to be run. This file can then be executed on any machine.",          "correct\_answer": "False",          "incorrect\_answers": [              "True"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "easy",          "question": "Linus Torvalds created Linux and Git.",          "correct\_answer": "True",          "incorrect\_answers": [              "False"          ]      }, | {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "easy",          "question": "The programming language 'Python' is based off a modified version of 'JavaScript'",          "correct\_answer": "False",          "incorrect\_answers": [              "True"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "medium",          "question": "AMD created the first consumer 64-bit processor.",          "correct\_answer": "True",          "incorrect\_answers": [              "False"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "easy",          "question": "'HTML' stands for Hypertext Markup Language.",          "correct\_answer": "True",          "incorrect\_answers": [              "False"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "easy",          "question": "In most programming languages, the operator ++ is equivalent to the statement '+= 1'.",          "correct\_answer": "True",          "incorrect\_answers": [              "False"          ]      },      {          "category": "Science: Computers",          "type": "boolean",          "difficulty": "hard",          "question": "The IBM PC used an Intel 8008 microprocessor clocked at 4.77 MHz and 8 kilobytes of memory.",          "correct\_answer": "False",          "incorrect\_answers": [              "True"          ]      }  ] |

* And we need a little change in our *quiz\_main.py* file. Just change the dictionary-keys:

**for** qes **in** question\_data:

    #*qes\_text = qes["text"]*

    qes\_text = qes["question"]

    #*qes\_ans = qes["answer"]*

    qes\_ans = qes["correct\_answer"]