Steven Cendejas
Mr. Orkney
CSP P.3
21 February 2018

Written Responses

2a: The video illustrates what my program does, what it's about, and how it works. My code works by starting at the top and reads down from top to bottom, it see's that it needs to print the given text and then define introduction. Once it defines introduction it will go down to the next block of code and print the given text. Then it will refer to the if if and else statement, along with the answers given by the user as a raw input. Based on the users answer it, the program will go throw a series of true or false statements in each line of if. If the statement is true it will go in that block, read down and refer to def, where the program will end. If the statement is not true, it will go in the else block and refer to def, where the program will end.

2b: During the incremental development of my program, the difficulties of this opportunity were taken from a collaborative point of view. I ran into some problems when deciding a concept of my code and what I was planning to do, understanding what an algorithm looks like in a program code, or figuring out what story/reference I will use. I resolved this problem during my progress of work by using my classmate, William Flores and teacher, Mr. Orkney as a resource of help to succeed the problem and resolve my understanding of how and what is going on. I came to a conclusion of a bike selector for people who are having trouble deciding what bike is right for them, using a filter that asks the user what the prefer in a bike such as, gears, suspension, speed or tires. During the time when I had to figure out what code format I was going to use, I took the opportunity of being independent and referred to my previous project of replit, that was powered by python. I used the code taken from that as a reference and replaced the story and setting of the code, with my own work and ideas.

2c:
```
answer = raw_input()
  if answer == "g":
      print "gears"
      bike_gears()
  if answer == "s":
      print "suspension"
      bike_suspension()
  if answer == "e":
      print "speed"
      bike_speed()
  else:
    print "tires"
    bike_tires()
```

This block of code and so on helps achieve the purpose of this program using an algorithm because it´s independently and mathematically identifying keys parts of the code that is needed to successfully execute this program. The code “answer = raw_input()” and right under it, “if answer == “g” etc. is an example of an algorithm, for the the raw input in the python kernel is the user's answer that they chose and selected. The user's answer was “g” so the program independently printed gears based on the users raw answer input of “g”. From there on, the code goes through a series of true or false statements independently and checks every if and else statement to identify (for example) if the answer is in fact “g” then, the program then reads the definition associated with the if statement with the answer “g” and then executes the command to take it to the next step with “def”. In this case the answer ”g” is defined as gears for that is what I programed it to do. With it being defined as gears I had to make a code block for def() bikegears.

2d:
```
def select_bike():
  print "Are you looking for Gears, suspension, speed or tires in a bike"
  print ("\n")
  print "Type g for gears, s for suspension, e for speed, or t for tires"
  answer = raw_input()
  if answer == "g":
      print "gears"
      bike_gears()
  if answer == "s":
      print "suspension"
      bike_suspension()
  if answer == "e":
      print "speed"
      bike_speed()
  else:
    print "tires"
    bike_tires()
```
In this program of a bike selector that I´ve developed individually on my own, I´ve developed it in a way to where there is a wide variety of choices with decisions to pick from. There are a total of four different types of choices that the user can input as an answer based on what they prefer to have as a upgrade in their bike. These inputs include gears, suspension, speed and tires. Along with a total of four different types of outputs, based on what the user´s first input was. These outputs include a mountain bike, cruiser bike, road bike or an aluminum trek bike. Mathematically from an abstraction, this program runs and reads through a series of three different given if statements in the program, if the users input is not found in the if statements, the program will independently refer to the one else statement and print what it is programmed to say.