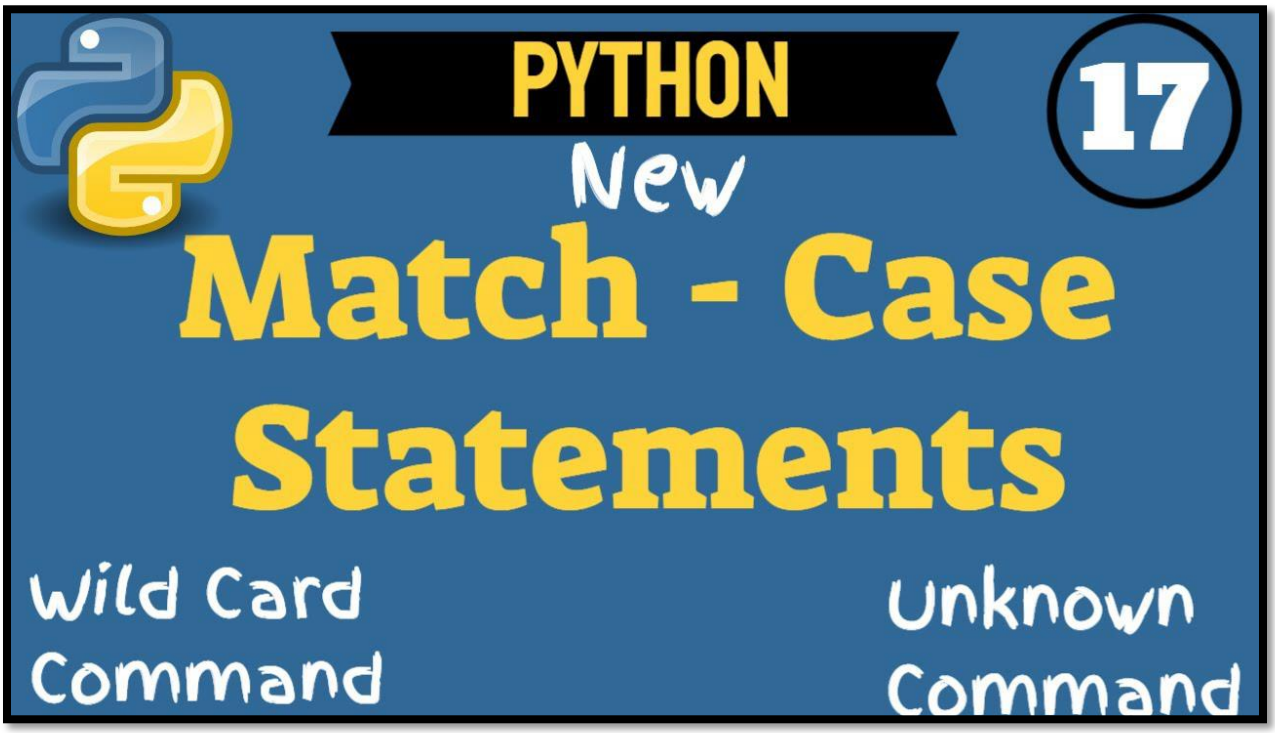
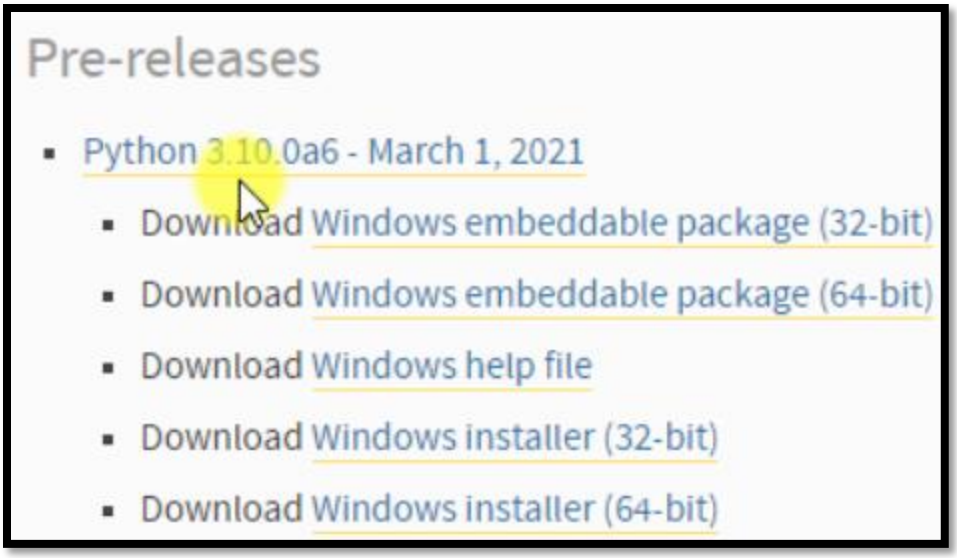
Rex Jones II

New Python
Match – Case Statements

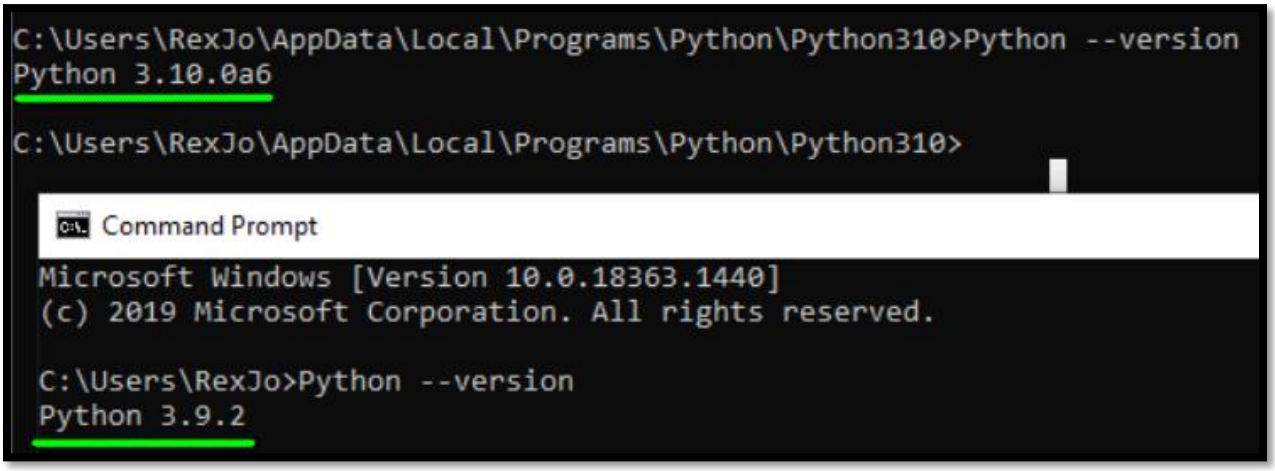


Python Video = https://youtu.be/Cj4VmVc13TY

In this session, we are going to use the match case statement using Python. It has a feature like the switch case statement from other languages but the match case statement has more features. We must install Python 3.10. This is alpha 6. I clicked the Windows Installer (64-bit).



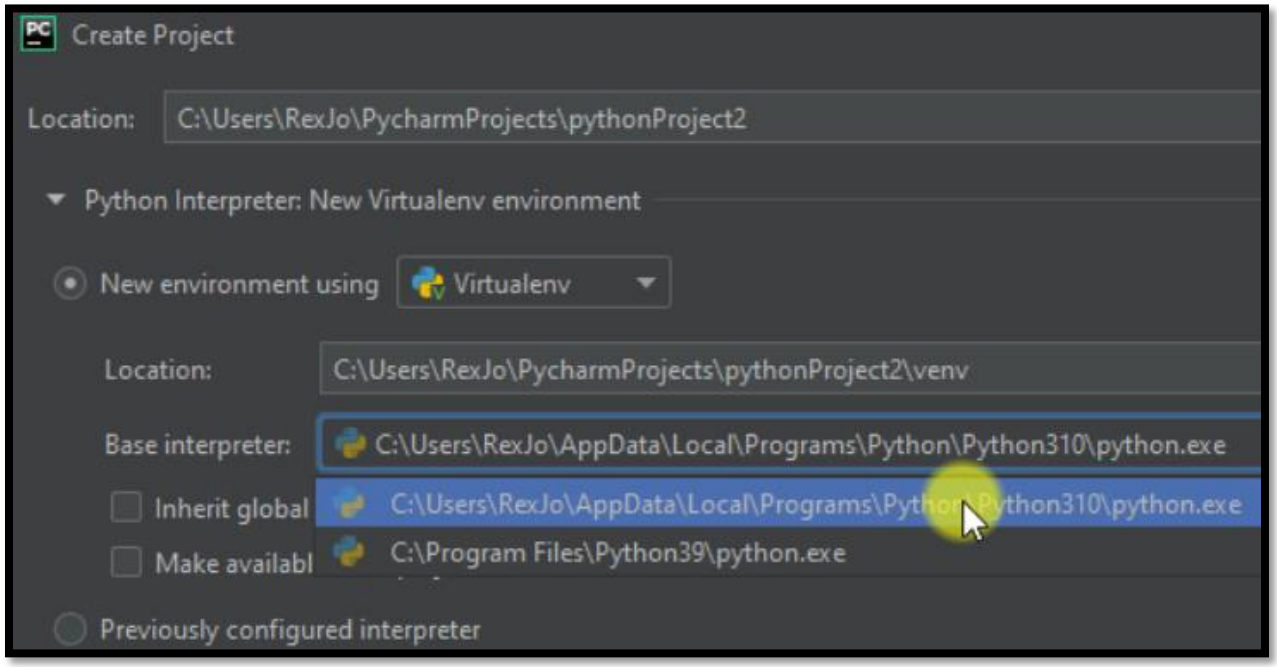
The executable file was added to my Downloads folder and Python was installed in the Programs folder. If I go to my command prompt and type Python --version. We see Python 3.9.2 but if I go to my Python310 folder then type cmd. Now if I type Python –version. You will see Python 3.10.0a6. I have both versions in my system.



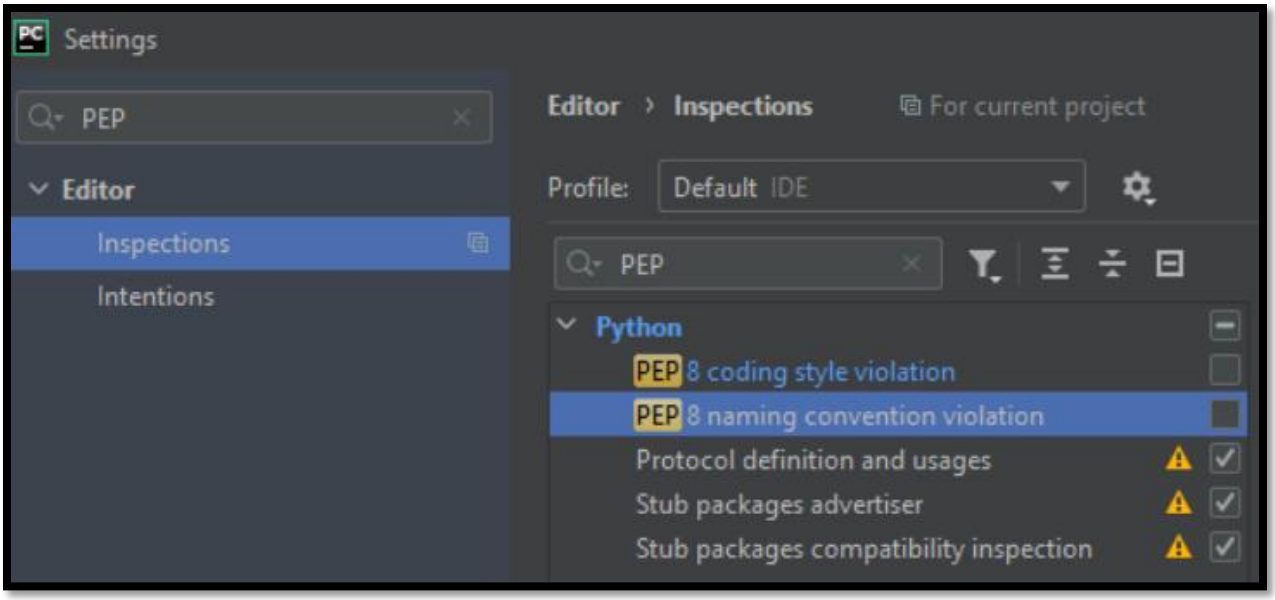
If you want more information then you can go to What's New In Python 3.10. New Feature, Parenthesized context managers, Better error messages in the parser, PEP 626: Precise line numbers for debugging but in this session I will demo PEP 634: Structural Pattern Matching. We see the description shows Structural pattern matching has been added in the form of a match statement and case statements of patterns with associated actions and here's the syntax.

```
match subject:
    case <pattern_1>:
        <action_1>
    case <pattern_2>:
        <action_2>
    case <pattern_3>:
        <action_3>
    case _:
        <action wildcard>
```

We see more features if I keep scrolling in this website. When I go to the console, there's something else I must do for Python. In PyCharm, the IDE I'm going to File – New Project then change the Base Interpreter to Python 310 and click the Create button.



There's one more thing. Go to File – Settings then type PEP in the search field. Uncheck both checkboxes under Python for PEP 8 coding style violation and convention violation. If we do not uncheck the boxes, our program will still execute but will show Syntax error like there is a problem although there is not a problem.



Now, let's write our code by starting with an input() function "Enter Car Name: " then assign the value to car =. Next is match car: The purpose of match is to take an expression then compare the expression value. Car is the variable name that will receive a value. We are going to evaluate car in the match statement. Next is to compare car with a pattern by writing case "audi": This is one of the patterns that will be compared with an action of print("Your car is an Audi"). If the value is audi then the print statement will take action by displaying Your car is an Audi on the console. The next pattern is case "bmw": print("Your car is a BMW"). Let's write one more pattern case "cadillac": print("Your car is a Cadillac").

```
car = input("Enter Car Name: ")

match car:
    case "audi":
        print("Your car is an Audi")
    case "bmw":
        print("Your car is a BMW")
    case "cadillac":
        print("Your car is a Cadillac")
```

The variable car will be compared to each pattern from top to bottom until a match is confirmed. If a match is not confirmed then we can take 2 approaches. We can write case with a wildcard which is an underscore _:

```
match car:
    case "audi":
        print("Your car is an Audi")
    case "bmw":
        print("Your car is a BMW")
    case "cadillac":
        print("Your car is a Cadillac")
    case _:
```

or a value like unknown: Both approaches is a catch all statement that will catch any value that's not in the case value pattern.

```
match car:
    case "audi":
        print("Your car is an Audi")
    case "bmw":
        print("Your car is a BMW")
    case "cadillac":
        print("Your car is a Cadillac")
    case unknown:
```

Let's look at both starting with unknown. The action is print("Your {unknown} Car Is Unknown"). Did you see how automatically an 'f' was placed in front of the quotes? That's how we format the string.

```
case unknown:
  print(f"Your {unknown} Car Is Unknown")
```

Let's Run. Enter Car Name: audi The console returned Your car is an Audi.

```
Enter Car Name: audi
Your car is an Audi
```

Run again. This time, I'm going to enter bmw. The Console shows Your car is a BMW.

```
Enter Car Name: bmw
Your car is a BMW
```

How about cadillac? The console shows Your car is a Cadillac.

```
Enter Car Name: cadillac
Your car is a Cadillac
```

For the catch all statement unknown command, let's enter tempo. The console shows Your tempo Car is Unknown. I like the unknown command

```
Enter Car Name: tempo
Your tempo Car Is Unknown
```

Now for the wildcard underscore. Change unknown to an underscore _ then update the print statement to "Your Value Is Not an Audi, BMW, or Cadillac".

```
case _:
  print("Your Value Is Not an Audi, BMW, or Cadillac")
```

Let's Run. Enter Car Name. How about camry. We see Your Value Is Not an Audi, BMW, or Cadillac. That's it for the new match case statement in Python using the Unknown command and wildcard.

Rex Jones II

```
Enter Car Name: camry
Your Value Is Not an Audi, BMW, or Cadillac
```

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/