microsoft / **aed-wonder-woman**

generated from microsoft/aed-learn-template

Code Issues Pull requests Actions Projects Wiki Security

main

aed-wonder-woman / 0-wonderwomanlp / 3-superquiz / includes / **2-python-basics-for-quiz-making.md**

sguthals images are in **History**

1 contributor

Raw Blame

122 lines (84 sloc) 5.86 KB

Python Basics Needed for a Wonder Woman Personality Quiz

Before you dive into the full personality quiz, let’s walk you through the basics for commanding the Python. If you already know how to use variables, functions, and conditionals, you can skip to the next unit in this module.

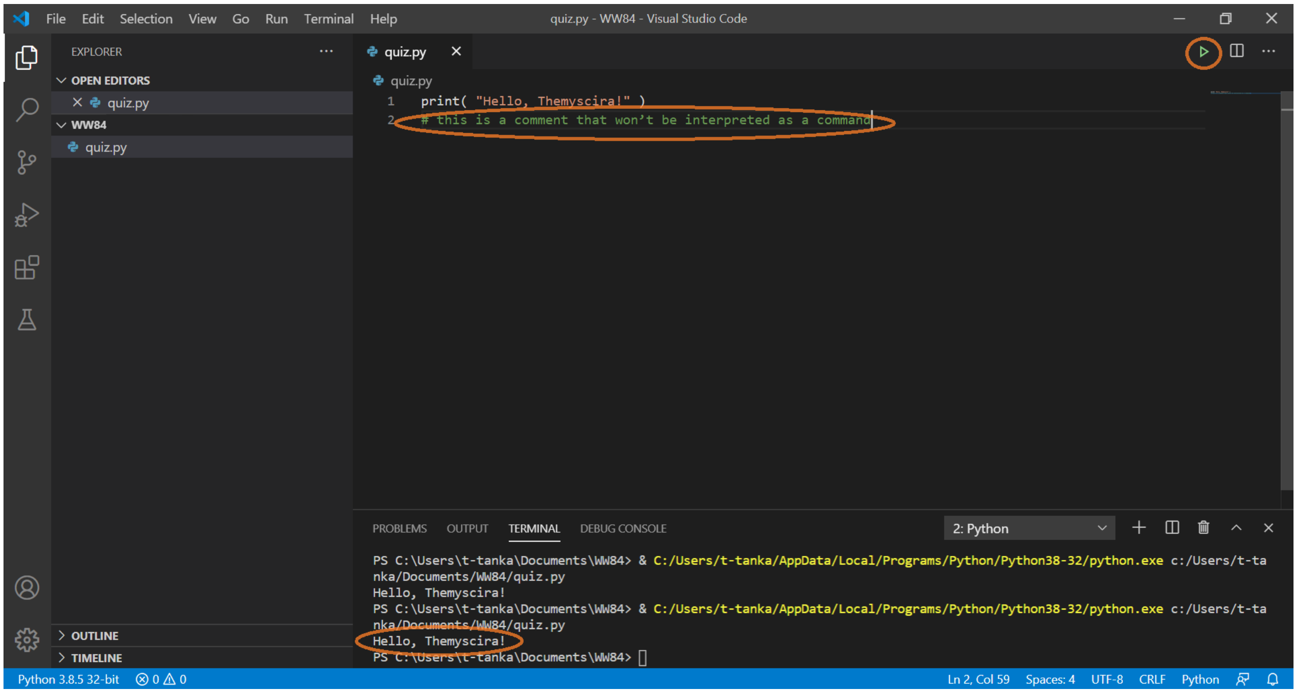
Comments

First, we can write comments to ourselves that the Python will ignore. On any line where we use the # symbol, the Python ignores everything after.

Try adding the following to your file:

```
# this is a comment that won’t be interpreted as a command
```

Press the Play button, and you should see the same behavior as before.



```
print( "Hello, Themyscira!" )
# this is a comment that won’t be interpreted as a command
```

```
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-ta
nka/Documents/WW84/quiz.py
Hello, Themyscira!
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-ta
nka/Documents/WW84/quiz.py
Hello, Themyscira!
PS C:\Users\t-tanka\Documents\WW84> 
```

Variables

If you want the Python to remember something, you can command it to “write it down” using a variable.

The following command tells the Python to remember the value 1984 using a variable named “year” with the special assignment `=` sign operator. Any time after this command, when the Python sees “year”, it will substitute the current value.

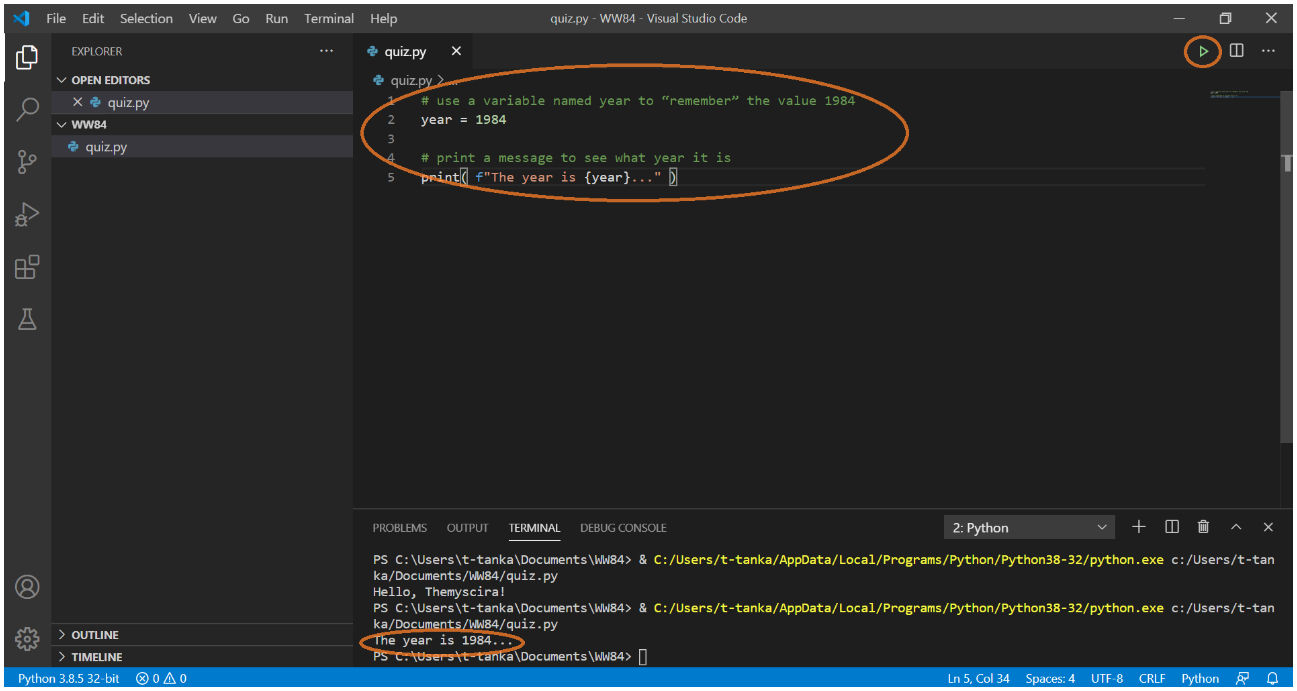
```
# use a variable named year to “remember” the value 1984
year = 1984
```

Try it out by replacing the commands in your file with the following. The print command can, if you put an f before what you want printed, use curly braces {} to surround a variable name -- this will make the Python print out the value of the variable.

```
# use a variable named year to “remember” the value 1984
year = 1984

# print a message to see what year it is
print( f"The year is {year}..." )
```

Press the Play Button, and you should see the year print out.



```
# use a variable named year to “remember” the value 1984
year = 1984

# print a message to see what year it is
print( f"The year is {year}..." )
```

```
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-tan
ka/Documents/WW84/quiz.py
Hello, Themyscira!
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-tan
ka/Documents/WW84/quiz.py
The year is 1984...
PS C:\Users\t-tanka\Documents\WW84> 
```

We can update the value of a variable using the same assignment (= sign) operator. The Python will figure out the right-hand side before storing it in the variable. Since “year” already has a value, this command will ask the Python to first substitute the current value of “year” (which is 1984), add 36 to it, then store that value in the variable “year”. This effectively overwrites the value that was written down originally. Try it out by replacing the commands in your file with the following.

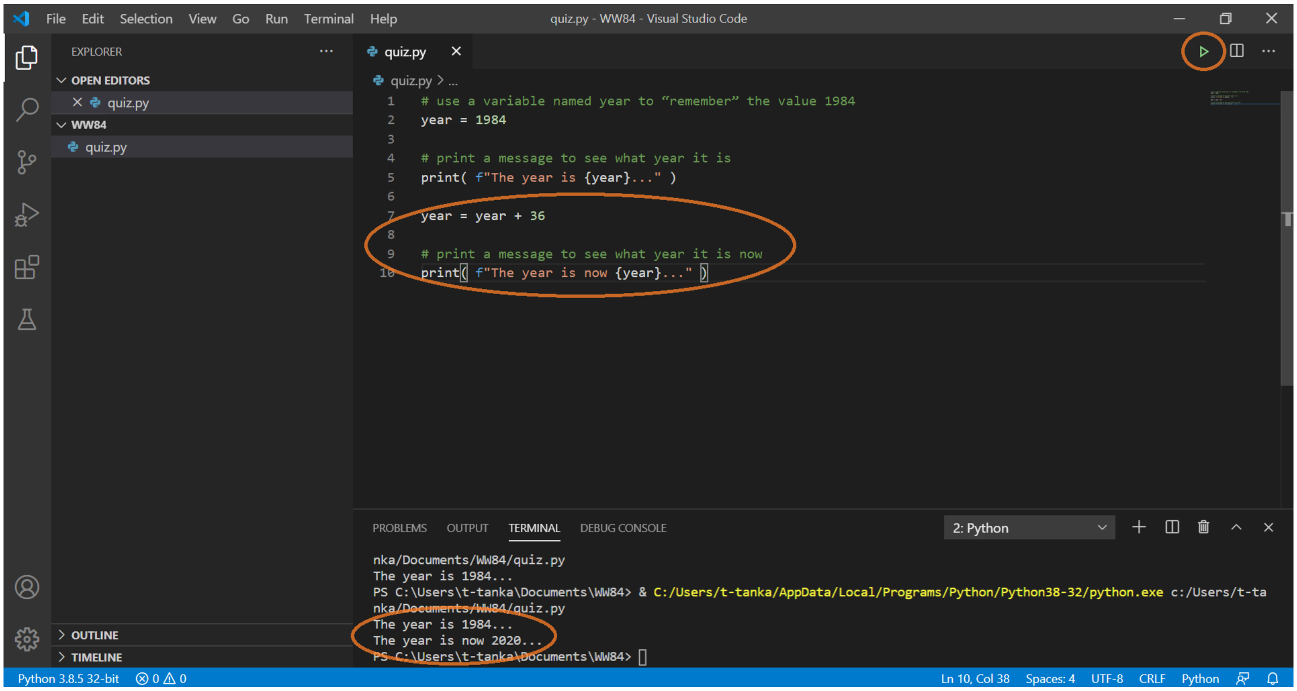
```
# use a variable named year to “remember” the value 1984
year = 1984

# print a message to see what year it is
print( f"The year is {year}..." )

year = year + 36

# print a message to see what year it is now
print( f"The year is now {year}..." )
```

Press the Play Button, and you should see the years print out.



```
# use a variable named year to “remember” the value 1984
year = 1984

# print a message to see what year it is
print( f"The year is {year}..." )

year = year + 36

# print a message to see what year it is now
print( f"The year is now {year}..." )
```

```
nka/Documents/WW84/quiz.py
The year is 1984...
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-ta
nka/Documents/WW84/quiz.py
The year is 1984...
The year is now 2020...
PS C:\Users\t-tanka\Documents\WW84> 
```

Conditionals

The Python can perform commands depending on some condition. We’re used to conditionals in the real world: “If you’ve been lassoed, then you're compelled to tell the truth.” The keyword here is `if` , which is followed by a condition, where we check whether or not the lasso of truth is around you. When the condition is met, the subsequent command must be followed. Python code only expects a condition to either be met or not — it’s either true or false.

To get a little more comfortable, consider the following strategy for how to spend a day *(the following is not code, it's just plain text to explain a scenario)*:

```
if it’s raining cats and dogs
    find a cat
    if the cat is hungry
        find some milk
        give the milk to the cat
    else
        pet the cat until it purrs or scratches you
else
    be happy that it’s not raining animals
    frolic outside
```

Now, how could this play out? It could be the case that:

- It’s raining cats and dogs
- The cat you find is hungry

In this scenario, you will do three things:

- You find a cat.
- You find some milk.
- You give the milk to the cat.

What if instead:

- It’s not raining cats and dogs?

In this scenario, you will do two things:

- You are happy that it’s not raining animals.
- You frolic outside.

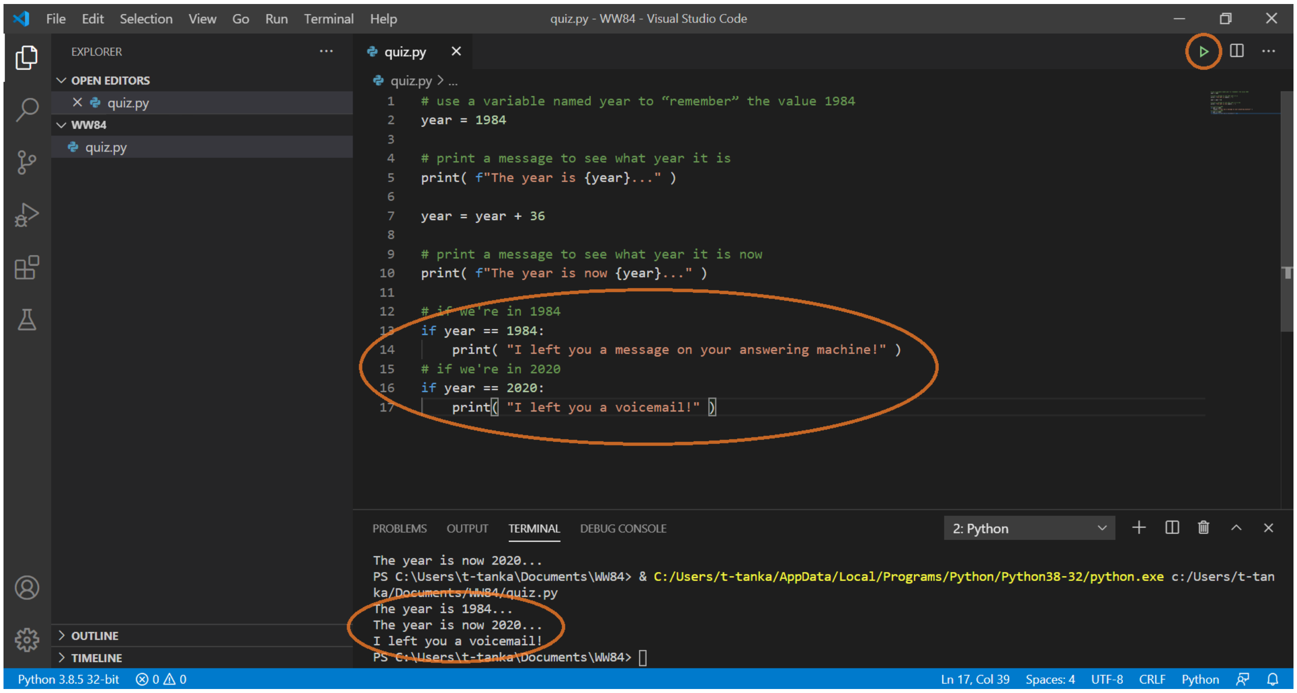
There are a few more possible scenarios, but let’s move on to how we can use conditionals to command the Python in our year example.

Conditionals in Python

Python expects a particular “syntax” (formulation of the commands). We won’t go into depth here; instead, just try out this example by adding the following to your file. (If you’re wondering what the `==` is doing, that is how we test if two numbers are equal; remember that a single `=` means assignment of a variable, so the computer would be confused).

```python
# if we're in 1984
if year == 1984:
    print( "I left you a message on your answering machine!" )
# if we're in 2020
if year == 2020:
    print( "I left you a voicemail!" )
```

Press the Play button, and you should see the state-of-the-art in messaging.



```python
# use a variable named year to “remember” the value 1984
year = 1984

# print a message to see what year it is
print( f"The year is {year}..." )

year = year + 36

# print a message to see what year it is now
print( f"The year is now {year}..." )

# if we're in 1984
if year == 1984:
    print( "I left you a message on your answering machine!" )
# if we're in 2020
if year == 2020:
    print( "I left you a voicemail!" )
```

```
The year is now 2020...
PS C:\Users\t-tanka\Documents\WW84> & C:/Users/t-tanka/AppData/Local/Programs/Python/Python38-32/python.exe c:/Users/t-tan
ka/Documents/WW84/quiz.py
The year is 1984...
The year is now 2020...
I left you a voicemail!
PS C:\Users\t-tanka\Documents\WW84> 
```

How would you change the year so that you get a message on your answering machine instead?

Booleans (extra)

Python puts a special meaning on the words “true” and “false”, and calls them boolean values. It turns out that booleans are studied in computer science (and philosophy) and there is an entire subject called boolean logic!

You might be comfortable with arithmetic, where numbers are operated on to produce other numbers; 1 + 2 is 3. The operands are the numbers 1 and 2, and the operator is the addition operator (+). In boolean logic, boolean values (True or False) are operated on to produce other boolean values. We might say, “if it is cold outside AND it is raining, then grab a parka.” Then both conditions must be met (must be True) for us to grab a parka. Here the operands are whether or not it is cold outside and whether or not it is raining; the AND operator

Now that we have the handle on some of the basics, let’s move on to making the quiz.