module1b

January 22, 2024

0.1 "How Old is Your Dog in Human Years?" Calculator

In this assignment, you will write a program that calculates a dog's age in human years. The program will prompt the user for an age in dog years and calculate that age in human years. Allow for int or float values, but check the user's input to make sure it's valid – it should be numeric and positive. Otherwise, let the user know their input is not valid.

You can use the following rules to approximately convert a medium-sized dog's age to human years: - For the first year, one dog year is equal to 15 human years - For the first 2 years, each dog year is equal to 12 human years - For the first 3 years, each dog year is equal to 9.3 human years - For the first 4 years, each dog year is equal to 8 human years - For the first 5 years, each dog year is equal to 7.2 human years - After that, each dog year is equal to 7 human years. (Note: This means the first 5 dog years are equal to 36 human years (5 * 7.2) and the remaining dog years are equal to 7 human years each.)

Print the result in the following format, substituting for **dog_age** and **human_age**: "The given dog age **dog_age** is **human_age** in human years." Round the result to 2 decimal places. Note: If there is a 0 in the hundredths place, you can drop it, e.g. 24.00 can be displayed as 24.0.

Considering invalid inputs: * Your program must ask the user for an age in dog years - hint: use the input() function * We are going to test invalid inputs - make sure that your code can handle negative value inputs and non-numerical inputs! * For invalid inputs, make sure that your printed response adheres to the following: - If a text-based input is provided, make sure your response contains the word 'invalid'.
- If a negative input is provided, make sure your response contains the word 'negative'.

0.2 A note on defining functions

Defining a function refers to the act of creating a function using the `def` keyword and giving it a name, as well as any arguments that need to be available in the body of the function. We'll see more about functions later in the course, but below the `def calculator():` line is the beginning of a function declaration, with all of the indented lines below it as the body of the function.

Similar to the way we name variables, a function may not contain spaces or special characters, with the exception of the underscore (_). Below are some examples of functions being defined.

```python
def say_hello():
    print("Hello!")
```

The above function named `say_hello` prints the word `"Hello!"`. It takes in no arguments - that is, there is nothing being passed into the function inside of the parentheses that would be available to the body of the function. Let's look an another example that does contain an argument:

```
def say_something_specific(thing_to_say):
    print(thing_to_say)
```

Here we have a function named `say_something_specific` with an argument `thing_to_say`. The argument could be anything (whenever we decide to call this function, we'll pass in whatever we'd like at the time we call it) and will be printed out similar to the way `"Hello!"` was printed in the first function.

```
def number_sum(num1, num2):
    sum = num1 + num2
    print("Sum is", sum)
    return sum
```

This function, `number_sum`, takes in two arguments separated by commas, named `num1` and `num2`. Arguments are essentially variables that are accessible throughout the function body. In this case, a variable `sum` is created with a value equal to the sum of our two arguments, `num1` and `num2`. We then print out `"Sum is <sum>"`, then finally **`return`** the sum. The return allows us to get a value out of the function after it's done executing. Whatever follows the `return` keyword will be sent back to the location in your code where the function was called. Let's take a look at one more example of how a function could be defined, then used in code. We'll use the function that we just created, `number_sum`.

```
# ...

a = 5
b = 3
sum_a_b = number_sum(a, b)
# At this point in the code, number_sum returned a (5) + b (3) = 8, so the value of
# sum_a_b = 8

sum_x_y = number_sum(10, 20)
# After the above line is executed, sum_x_y will be equal to 30
```

[78]:
```
import traceback

def calculator():

    # Get dog age
    age = input("Input dog years: ")

    try:
        # Cast to float
        d_age = float(age)

        # If user enters negative number, print message
        # Otherwise, calculate dog's age in human years
        # your code here

        if d_age < 0:
            print("The number can't be negative")
        elif d_age == 1:
            human_age = 15.0
        elif d_age == 2:
            human_age = 24.0
        elif d_age == 3:
            human_age = 27.9
        elif d_age == 4:
            human_age = 32.0
        elif d_age == 5:
            human_age = d_age * 7.2
        else:
            human_age = (5 * 7.2) + (d_age - 5) * 7

        print("The given dog age {} is {} in human years.".format(age,␣
↪human_age))

    except:
        print(age, "is an invalid age.")
        print(traceback.format_exc())

calculator()
```

```
Input dog years: 1
The given dog age 1 is 15.0 in human years.
```

[79]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
↪format(checker.test_one(1)))
checker.assert_almost_equal(
    checker.parse_dog_age(checker.test_one(1)),
    15,
    "Expected a 1 year old dog to be 15 in human years. Make sure your print␣
↪statement is formatted as:\n\t \
    'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
```

```
---------------
'The given dog age 1 is 15.0 in human years.
'
Success!
```

[80]:

```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_two(2)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_two(2)), 24,
    "Expected a 2 year old dog to be 24 in human years. Make sure your print␣
 ↪statement is formatted as:\n\t \
    'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 2 is 24.0 in human years.
'
Success!
```

[81]:

```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_three(3)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_three(3)), 27.9,
            "Expected a 3 year old dog to be 27.9 in human years. Make sure␣
 ↪your print statement is formatted as:\n\t \
    'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 3 is 27.9 in human years.
'
Success!
```

[82]:

```
##########################
### TEST YOUR SOLUTION ###
```

```
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_four(4)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_four(4)), 32,
            "Expected a 4 year old dog to be 32.0 in human years. Make sure␣
 ↪your print statement is formatted as:\n\t \
    'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 4 is 32.0 in human years.
'
Success!
```

[83]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_five(5)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_five(5)), 36,
            "Expected a 5 year old dog to be 36.0 in human years. Make sure␣
 ↪your print statement is formatted as:\n\t \
    'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 5 is 36.0 in human years.
'
Success!
```

[84]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_six(6)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_six(6)), 43,
            "Expected a 6 year old dog to be 43.0 in human years. Make sure␣
↪your print statement is formatted as:\n\t \
        'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 6 is 43.0 in human years.
'
Success!
```

[85]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
↪format(checker.test_seven(7)))
checker.assert_almost_equal(checker.parse_dog_age(checker.test_seven(7)), 50.0,
            "Expected a 7 year old dog to be 50.0 in human years. Make sure␣
↪your print statement is formatted as:\n\t \
        'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 7 is 50.0 in human years.
'
Success!
```

[86]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
↪format(checker.test_twopointfive(2.5)))
checker.assert_true(checker.parse_dog_age(checker.test_twopointfive(2.5)) >= 23␣
↪and
                    checker.parse_dog_age(checker.test_twopointfive(2.5)) <= 29,
            "Expected a 2.5 year old dog to be between 23 and 29 in human years.
↪ Make sure your print statement is formatted as:\n\t \
        'The given dog age <input_value> is <calculated_age> in human years.'")
print("Success!")
```

```
Received the below output from your program:
---------------
'The given dog age 2.5 is 18.5 in human years.
'


      ␣
 ↪---------------------------------------------------------------------------

        AssertionError                            Traceback (most recent call␣
 ↪last)

        <ipython-input-86-981676349107> in <module>
          9                     checker.parse_dog_age(checker.
 ↪test_twopointfive(2.5)) <= 29,
         10             "Expected a 2.5 year old dog to be between 23 and 29 in␣
 ↪human years. Make sure your print statement is formatted as:\n\t \
    ---> 11     'The given dog age <input_value> is <calculated_age> in human␣
 ↪years.'")
         12 print("Success!")


        /opt/conda/lib/python3.7/unittest/case.py in assertTrue(self, expr, msg)
        703         if not expr:
        704             msg = self._formatMessage(msg, "%s is not true" %␣
 ↪safe_repr(expr))
    --> 705             raise self.failureException(msg)
        706
        707     def _formatMessage(self, msg, standardMsg):


        AssertionError: False is not true : Expected a 2.5 year old dog to be␣
 ↪between 23 and 29 in human years. Make sure your print statement is formatted␣
 ↪as:
                 'The given dog age <input_value> is <calculated_age> in human␣
 ↪years.'
```

[87]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
 ↪format(checker.test_negative(-1)))
```

```
checker.assert_true('negative' in checker.test_negative(-1).lower(), "Make sure␣
↪negative inputs print out a message containing the word negative!")
print("Success!")
```

```
Received the below output from your program:
---------------
'The number can't be negative
-1 is an invalid age.
Traceback (most recent call last):
  File "<ipython-input-78-f52096d44553>", line 32, in calculator
    print("The given dog age {} is {} in human years.".format(age, human_age))
UnboundLocalError: local variable 'human_age' referenced before assignment

'
Success!
```

[88]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
import checker
checker.calculator = calculator

print("Received the below output from your program:\n---------------\n'{}'".
↪format(checker.test_text_input('n')))
checker.assert_true('invalid' in checker.test_text_input('n').lower(), "Make␣
↪sure text inputs print out a message containing the word invalid!")
print("Success!")
```

```
Received the below output from your program:
---------------
'n is an invalid age.
Traceback (most recent call last):
  File "<ipython-input-78-f52096d44553>", line 10, in calculator
    d_age = float(age)
ValueError: could not convert string to float: 'n'

'
Success!
```

[ ]:
```
```

[ ]:
```
```