Homework: Sound frequency

Background

An important quality of periodic sound (that is, sound that occurs in a repeating pattern, such as a sine wave) is *frequency*—the rate at which the pattern repeats. Frequency is usually measured in hertz (Hz), the number of cycles per second.

Humans perceive frequency as pitch, with higher frequencies corresponding to higher pitches. Our perception of pitch is logarithmic; for example, we perceive the difference between 100 Hz and 200 Hz to be much greater than the difference between 1000 Hz and 1100 Hz.

In music, the interval between two pitches whose frequencies differ by a factor of 2 is called an octave. Humans perceive pitches an octave apart as belonging to the same pitch class; this is called "octave equivalence"[1]. In modern Western music, octaves are subdivided into twelve smaller, exponentially spaced steps called "semitones" or "half steps". The system of exponential spacing is called "equal temperament".

The sequence of ascending semitones within an octave is called a chromatic scale. In English, the following names are commonly used for the pitches in the chromatic scale: A, A♯, B, C, C♯, D, D♯, E, F, F♯, G, G♯ (the ♯ symbol is pronounced "sharp").[2][3] The next half-step above G♯ is A, one octave above the previous A; the pattern repeats infinitely. A system called "scientific pitch notation" is used to simultaneously indicate the pitch class and octave of a musical pitch. In this system, a letter (and sharp or flat sign, if applicable) indicating the pitch class is followed by a subscript number representing an octave. By convention, the lowest pitch of a given octave is C. C_1 is the lowest C on a piano; C_8 is the highest C on the piano. $C_{♯1}$ is the C♯ above C_1; D_1 is the D above C_1, and so on.

In equal temperament, the canonical (or optimally tuned) frequencies of all pitches can be calculated relative to a reference pitch whose frequency is known. By modern convention, the standard reference pitch is A_4 and the standard frequency used for this pitch is 440 Hz. (Some people assign a different frequency to A_4; 432 Hz is a popular alternative at present.)

Most people accustomed to Western music will perceive all musical pitches in terms of the twelve pitch classes of the chromatic scale, including pitches other than the canonical frequencies for these pitch classes. Pitches that deviate significantly from their nearest canonical frequency are said to be "out of tune"; pitches that are too high are called "sharp", while those that are too low are called "flat". A common unit for measuring differences of intonation is the "cent", which represents a 100th of a semitone or a 1200th of an octave.

Different animals are able to perceive different ranges of frequencies. Humans have a nominal hearing range of 20 Hz - 20 kHz,[3] although adults gradually lose their ability to hear the highest frequencies in this range. Dogs can hear frequencies as high as 44 kHz; cats can go up to 77 kHz. Porpoises can hear over an octave above that, up to 150 kHz, and greater wax moths can hear frequencies as high as 300 kHz, making them the animals with the highest hearing range of all. On the low end, pigeons have been shown to perceive frequencies as low as 0.5 Hz.

Problem statement

Write a program that takes a frequency and, optionally, a reference frequency for A_4, and determines

- the corresponding pitch and octave
- the degree to which the frequency is out of tune, given the reference frequency
- which species from a prioritized list can hear the frequency, if any

---

Instructions

Using the provided template file, `frequency.py`, implement the following functions. The template imports some functionality from standard libraries, notably the `log2()` function for calculating logarithms of base 2. It includes a function, `parse_args()`, which processes command-line arguments, and an `if __name__ == "__main__":` statement which runs the entire program. You will not need to edit the template code other than to remove the comments on lines 9-14. You will write your functions after the import statements and before the `parse_args()` function.

`get_pitch()` function

Write a function named `get_pitch()`. This function will determine the pitch of the specified frequency.

Your function should take one required parameter (a frequency whose pitch you will determine) and one optional parameter (a value to use for the note A_4; use 440.0 as the default value for this parameter).

Use the formula

$$\left\lfloor 12 \left(\log_2 \left(\frac{f}{A_4} \right) \bmod 1 \right) \right\rceil$$

(where $\lfloor x \rceil$ means round x to the nearest integer, f is the frequency whose pitch you wish to determine, A_4 is the value to use for the note A_4, and mod is the modulo operation). This formula will result in a number between 0 and 11, where the numbers correspond to the following notes:

Number	Pitch
0	A
1	A♯
2	B
3	C
4	C♯
5	D
6	D♯
7	E
8	F
9	F♯
10	G
11	G♯

Convert the result of the formula to a string (use the hash character `#` in place of the sharp sign ♯) and return the string.

`get_octave()` function

Write a function named `get_octave()`. This function will determine which octave the specified frequency belongs to, as an integer. For example, A_4 belongs to octave 4.

Your function should take one required parameter (a frequency whose octave you will determine) and one optional parameter (a value to use for the note A_4; use 440.0 as the default value for this parameter).

Use the formula

$$\left\lfloor \frac{\left\lfloor 12 \left(\log_2 \left(\frac{f}{A_4} \right) \right) \right\rceil + 57}{12} \right\rfloor$$

(where $\lfloor x \rceil$ means round x to the nearest integer, $\left\lfloor \frac{y}{z} \right\rfloor$ means use floor division to divide y by z, f is the frequency whose octave you wish to determine, and A_4 is the value to use for the note A4). Return the result of this calculation.

`check_intonation()` function

Write a function named `check_intonation()`. This function will determine how far out of tune the specified frequency is, in cents, as an integer. A negative number will indicate that the specified frequency is flat; a positive number will indicate that the specified frequency is sharp; 0 will indicate that the specified frequency is in tune.

Your function should take one required parameter (a frequency whose intonation you will analyze) and one optional parameter (a value to use for the note A_4; use 440.0 as the default value for this parameter).

Use the formula

$$\left\lfloor 1200 \left(\log_2 \left(\frac{f}{A_4} \right) \right) + 50 \right\rfloor \bmod 100 - 50$$

(where $\lfloor x \rceil$ means round x to the nearest integer, f is the frequency whose intonation you wish to analyze, A_4 is the value to use for the note A_4, and mod is the modulo operation). Return the result of this calculation.

`who_can_hear()` function

Write a function named `who_can_hear()`. This function will return a string indicating a species that can hear a particular frequency, or `None` if it is not aware of a species that can hear the specified frequency.

Your function should take one required parameter (a frequency).

Here are the values the function should return, in order from highest to lowest priority:

- `"human"`, for frequencies between 20 and 20,000 Hz, inclusive
- `"pigeon"`, for frequencies below the range of human hearing down to 0.5 Hz, inclusive
- `"dog"`, for frequencies above the range of human hearing up to 44,000 Hz, inclusive
- `"cat"`, for frequencies above the range of dog hearing up to 77,000 Hz, inclusive
- `"porpoise"`, for frequencies above the range of cat hearing up to 150,000 Hz, inclusive
- `"greater wax moth"`, for frequencies above the range of porpoise hearing up to 300,000 Hz, inclusive
- `None` (note: this is not a string!) for all other frequencies

`main()` function

Write a function named `main()`. This function will invoke the other functions and display output to the user.

Your function should take one required parameter (a frequency to be analyzed) and one optional parameter (a value to use for the note A_4; use 440.0 as the default value for this parameter).

Your function should raise a ValueError if the frequency to be analyzed is less than or equal to zero.

Your function should print three lines to stdout. The first line should indicate the frequency, pitch, and octave. The second line should indicate the degree of intonation and the frequency used for A_4. The third line should indicate which species, if any, can hear the frequency. The strings printed by your function should match the examples shown below.

Arguments: 554.5 (default A4)

```
554.5 Hz is C♯5.
It is in tune if A4=440.0 Hz.
It is within the hearing range of a human.
```

Arguments: 49,000.0, A4=432

```
49000.0 Hz is G11.
It is 9 cents flat if A4=432.0 Hz.
It is within the hearing range of a cat.
```

Arguments: 0.25 (default A4)

```
0.25 Hz is C-6.
It is 38 cents flat if A4=440.0 Hz.
I don't know of a species that can hear this frequency.
```

Other instructions

- Please write docstrings for each of your functions. Your docstrings should start with a brief statement of the function's purpose. Include an "Args" section to document any arguments, and a "Returns" section to describe the return value (if any). Your `main()` function will not need a "Returns" section, but it should include a "Raises" section and a "Side effects" section (because printing is a side effect). Docstrings were covered in the lectures here: https://youtu.be/HTfv83IPlQY?t=1415. There's an ELMS page about them here: https://umd.instructure.com/courses/1299872/pages/docstrings.
- Please keep your lines of code to 80 characters or less. If you need help breaking up long lines of code, please see https://umd.instructure.com/courses/1299872/pages/how-to-break-up-long-lines-of-code.

Testing your code

Running your program at the command line

The template is designed to use command-line arguments. To run your program within the VS Code built-in terminal, first make sure you have opened (in VS Code) the directory where your program is saved. If necessary, you can go to the VS Code File menu and select "Open..." on macOS or "Open Folder..." on Windows, and navigate to the directory where your program is.

Then, open the VS Code built-in terminal. Type `python3` (on macOS) or `python` (on Windows) followed by a space, the name of the program, another space, and the frequency you wish to analyze. If you want to specify a different value for A_4 than the default value, add another space, `-a4`, another space, and the value you wish to use for A_4. Below are some examples.

```
python3 frequency.py 780.5
python3 frequency.py 18 -a4 432
```

Test script

A test script (`test_frequency.py`) is provided to help you test your code. In order to use the test suite, you will need to install Pytest. To do this, type the following at the command line (Windows users, type `python` instead of `python3`):

```
python3 -m pip install --upgrade pip
python3 -m pip install --upgrade pytest
```

You will need to save the test script in the same directory as your program. If your program is not named `frequency.py`, you will need to edit line 4 of the test script (see instructions in the test script).

To run the test script within the VS Code built-in terminal, make sure you have opened the directory where your file is saved. Then open the terminal and type the following at the command line:

```
pytest test_frequency.py
```

If that does not work, try

```
python3 -m pytest test_frequency.py
```

If your program passes all tests, you will see output like this:

```
================ test session starts ================
platform linux -- Python 3.8.5, pytest-6.0.1, py-1.9.0, pluggy-0.13.1
rootdir: /home/aric/Documents/INST326/
plugins: anyio-2.0.2
collected 5 items

test_frequency.py .....                          [100%]

================= 5 passed in 0.04s =================
```

Academic integrity

This assignment is to be done by you individually, without outside help of any kind (including, but not limited to, help from classmates, tutors, or the internet). Disseminating these instructions in whole or in part without written permission of the instructor is considered an infraction of academic integrity.

---

1. https://en.wikipedia.org/wiki/Octave
2. The notes with ♯ have equivalent names that use the ♭ ("flat") symbol: A♯=B♭, C♯=D♭, D♯=E♭, F♯=G♭, and G♯=A♭. In this assignment, we will use sharps exclusively.
3. This and all other hearing range data were drawn from https://en.wikipedia.org/wiki/Hearing_range, with the exception of greater wax moth hearing range, from https://www.nature.com/news/moth-smashes-ultrasound-hearing-records-1.12561, and pigeon hearing range

data, from https://en.wikipedia.org/wiki/Perception_of_infrasound#Species

Last updated 2021-02-03 04:16:46 -0500