Tutorial Exercises Week 3 - Solutions

Question 1

Write a function called `is_even()` which takes an integer as an input and returns `True` if the number is even and `False` if the number is odd.

```
def is_even(x):
    return x % 2 == 0
[(i, is_even(i)) for i in range(10)]
```

```
[(0, True),
 (1, False),
 (2, True),
 (3, False),
 (4, True),
 (5, False),
 (6, True),
 (7, False),
 (8, True),
 (9, False)]
```

Question 2

Write a function called `is_prime()` which takes an integer as an input and returns `True` if the number is prime (a natural number greater than one that is only divisible by itself and one) and `False` if the number is not prime.

```
def is_prime(x):
    if x in [0, 1]:
        return False
    for i in range(2, x):
        if x % i == 0:
            return False
    return True
[(i, is_prime(i)) for i in range(10)]
```

```
[(0, False),
 (1, False),
 (2, True),
```

```
 (3, True),
 (4, False),
 (5, True),
 (6, False),
 (7, True),
 (8, False),
 (9, False)]
```

The function could be made more efficient by having the loop only go as far as $\lfloor \sqrt{x} \rfloor$ (the square root of the input rounded down). If no factor was found by that point, none will be found after it either.

Question 3

At the university grades are between 1 and 10. Typically we take the percentage of answers correct in an exam, divide by 10 and round to the nearest half. So for example a grade of 72.4% is rounded down to 7.0 whereas a 72.5% is rounded up to 7.5. However, there are some exceptions to this:

- A 5.5 is not allowed. 54.9% would be rounded down to a 5. 55.0% would be rounded up to a 6.0.
- A 0.0 or a 0.5 is not allowed. Any grade below 10% is rounded up to a 1.0.

Create a Python function which takes the percentage grade as an input and gives the rounded grade as output, treating the special cases appropriately. Test out this function on different grades.

You may find the `round()` function useful for this.

We give an extra `1e-6` to avoid grading down due to floating-point precision:

```
def roundedGrade(x):
    if (x < 10):
        return 1.0
    elif (x >= 55 - 1e-6 and x < 60):
        return 6.0
    elif (x >= 50 and x < 55 - 1e-6):
        return 5.0
    else:
        return round((x + 1e-6) / 5, ndigits=0) / 2
[(i, roundedGrade(i)) for i in [40, 54.9, 55, 57, 62.49, 62.5, 77.49, 77.5]]
```

```
[(40, 4.0),
 (54.9, 5.0),
 (55, 6.0),
 (57, 6.0),
 (62.49, 6.0),
 (62.5, 6.5),
```

```
 (77.49, 7.5),
 (77.5, 8.0)]
```

Question 4

Runners typically measure their "pace" in terms of how many minutes and seconds it takes to run one kilometer. For example, the marathon world record was run at a 2:51 pace, meaning on average it took 2 minutes and 51 seconds to run each of the 42.195 kilometers in the marathon. This translates to 21.05km/h. Write a Python function that takes pace as an input (in 2 arguments: minutes and seconds) and returns the speed in kilometers per hour.

```
def pace2speed(mins, secs):
    mins = mins + secs/60
    return 60 / mins
[pace2speed(x[0], x[1]) for x in [(5, 00), (4, 00), (2, 51)]]
```

```
[12.0, 15.0, 21.052631578947366]
```

Question 5

Write a function that does the inverse of what is asked in Question 2. The function should take as input a speed in km/h and return the pace as a tuple of two elements (with minutes and seconds) as output.

```
def speed2pace(speed):
    mins = 60 / speed
    return (int(mins // 1), mins % 1 * 60)
[speed2pace(x) for x in [12, 13, 14, 15]]
```

```
[(5, 0.0), (4, 36.9230769230769), (4, 17.142857142857135), (4, 0.0)]
```

Question 6

The dot product of two equal-length vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ is given by:

$$x \cdot y = \sum_{i=1}^{n} x_i y_i$$

Calculate the dot product of the two lists of numbers below in two ways:

1. Using only Python's built-in functions (such as with a `for` loop or list comprehensions).
2. Using `numpy` with the `np.dot()` function.

```
x = [2, 6, 4, 9, 10]
y = [11, 5, 3, 9, 2]
```

```
# For loop approach:
z = 0
for i in range(len(x)):
  z += x[i] * y[i]
z
```

```
165
```

```
# List comprehension approach:
sum([i * j for i, j in zip(x, y)])
```

```
165
```

```
import numpy as np
np.dot(x, y)
```

```
165
```

Question 7

Create a function in Python that evaluates:

$$f(x) = 1 + x + x^2 + x^3$$

Then use the *bisection method* to find the point where $f(x)$ crosses zero. That is, find the value of x that solves $f(x) = 0$. Let $\epsilon_{tol} = 0.000001$ be the prespecified tolerance level. Follow these steps in your code:

1. Start with a guess a where $f(a) < 0$ and a guess b where $f(b) > 0$.
2. Get the midpoint between a and b with $c = \frac{a+b}{2}$.
3. Two possibilities:

- If $|f(c)| \leq \epsilon_{tol}$, iteration is complete. The value c (approximately) solves $f(c) = 0$.
- If $|f(c)| > \epsilon_{tol}$, then set $a = c$ if $f(c) < 0$ and otherwise set $b = c$. Return to step 2.

Hints: For our function $f(x)$, $a = -10$ and $b = 10$ will work as initial guesses. To do the iteration you should use a `while` loop. The structure of the code will be similar to the code we used to approximate the square root approximate the square root in the lectures.

```
def f(x):
    return 1 + x + x ** 2 + x ** 3

a = -10
b = 10
tol = 0.000001
dist = tol + 1
```

```
while dist > tol:
    c = (a + b) / 2
    dist = abs(f(c))
    if f(c) < 0:
        a = c
    else:
        b = c
c
```

```
-1.000000238418579
```