The following exercises are related to the Python programming language [1].

1. Write a function `sumultiply` that takes two integer arguments and returns their product. The function should not use the $*$ or / operators. For example:

```
> sumultiply(11, 13)
143
> sumultiply(5, 123)
615
```

2. Write a function `ispalindrome` that takes a string and returns `True` if the string is a palindrome and `False` otherwise. For example:

```
> ispalindrome("radar")
True
> ispalindrome("radars")
False
```

3. Write a function `simpleinterest` that, for a loan with simple interest, takes a principal amount p, an interest rate r, and a number of periods, and returns the total amount repaid.

```
> simpleinterest(1000, 3, 5)
1150.0
> simpleinterest(1000, 7, 10)
1700.0
```

4. Write a function `compoundinterest` that, for a loan with compound interest, takes a principal amount p, an interest rate r, and a number of periods, and returns the total amount repaid.

```
> compoundinterest(1000, 5, 10)
1628.89
> compoundinterest(1000, 7, 5)
1967.15
```

5. Write a function `newtonsroot` that takes a number x and returns its square root correct to six decimal places as calculated by Newton's method. Newton's method is to make an initial (random) guess r_0 at the square root, and to repeatedly improve it as follows:

$$r_{i+1} = r_i - \frac{r_i^2 - x}{2r_i}$$

For example:

```
> newtonsroot(100)
10.0
> newtonsroot(144)
12.0
```

6. Write a function `pitondecs` that takes an integer n and returns π correct to n decimal places. For example:

```
> pitondecs(2)
3.14
> pitondecs(6)
3.141593
```

7. Write a function `etondecs` that takes an integer n and returns ϵ correct to n decimal places. For example:

```
> etondecs(2)
2.72
> etondecs(6)
2.718282
```

8. Write a function `caesar` that takes a string and an integer n and returns the string with each letter shifted n places in the alphabet. For example:

```
> caesar('abcd', 3)
'defg'
> caesar('Hello, world!', 2)
'Jgnnq, yqtnf!'
```

9. Write a function `sortlist` that takes a list of integers and returns a copy of it sorted. Note that Python has a built-in sort function, but try to solve this problem without using it. For example:

```
> sortlist([3,1,2])
[1,2,3]
> sortlist([10,-9,5,-1,0])
[-9,-1,0,5,10]
```

10. Write a function `countstr` that takes string and returns, for each character in the string, the number of times the character is contained in it. You might use a dictionary for this purpose. For example:

```
> countstr('aaacbb')
{'a': 3,'c': 1,'b': 2}
> countstr('Hello, world!')
{'H': 1,'e': 1,'l': 3,'o': 2,',': 1,' ': 1,'w': 1,'r': 1,'d': 1,'!': 1}
```

References

[1] Python Software Foundation. Welcome to python.org.