SELECTED SOLUTIONS

CHAPTER 1

Exercise Set 1.1.1

1. **for** n **in range**(11,21):
2. **for** n **in range**(3,28,4):
3. **for** n **in range**(1,16,2):
4. **for** n **in range**(5,0,-1):
5. **for** n **in range**(100, 9, -10):

Exercise Set 1.1.2

1. $a_1 = 11, d = 1$
2. $a_1 = 3, d = 4$
3. $a_1 = 1, d = 2$
4. $a_1 = 5, d = -1$
5. $a_1 = 100, d = -10$

Exercise Set 1.1.3

1. **print**(3*n)
2. **print**(n+1)
3. **print**(2*n+1)
4. **print**(-1*n)
5. **print(**2**n)
6. **print**(n**2)

7. #1, 2, 3, 4 are arithmetic

Exercise Set 1.1.4

1. $\{a_n\} = \{3n\}$

2. $\{a_n\} = \{n+1\}$

3. $\{a_n\} = \{2n+1\}$

4. $\{a_n\} = \{-n\}$ or $\{-1 \cdot n\}$

5. $\{a_n\} = \{2^n\}$

6. $\{a_n\} = \{n^2\}$

Exercise Set 1.1.5

1. **print(Fraction**(n+1,n))

2. **print(Fraction**(n+1,n+3))

3. **print(Fraction**(1,2*n))

4. **print(Fraction**(1,2**n))

5. **print(Fraction**(1,3**(n-1)))

6. No, 1/3 and 1/5 are represented by some very odd fractions.

Exercise Set 1.1.6

1. **print**(4*n+n**3)

2. Modify the program in #1 so that it asks the user to input a value of n, and then print the product for just that subscripted value.

```
n = int(input("n? "))
print(4*n+n**3)
```

Exercise Set 1.2.1

1. $a_n = 17 + (n - 1)13 = 4 + 13n$

   17, 30, 43, 56, 69

2. $a_n = 200 + (n - 1)(-10) = 210 - 10n$

   190, 180, 170, 160, 150

3. $a_n = -25 + (n - 1)(5) = -30 + 5n$

   -25, -20, -25, -10, -5

Exercise Set 1.2.2

1. (a) $a_n = -9 + 4n$ (b) -5, -1, 3, 7, 11 (c) $a_{100} = 391$

2. (a) $a_n = 32 - 6n$ (b) 26, 20, 14, 8, 2 (c) $a_{100} = -568$

3. (a) $a_n = -16 + 3n$ (b) -13, -10, -7, -4, -1 (c) $a_{100} = 284$

4. (a) $a_n = -59 + 6n$ (b) -53, -47, -41, -35, -29 (c) $a_{100} = 541$

Exercise Set 1.2.3

1. The effect of \n in a print statement is that it skips a line of output.

2. Modify Program 1.2.4 so that a third input n is requested, and the program outputs a_n using the formula for general term.

```
a1 = int(input("a1? "))
d = int(input("d? "))
n = int(input("n? "))
an = a1+(n-1)*d
print("a",n,"=",an)
```

3. Modify Program 1.2.4 so that instead of printing only the nth term, it outputs the first n terms.

```
a1 = int(input('a1? '))
d = int(input('d? '))
```

```
n = int(input('n? '))

print("The first",n,"terms are")
for n in range(1,n+1):
   an = a1+(n-1)*d
   print(an)
```

4. Write a program that will ask the user to input any two terms in the sequence. Calculate and output the values of a_1 and d.

In this program, $a_j = b$ and $a_k = c$.

```
b = int(input("1st term? "))
j = int(input("subscript of 1st term? "))
c = int(input("2nd term? "))
k = int(input("subscript of 2nd term? "))
a1 = b-d*(j-1)
print("a1 is",a1)
d = (c-b)/(k-j)
print("d is",d)
```

Exercise Set 1.2.4

1. 11, 20, 29, 38, 47

2. 100, 20, 4, $\frac{4}{5}$, $\frac{4}{25}$

3. 2, 4, 15, 256, 65536

4. 1, 1, 2, 3, 5

5. $a_1 = 7$, $a_n = a_{n-1} + 2$

6. $a_1 = -20$, $a_n = a_{n-1} - 20$

7. $a_1 = 14$, $a_n = a_{n-1} - 6$

Exercise Set 1.2.5

1. Modify Program 1.2.5 so that instead of assigning 10 to a and 5 to d, the user is prompted to enter values for a and d. Then output only the first 5 terms, not the first 9 terms.

```
a = int(input("a? "))
```

```
d = int(input("d? "))
print("The first 5 terms are:")
print(a)
for n in range(1,5):
   a = a + d
   print(a)
```

2. Modify the program in exercise 1 so that it also asks the user to input how many terms they want (rather than 5).

```
a = int(input("a? "))
d = int(input("d? "))
n = int(input("How many terms? "))
print("The first",n,"terms are:")
print(a)
for k in range(1,n):
   a = a + d
   print(a)
```

Exercise Set 1.3.1

1. Modify Program 1.3.1 so that it makes an additional input request for how many terms are desired.

```
a1 = int(input("a1? "))
r =  int(input("r? "))
n = int(input("How many terms? "))
print("The 1st",n,"terms are:")
for k in range(1,n+1):
   print(a1*r**(k-1))
```

Exercise Set 1.3.2

1. $a_1 = 1, r = 10, a_n = 10^{n-1}$

2. $a_1 = \frac{1}{10}, r = \frac{1}{10}, a_n = \frac{1}{10}\left(\frac{1}{10}\right)^{n-1}$ or $a_n = \left(\frac{1}{10}\right)^n$

3. $a_1 = 2, r = 3, a_n = 2(3)^{n-1}$

4. $a_1 = 1, r = -1, a_n = (-1)^{n-1}$

5. $a_1 = 1, r = \frac{1}{2}$

$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}$

6. $a_1 = 4, r = 3$

12, 36, 108, 324, 972

7. $a_1 = \frac{1}{3}, r = 3$

$\frac{1}{3}$, 1, 3, 9, 27

8. $a_1 = 1$ so $a_n = 3^{n-1}$

9. $a_1 = \frac{7}{15}, r = 15$ so $a_n = \frac{7}{15}(15)^{n-1}$

Exercise Set 1.3.3

1. $a_1 = 10, r = 3, a_{n+1} = 3a_n$

2. $a_1 = 1, r = ¾, a_{n+1} = \frac{3}{4}a_n$

3. $a_1 = \frac{1}{3}, r = \frac{2}{3}, a_{n+1} = \frac{2}{3}a_n$

4. $a_1 = 1, r = -2, a_{n+1} = -2a_n$

Exercise Set 1.3.4

1. Modify the program you wrote in Exercise Set 1.2.5 #2 (which generated an arithmetic sequence using the recursive definition) so that it generates a geometric sequence using the recursive definition.

```
a = int(input("a? "))
r = int(input("r? "))
n = int(input("How many terms? "))
print("The first",n,"terms are:")
print(a)
for k in range(1,n):
   a = a*r
   print(a)
```

Exercise Set 1.3.5

1. Modify the program you wrote in Exercise Set 1.3.4 #1 so that it accepts fraction inputs.

```
from fractions import Fraction
a = Fraction(input("a? "))
r = Fraction(input("r? "))
n = int(input("How many terms? "))
print("The first",n,"terms are:")
print(a)
for k in range(1,n):
   a = a*r
   print(a)
```

Integers and decimals can be input for a and r, however the output will be simplified fractions.

Exercise Set 1.4.1

1. First 10 Factorial numbers:

1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800

2. Program to calculate and output first 10 Factorial numbers:

```
f = 1
print(f)
for n in range(2,11):
   f = f*n
   print(f)
```

3. First 10 Fibonacci numbers:

1	1
2	1
3	2
4	3
5	5
6	8
7	13
8	21
9	34
10	55

Exercise Set 1.4.2

1. Program to generate first 10 terms of the sequence with general term $a_n = \sqrt{n}$.

```
from math import *
for n in range(1,11):
   print("\u221A",n)
```

2. Program to calculate and output first 10 Fibonacci numbers using the general term:

```
from math import *
for n in range(1,11):
   print(round(((1+ sqrt(5))**n-(1- sqrt(5))**n)/((2**n)*( sqrt(5)))))
```

The first 10 terms are an exact match of the actual sequence!

3. Write a program to approximate √ p using the recursively defined Babylonian sequence.

```
from math import *
p = int(input("p? "))
k = float(input("initial guess of square root of p? "))
a = k
for n in range(1,11):
   a = (1/2)*(a+p/a)
   print(a)
print("Actual square root of",p,"is",sqrt(p))
```

4. Write a program that calculates at least the first 10 terms of the Golden Ratio.

One way to do it is to modify Program 1.4.1.

```
a = 1
```

```
b = 1
print(b/a)
for n in range(1,11):
   c = a + b
   print(c/b)
   a = b
   b = c
```

The ratios appear to be approaching the constant 1.618.

Exercise Set 1.5.1

1. The program gives an error message that variable 's' is not defined.

2. same result

3. 5050

4. 105

5. 120

6. Modify Program 1.5.1 so that it outputs a "running total" each time it traverses the loop.

```
a1 = int(input("a1? "))
d = int(input("d? "))
n = int(input("How many terms? "))
s = 0
for i in range(1,n+1):
   t = a1 + (i-1)*d
   s += t
   print(s)
print("Sum of 1st",n,"terms is",s)
```

7. Modify Program 1.5.1 so that it calculates the sum of a geometric sequence instead of an arithmetic sequence.

```
a1 = int(input("a1? "))
r = int(input("r? "))
n = int(input("How many terms? "))
s = 0
for i in range(1,n+1):
   t = a1*r**(i-1)
```

```
    s += t

print("Sum of 1st",n,"terms is",s)
```

The sum of the first 8 terms of $3(2)^{n-1}$ is 765.

8. Modify the program you wrote in Exercise 7 so that it handles fractions.

```
from fractions import Fraction
a1 = Fraction(input("a1? "))
r = Fraction(input("r? "))
n = int(input("How many terms? "))
s = 0
for i in range(1,n+1):
    t = a1*r**(i-1)
    s += t

print("Sum of 1st",n,"terms is",s)
```

The sum of the first 6 terms is $\frac{63}{32}$.

Exercise Set 1.5.2

1. $\sum_{k=1}^{7}(k+1)^2 = 2^2 + 3^2 + 4^2 + \cdots + 8^2 = 203$
2. $\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \cdots + \frac{13}{14} = \sum_{k=1}^{13} \frac{k}{k+1}$
3. $1 + \sqrt{2} + \sqrt{3} + 2 + \sqrt{5} + \sqrt{6} + \sqrt{7} + \sqrt{8} + 3 = \sum_{k=1}^{9} \sqrt{k}$
4. $\frac{2}{3} + \frac{4}{9} + \frac{8}{27} + \frac{16}{81} + \frac{32}{243} + \frac{64}{729} = \sum_{k=1}^{6} \left(\frac{2}{3}\right)^k$

Exercise Set 1.5.3

1. $a_{46} = -133$, $S_{46} = -3013$

2. $a_{90} = -177$, $S_{90} = -7920$

3. $a_{98} = 558$, $S_{98} = 30919$

4. $a_{52} = -299$, $S_{52} = 16406$

5. Sum of first n terms of arithmetic sequence with first term a1 and common difference d:

```
a1 = int(input("a1? "))
```

```
d = int(input(“d? “))
n = int(input(“How many terms? “))
an = a1+(n-1)*d                    # This is the last term
s = (n/2)*(a1 + an)
print(“Sum of 1st”,n,”terms is”,s)
```

Exercise Set 1.5.4

1. Modify Program 1.5.2 so that it handles sequences with fractions.

```
from fractions import Fraction
a1=Fraction(input("a1? "))
d=Fraction(input("d? "))
an=Fraction(input("an? "))
n = (an-a1 + d)/d
s= (n/2)*(a1 + an)
print("Sum of 1st",int(n),"terms is",s)
```

2. n = 24 terms

3. S_{30} = 1185 seats

4. n = 13 years

5. S_{24} = 3258

Exercise Set 1.5.5

1. Modify Program 1.5.1 so that it calculates sum of geometric rather than arithmetic sequence.

```
a1 = int(input("a1? "))
r = int(input("r? "))
n = int(input("How many terms? "))
s = 0
for i in range(1,n+1):
   t = a1*r**(i-1)
   s = s + t
print("Sum of 1st",n,"terms is",s)
```

2. -2046

3. $\frac{364}{3}$

4. 4.98186 (rounded to 5 decimal places)

Exercise Set 1.5.6

1. -2046; $\frac{364}{3}$; 4.98186 (rounded to 5 decimal places)

2. Modify 1.5.1 Enhanced (solution to Exercise Set 1.5.2 Exercise 5) so that it calculates the sum of a geometric rather than arithmetic sequence.

```
from fractions import Fraction
a1 = Fraction(input("a1? "))
r = Fraction(input("r? "))
n = int(input("How many terms? "))
s = a1*(1-r**n)/(1-r)
print("sum of 1st",n,"terms is",s)
```

3. -1023

Exercise Set 1.5.7

1. Geometric sequence with $a_1 = 42000$ and $r = 1.03$. At the beginning of the fifth year your salary will be $a_5 = 42000(1.03)^4 = \$47{,}271.37$.

2. Geometric sequence with a1 = 15000 and r = .85. After five years, the equipment is worth $a_6 = 15000(.85)^5 = \$6{,}655.58$.

3. (a) $a_{10} = 2(.9)^9 \approx .775$ ft.

(b) The 8th swing is the first time the length is less than 1 ft. ($a_8 \approx .957$ ft)

(c) $S15 = 2\frac{1-.09^{15}}{1-.9} \approx 15.882$ ft.

(d) Theoretically, it never stops. From a physics perspective, we'll consider it stopped after 74 swings ($a_{74} \approx .0009$). $S_{74} = 2\frac{1-.09^{74}}{1-.9} \approx 19.992$ ft. Consider this about 20 ft.

4. Geometric sequence with $a_1 = 1$ and $r = 2$.

(a) # of grains in 64th square is $a_{64} = 2^{63} \approx 9.223 \times 10^{18}$ grains of rice.

(b) Total # of grains of rice on chessboard is $S_{64} = \frac{1-2^{64}}{1-2} \approx 1.845 \times 10^{19}$

Reasonable is a subjective term. The commoner's request could have been considered reasonable but it was not *feasible*.

5. Option 1 is one million dollars. Option 2 is a geometric series with $a_1 = .01$ and $r = 2$. The sum of all 30 days would be $S_{30} = .01\frac{(1-2^{30})}{(1-2)} = \$10{,}737{,}418.23$. That's over ten billion dollars. I would choose Option 2.

6. Sum of an arithmetic sequence with 8 terms, $a_1 = 100$, $a_8 = 800$. You would lose a total of 3600 points.

Exercise Set 1.5.8

1. (a) $a_1 = 1$, $r = ½$ (b) It converges because $|r| < 1$. (c) $S = \frac{1/2}{1-1/2} = 1$

2. (a) $a_1 = 8$, $r = 1.5$ (b) It does not converge because $|r| > 1$.

3. (a) $a_1 = 8$, $r = \frac{1}{3}$ (b) It converges because $|r| < 1$. (c) $S = \frac{8}{1-1/3} = 12$.

4. (a) $a_1 = 4$, $r = -½$ (b) It converges because $|r| < 1$. (c) $S = \frac{4}{1-(-1/2)} = \frac{8}{3}$.

Exercise Set 1.5.9

1. $a_1 = .9$, $r = .1$; $S = \frac{.9}{1-.1} = 1$

2. $a_1 = .8$, $r = .1$; $S = \frac{.8}{1-.1} = \frac{8}{9}$

3. 'round down'

4. 'one third does not equal 0.33'

5. Write a program that asks the user to input a nonzero number. Determine and output whether the number is positive or negative.

```
a = float(input("a? "))
if a>0:
   print("positive")
else:
   print("negative")
```

6. Write a program that asks the user to input their age. If they are 18 or older, output "You can drink legally." Otherwise, output "You cannot drink legally."

```
age = float(input("age? "))
if age>=18:
   print("you can drink legally")
else:
   print("you cannot drink legally")
```

Exercise Set 1.5.10

1.A. '10% off is a better deal'
B. '$10 off is a better deal'
C. a price of $100 will result in '10% off is the same as $10 off'

2. Write a program that asks the user to input the weight of their first class letter, in oz. Calculate and output the cost of postage using the 2020 postage table.

```
oz = float(input("weight in oz? "))
if oz <= 1:
   print("0.55")
elif oz <= 2:
   print("0.70")
elif oz <= 3:
   print("0.85")
else:
   print("I don't know how much it is for more than 3 oz")
```

3. Write a program that asks the user to input the total point value of three playing cards – a number between 3 (highly unlikely) and 30. Output the appropriate message based on the point value.

```
sum = int(input("sum of 3 cards? "))
if sum <=16:
   print("hit")
elif sum <= 21:
   print("stay")
else:
   print("bust")
```

Exercise Set 1.6.1

1. $\frac{1}{2}, -2, \frac{9}{2}, -8, \frac{25}{2}$

2. $-\frac{1}{2}, \frac{1}{4}, -\frac{1}{6}, \frac{1}{8}, -\frac{1}{10}$

3. $1, -\sqrt{2}, \sqrt{3}, -2, \sqrt{5}$

4. $a_n = (-1)^{n+1}(3n - 1)$

5. $a_n = (-1)^{n+1}\left(\frac{2}{3}\right)^n$

6. $a_n = (-1)^n\left(\frac{1}{2n}\right)$

7. $a_n = (-1)^{n+1}$

8. Write a program that outputs the first 10 terms of $-\frac{1}{2}, \frac{1}{4}, -\frac{1}{6}, \frac{1}{8}, -\frac{1}{10}$

```
from fractions import Fraction
for n in range(1,11):
    a = ((-1)**n)*Fraction(1, 2*n)
    print(a)
```