Contents

Part I

Basics

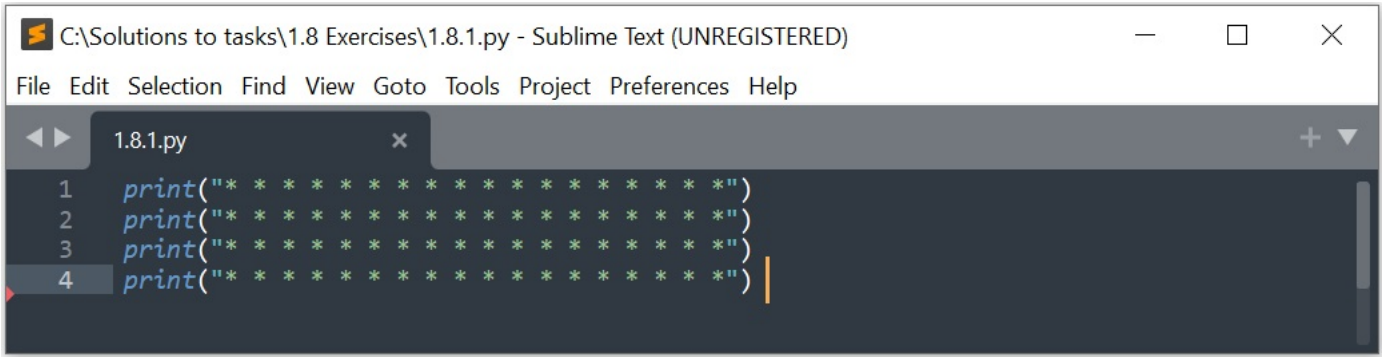
Chapter 1

Getting Started

1.1 Exercises 1.8

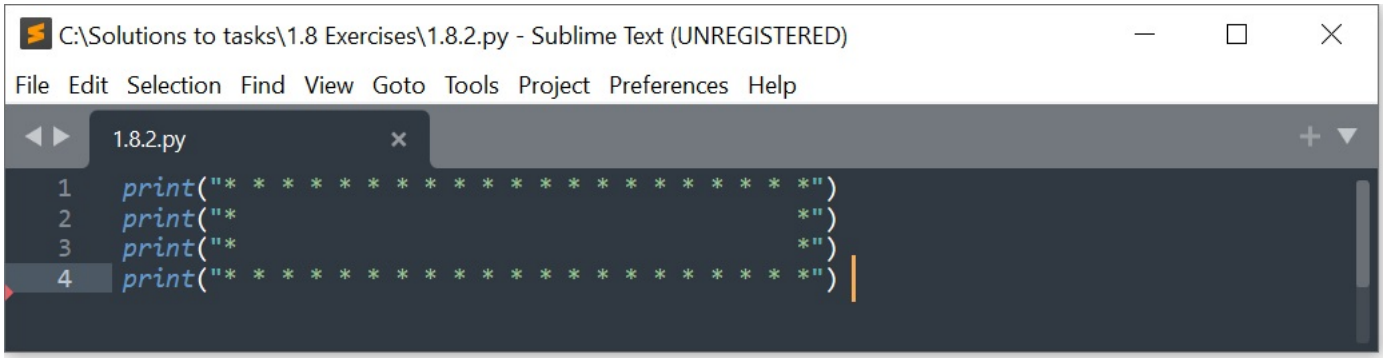
1.Print a box like the one below

```
**********
**********
**********
**********
```



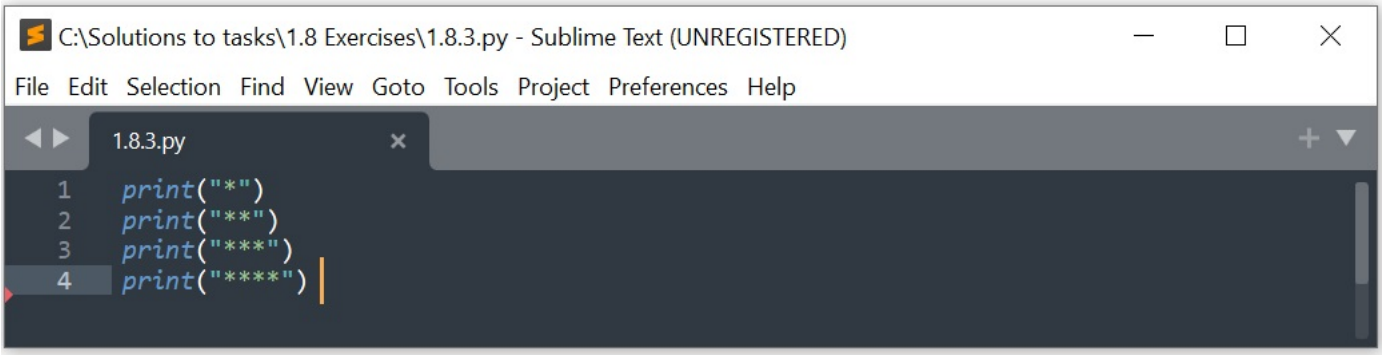
2.Print a box like the one below.

```
**********
*        *
*        *
**********
```

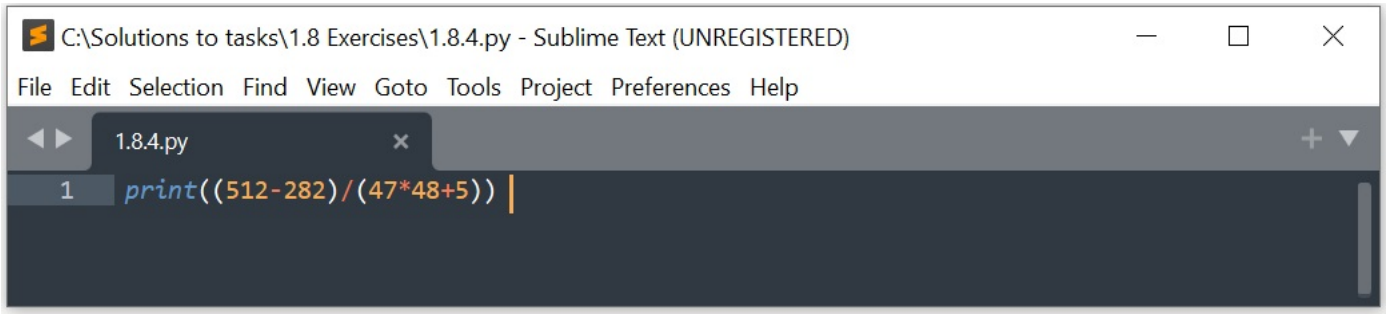


3.Print a triangle like the one below.

```
*
**
***
****
```

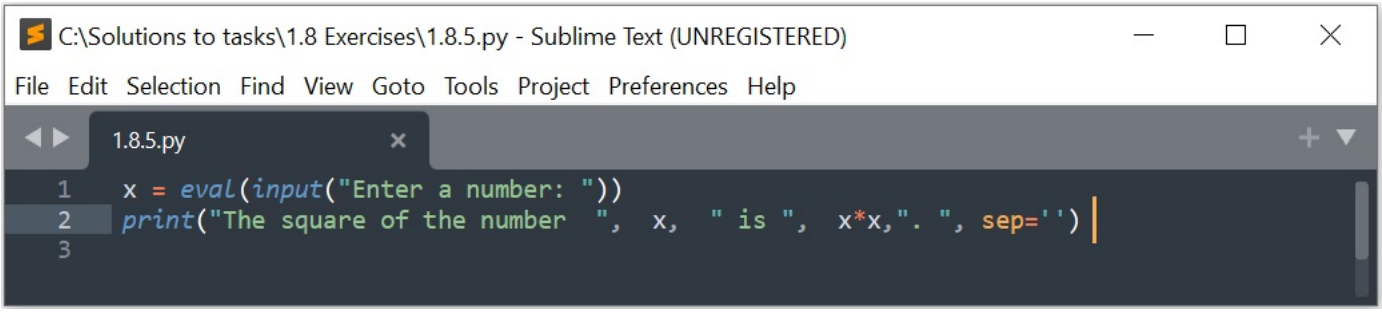


4. Write a program that computes and prints the result of $\frac{512-282}{47*48+5}$. It is roughly .1017.



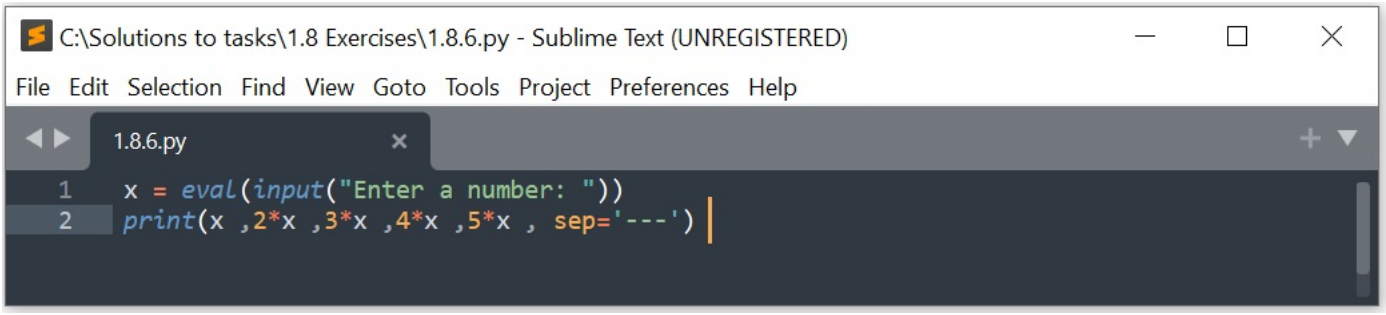
5.Ask the user to enter a number. Print out the square of the number, but use the *sep* optional argument to print it out in a full sentence that ends in a period. Sample output is shown below.

```
Enter a number: 5
The square of 5 is 25.
```

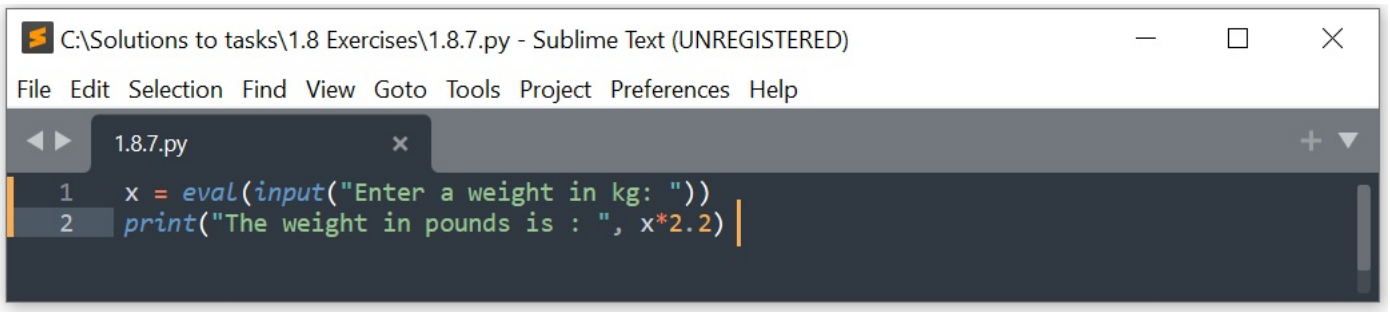


6. Ask the user to enter a number x. Use the *sep* optional argument to print out x, 2x, 3x, 4x, and 5x, each separated by three dashes, like below.

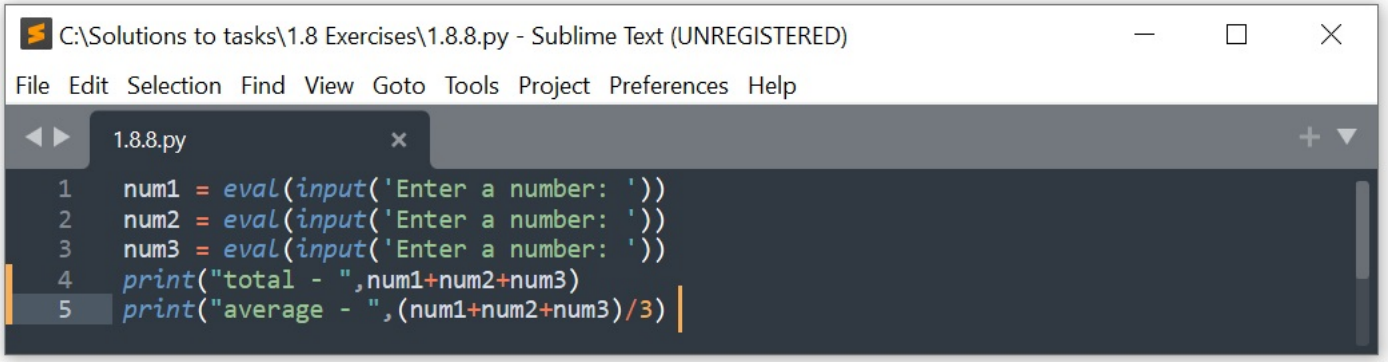
```
Enter a number: 7
7– – –14– – –21– – –28– – –35
```



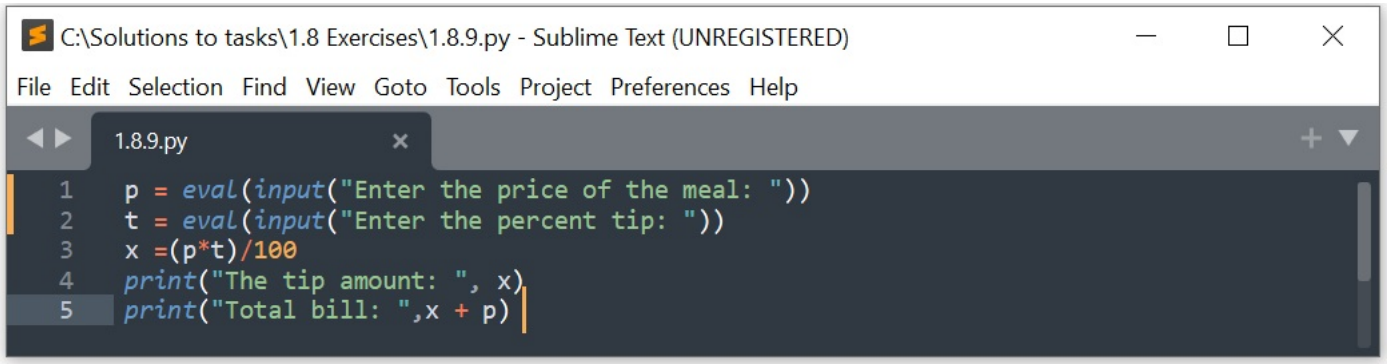
7. Write a program that asks the user for a weight in kilograms and converts it to pounds. There are 2.2 pounds in a kilogram.



8.Write a program that asks the user to enter three numbers (use three separate **input** statements). Create variables called *total* and *average* that hold the sum and average of the three numbers and print out the values of *total* and *average*.



9.A lot of cell phones have tip calculators. Write one. Ask the user for the price of the meal and the percent tip they want to leave. Then print both the tip amount and the total bill with the tip included.

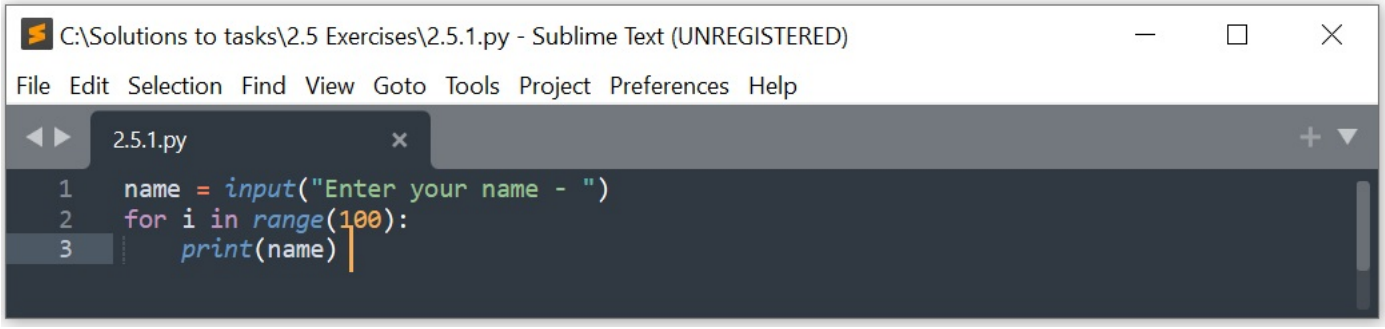


Chapter 2

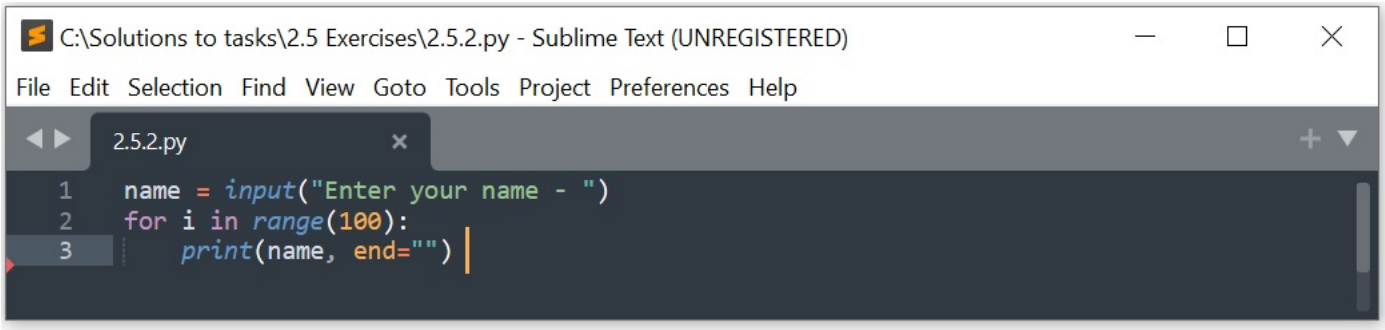
For loops

2.1 Exercises 2.5

1.Write a program that prints your name 100 times.

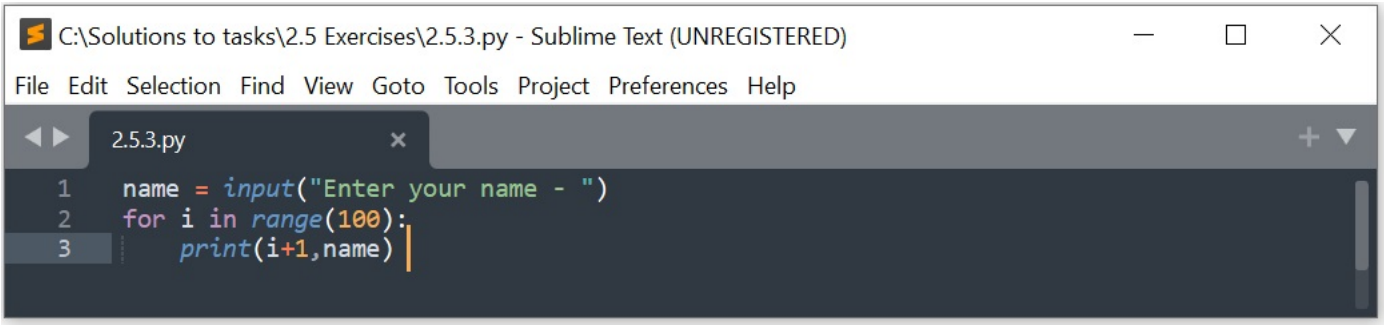


2.Write a program to fill the screen horizontally and vertically with your name. [Hint: add the option $end = ''$ into the **print** function to fill the screen horizontally.]



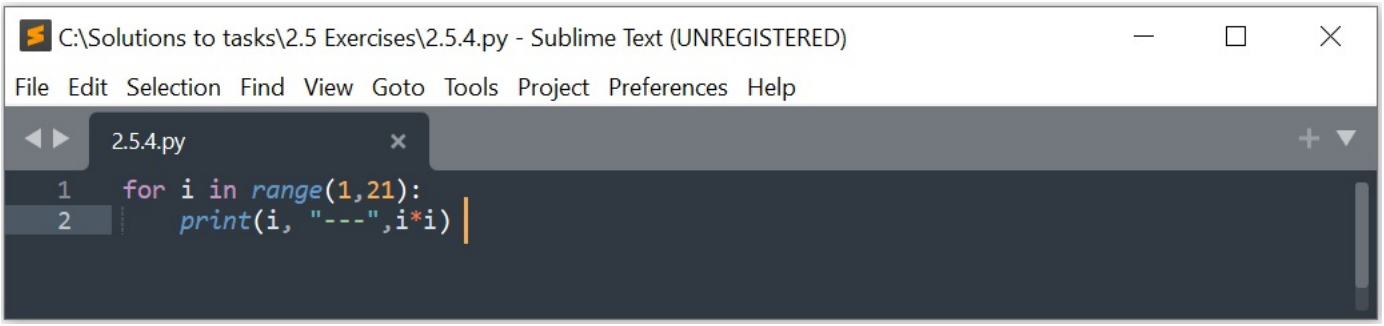
3.Write a program that outputs 100 lines, numbered 1 to 100, each with your name on it. The output should look like the output below.

```
1 Your name
2 Your name
3 Your name
4 Your name
. . .
100 Your name
```

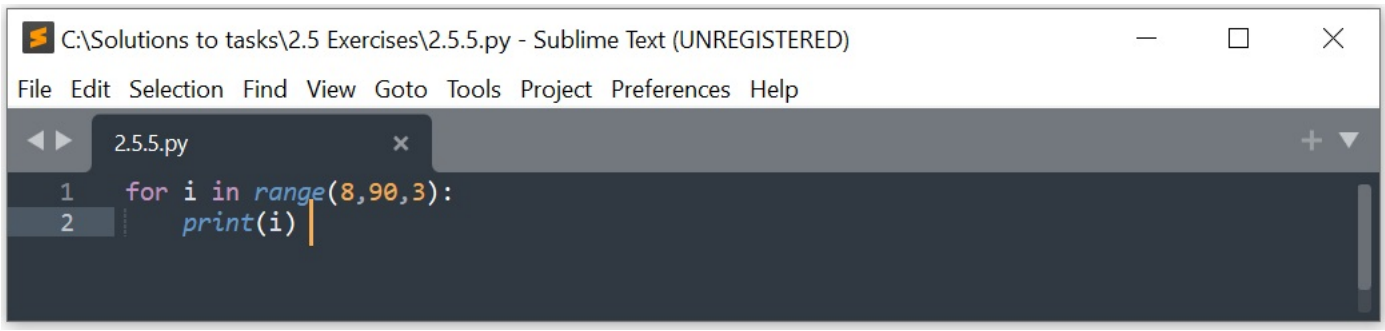


4.Write a program that prints out a list of the integers from 1 to 20 and their squares. The output should look like this:

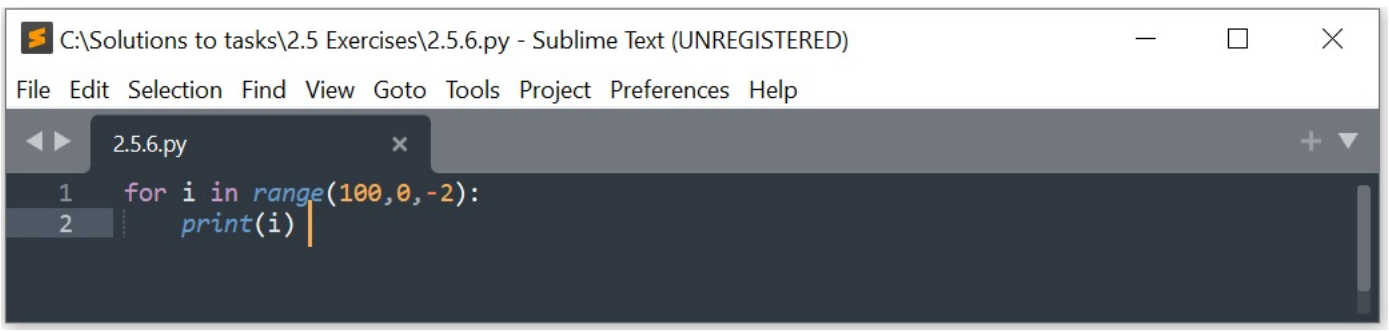
1 – – – 1
2 – – – 4
3 – – – 9
. . .
20 – – – 400



5. Write a program that uses a **for**, loop to print the numbers 8, 11, 14, 17, 20, . . . , 83, 86, 89.



6. Write a program that uses a **for**, loop to print the numbers 100, 98, 96, . . . , 4, 2.



7. Write a program that uses exactly four **for** loops to print the sequence of letters below

AAAAAAAAAABBBBBBBCDCDCDCDEFFFFFFG

C:\Solutions to tasks\2.5 Exercises\2.5.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

2.5.7.py

```
1  for i in range(10):
2      print("A", end="")
3  for i in range(7):
4      print("B", end="")
5  for i in range(4):
6      print("C", end="")
7      print("D", end="")
8  print("E", end="")
9  for i in range(6):
10     print("F", end="")
11 print("G", end="")
```

8.Write a program that asks the user for their name and how many times to print it. The program should print out the user's name the specified number of times.

C:\Solutions to tasks\2.5 Exercises\2.5.8.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

2.5.8.py

```
1  name = input("Enter your name - ")
2  x = eval(input("How many times: "))
3  for i in range(x):
4      print(name)
```

9.The Fibonacci numbers are the sequence below, where the first two numbers are 1, and each number thereafter is the sum of the two preceding numbers. Write a program that asks the user how many Fibonacci numbers to print and then prints that many.

1,1,2,3,5,8,13,21,34,55,89...

C:\Solutions to tasks\2.5 Exercises\2.5.9.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

2.5.9.py

```
1  x = eval(input("How many Fibonacci numbers? - "))
2  a = 0
3  b = 1
4  print(b, end = ",  ")
5  for i in range(x):
6      c = a + b
7      print(c, end =",  ")
8      a = b
9      b = c
```

10. Use a **for** r loop to print a box like the one below. Allow the user to specify how wide and how high the box should be. [Hint:**print**('* '*10) prints ten asterisks.]

```
**********
**********
**********
**********
```

C:\Solutions to tasks\2.5 Exercises\2.5.10.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

2.5.10.py

```
length = eval(input("How wide is a box?: "))
height = eval(input("How high is a box?: "))
for i in range(height):
    print(" * " *length)
```

11. Use a **for** loop to print a box like the one below. Allow the user to specify how wide and how high the box should be.

```
**********
*        *
*        *
**********
```

C:\Solutions to tasks\2.5 Exercises\2.5.11.py - Sublime Text (UNREGISTERED)

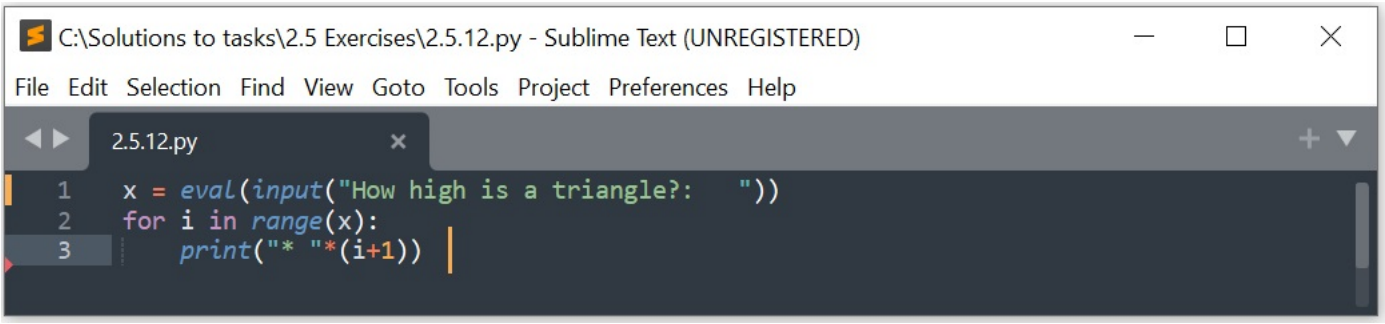
File Edit Selection Find View Goto Tools Project Preferences Help

2.5.11.py

```
"""
Run the program using py.exe.
Output of IDLE Shell 3.11.4 is incorrect

"""
length = eval(input("How wide is a box?: "))
height = eval(input("How hight is a box?: "))
print("* "*length)
for i in range(1,(height-1)):
    print("*" , "  "*(length-2)+ "*" )

print( "* "*length)
input()
```

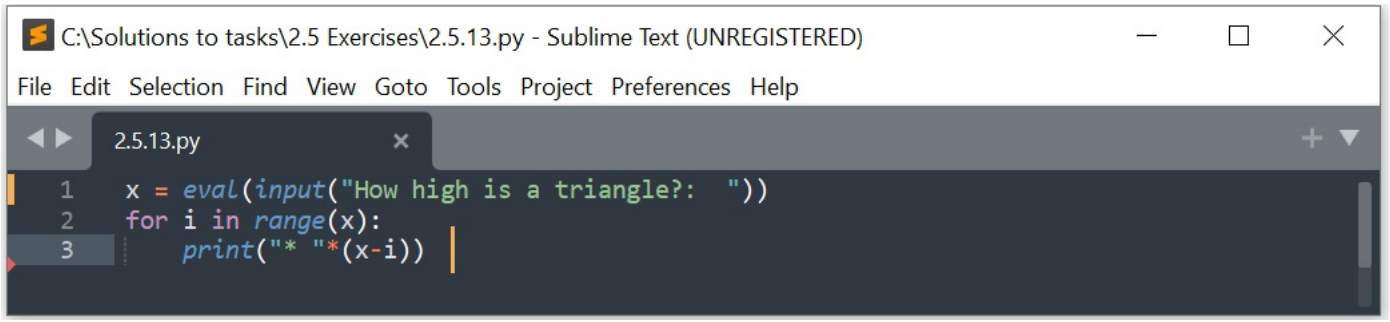
12. Use a **for** loop to print a triangle like the one below. Allow the user to specify how high the triangle should be.

```
*
**
***
****
```



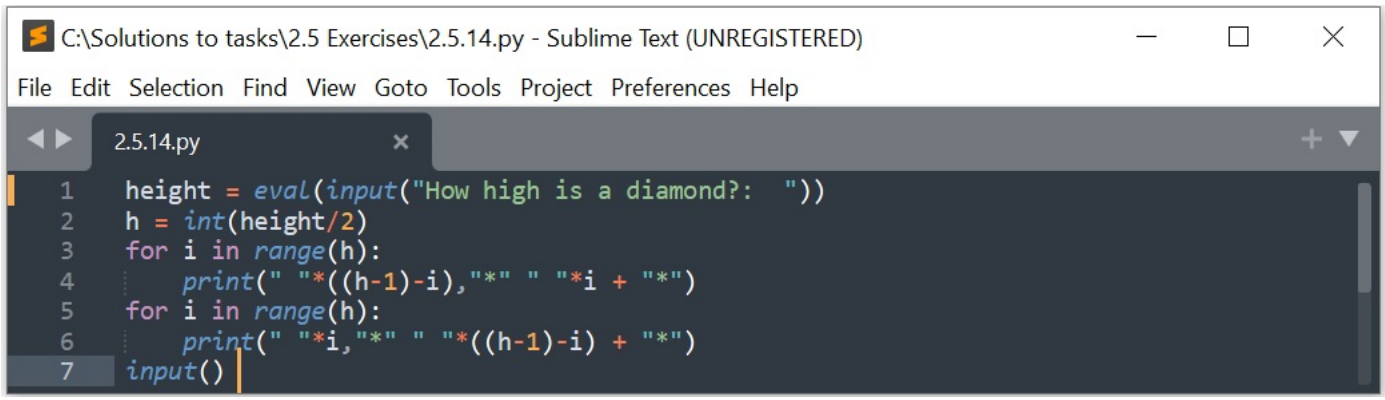
13. Use **for** loop to print an upside down triangle like the one below. Allow the user to specify how high the triangle should be.

```
****
***
**
*
```

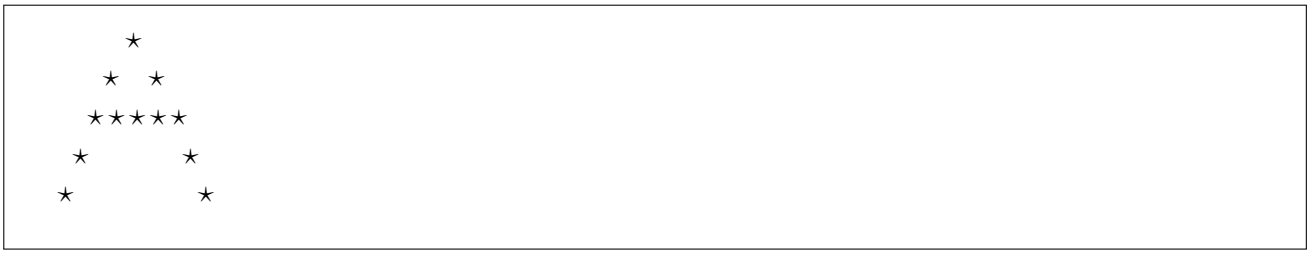


14. Use **for** loops to print a diamond like the one below. Allow the user to specify how high the diamond should be.

```
   *
  ***
 *****
*******
 *****
  ***
   *
```



15.Write a program that prints a giant letter A like the one below. Allow the user to specify how large the letter should be.



C:\Solutions to tasks\2.5 Exercises\2.5.15.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

2.5.15.py

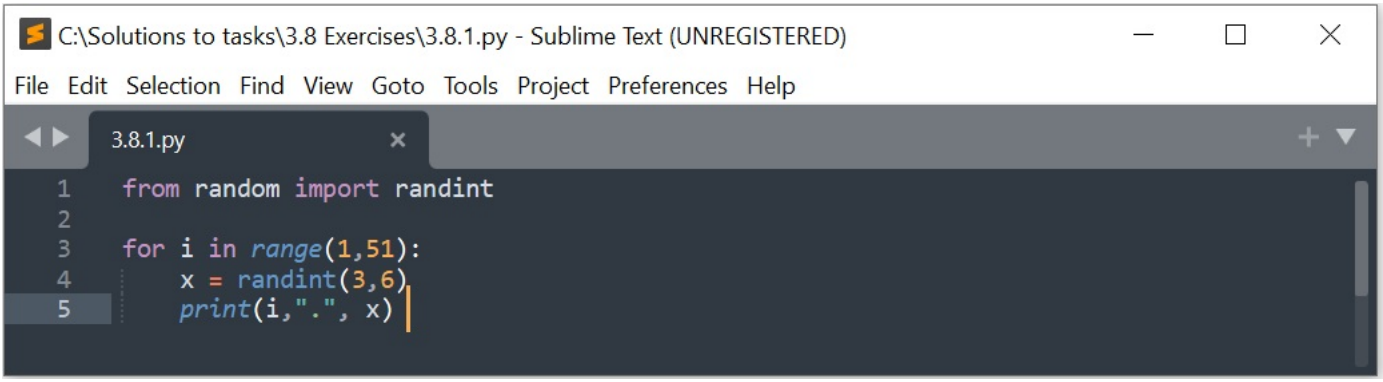
```
# written by Antonenko Igor

"""
Run the programm using py.exe.
Output of IDLE Shell 3.11.4 is incorrect.

"""
height = eval(input("How height is a letter A?: "))
print(" "*(height)+" * ")
for i in range(int(height/1.4)-1):
    print(" "*(height-i)+"*" +" "*(i*2)+ " *", )
print(" "*((height-i)-1)+ "*"*5+"*"*(i*2))
for i in range(int((height/1.4)-1) + 1, height-1):
    print(" "*(height-i)+"*" +" "*(i*2)+ " *",)
input()
```

Chapter 3

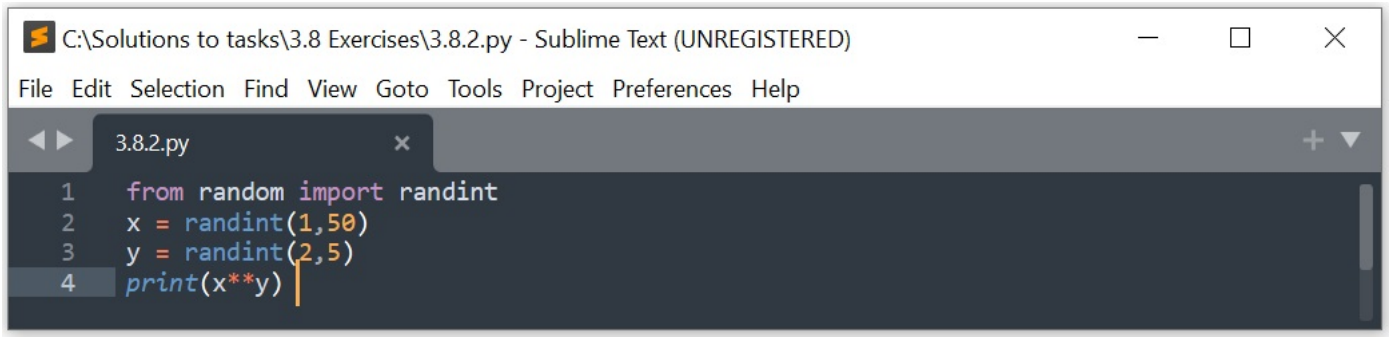
Numbers

3.1 Exercises 3.8

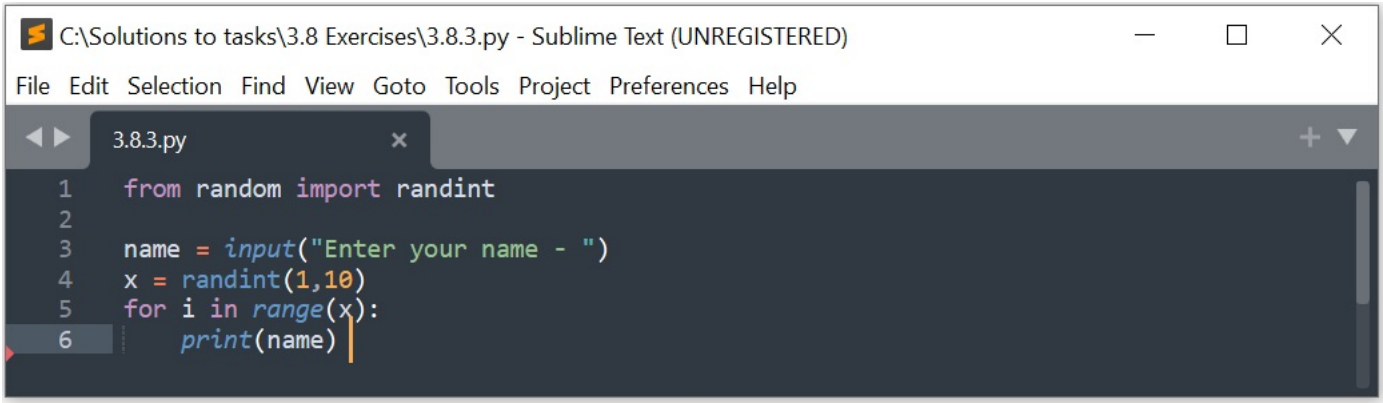
1. Write a program that generates and prints 50 random integers, each between 3 and 6.



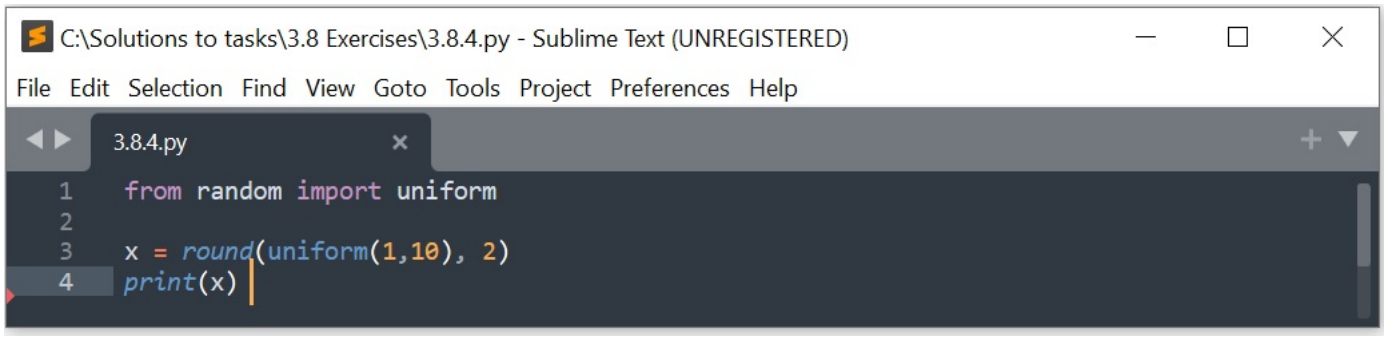
2. Write a program that generates a random number, x between 1 and 50, a random number y between 2 and 5, and computes x^y.



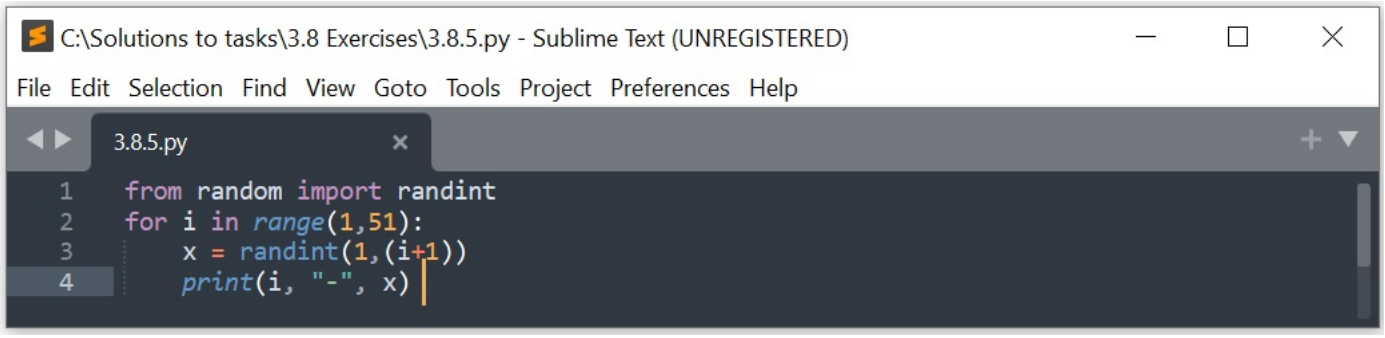
3. Write a program that generates a random number between 1 and 10 and prints your name that many times.



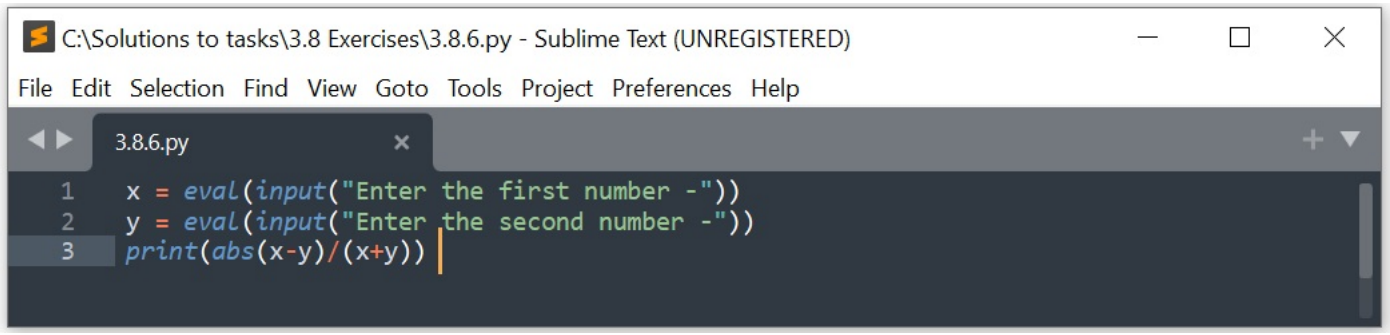
4. Write a program that generates a random decimal number between 1 and 10 with two decimal places of accuracy. Examples are 1.23, 3.45, 9.80, and 5.00.



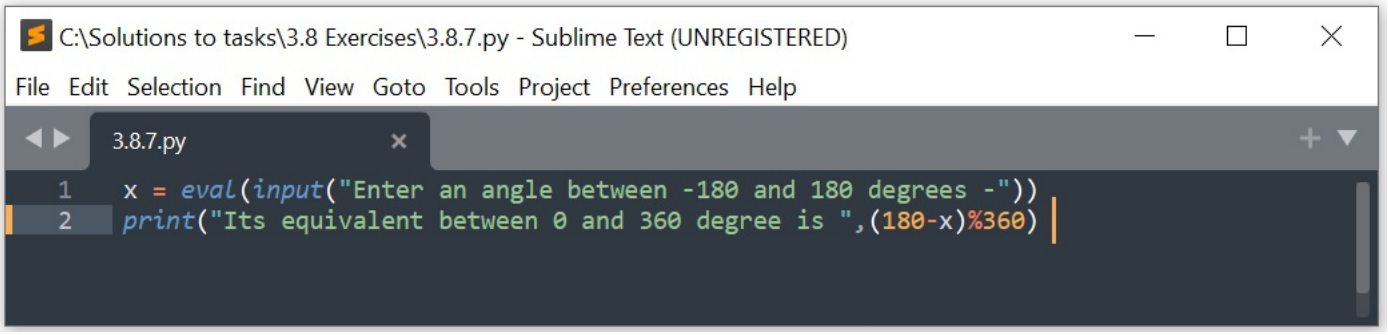
5. Write a program that generates 50 random numbers such that the first number is between 1 and 2, the second is between 1 and 3, the third is between 1 and 4,. . . , and the last is between 1 and 51.



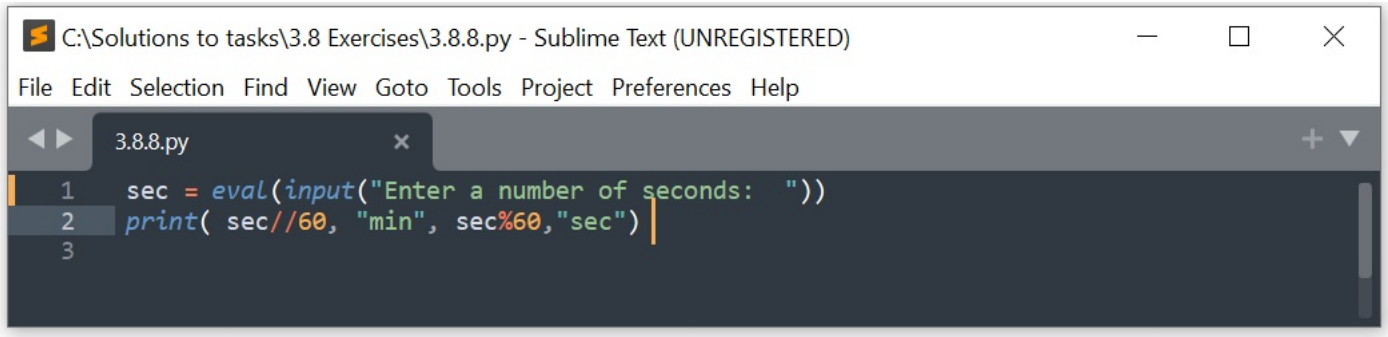
6. Write a program that asks the user to enter two numbers, x and y, and computes $\frac{|x-y|}{x+y}$.



7. Write a program that asks the user to enter an angle between $-180°$ and $180°$. Using an expression with the modulo operator, convert the angle to its equivalent between $0°$ and $360°$.

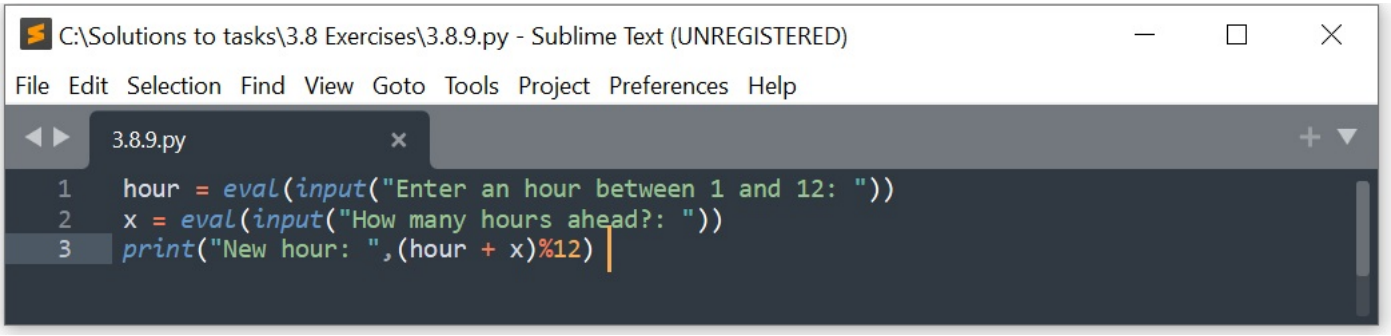


8. Write a program that asks the user for a number of seconds and prints out how many minutes and seconds that is. For instance, 200 seconds is 3 minutes and 20 seconds. [Hint: Use the // operator to get minutes and the %, operator to get seconds.]

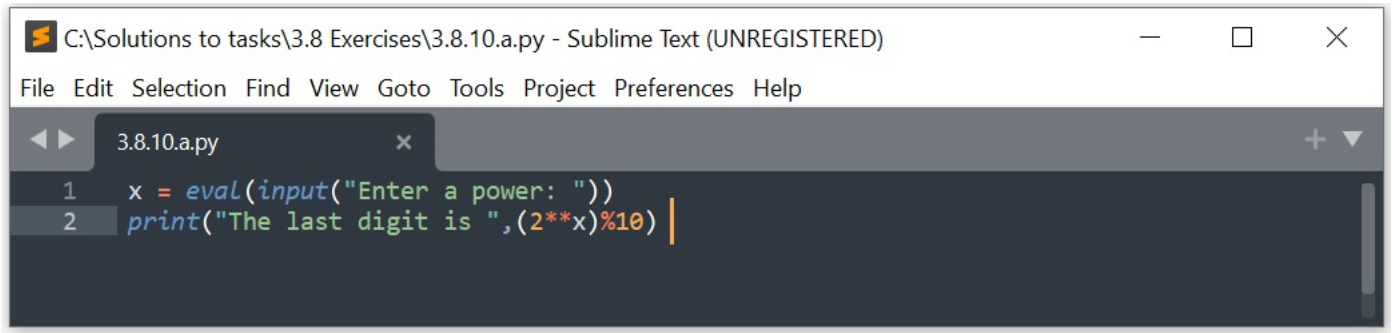


9. Write a program that asks the user for an hour between 1 and 12 and for how many hours in the future they want to go. Print out what the hour will be that many hours into the future. An example is shown below.

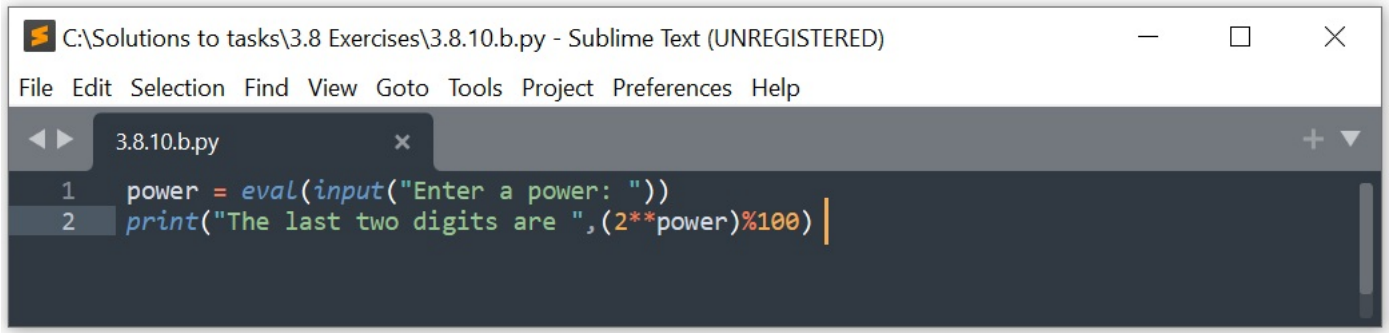
```
Enter hour: 8
How many hours ahead? 5
New hour: 1 o'clock
```



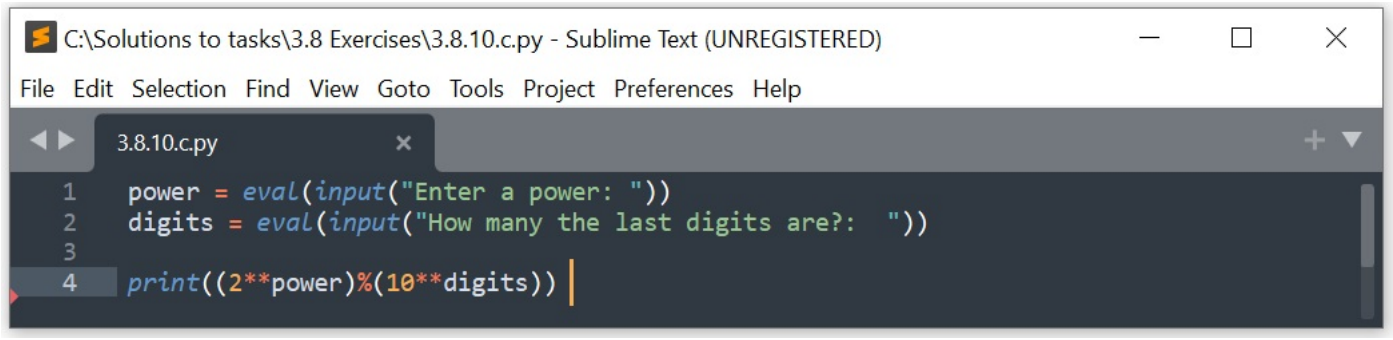
10.(a) One way to find out the last digit of a number is to mod the number by 10. Write a program that asks the user to enter a power. Then find the last digit of 2 raised to that power.



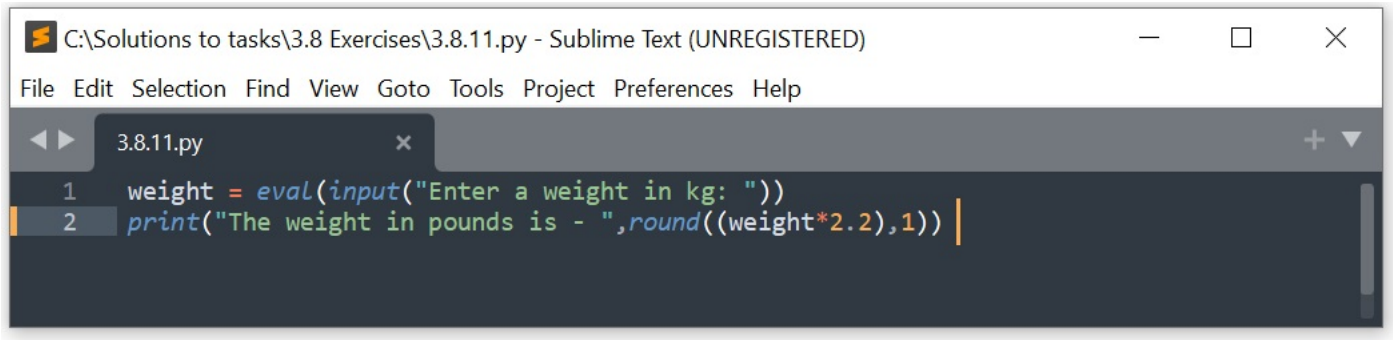
(b) One way to find out the last two digits of a number is to mod the number by 100. Write a program that asks the user to enter a power. Then find the last two digits of 2 raised to that power.



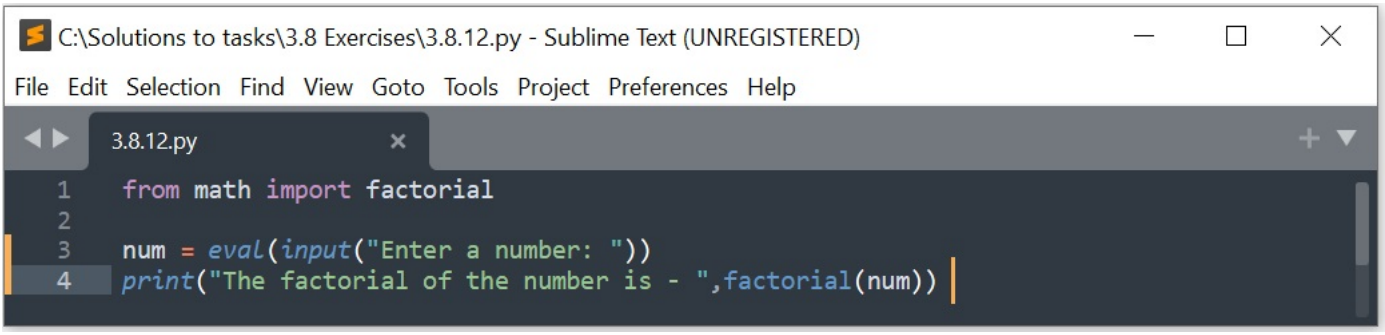
(c) Write a program that asks the user to enter a power and how many digits they want. Find the last that many digits of 2 raised to the power the user entered.



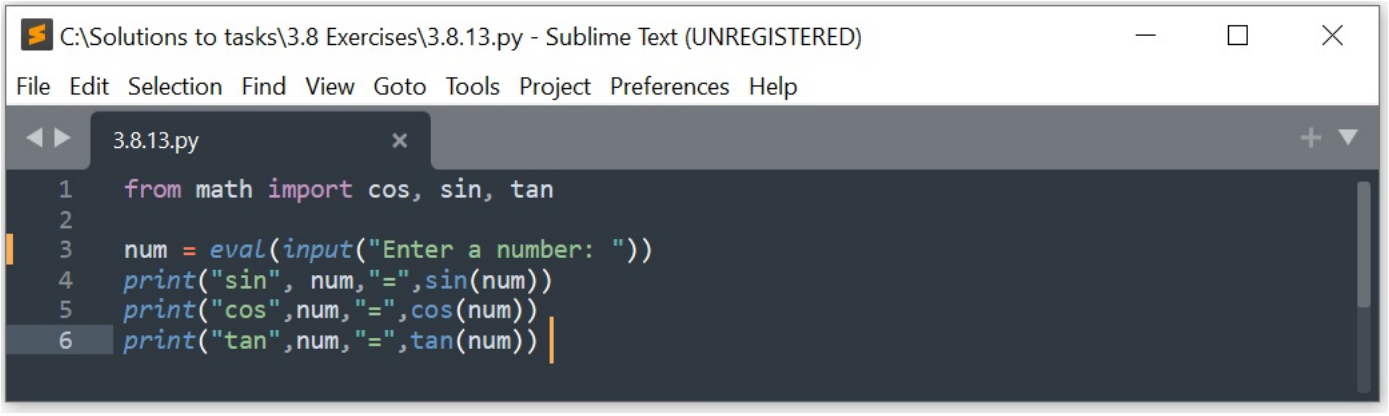
11. Write a program that asks the user to enter a weight in kilograms. The program should convert it to pounds, printing the answer rounded to the nearest tenth of a pound.



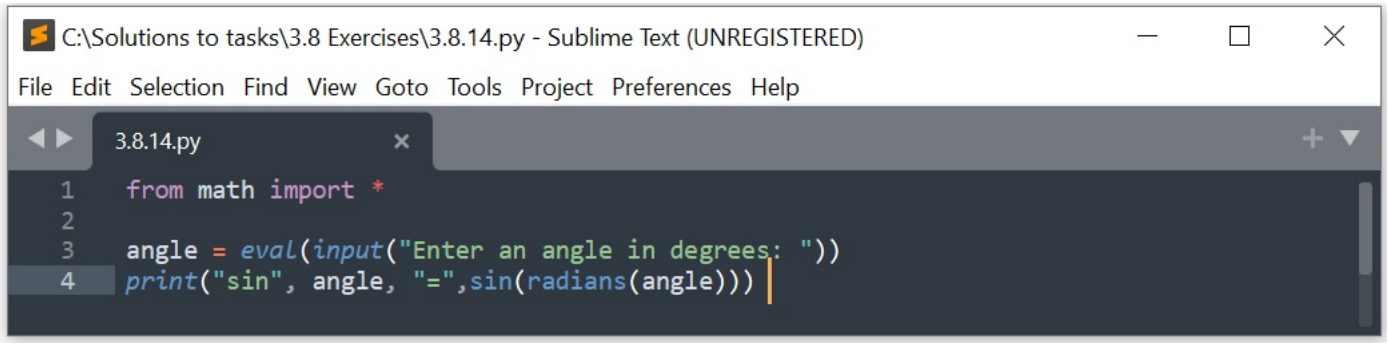
12. Write a program that asks the user for a number and prints out the factorial of that number.



13. Write a program that asks the user for a number and then prints out the sine, cosine, and tangent of that number.

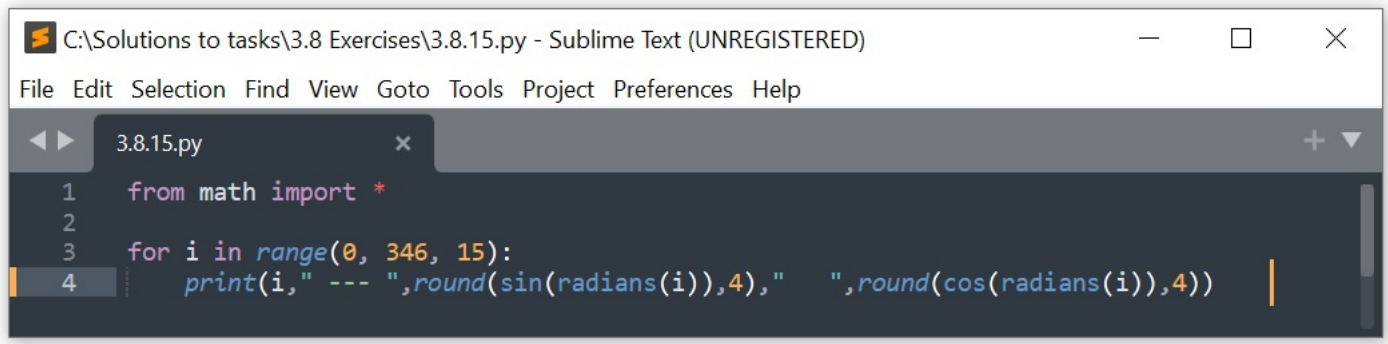


14. Write a program that asks the user to enter an angle in degrees and prints out the sine of that angle.



15. Write a program that prints out the sine and cosine of the angles ranging from 0 to 345° in 15°increments. Each result should be rounded to 4 decimal places. Sample output is shown below:

0	- - -	0.0	1.0
15	- - -	0.2588	0.9659
30	- - -	0.5	0.866
...			
345	- - -	-0.2588	0.9659



16. Below is described how to find the date of Easter in any year. Despite its intimidating appearance, this is not a hard problem. Note that $\lfloor \mathbf{x} \rfloor$ is the *floor* function, which for positive numbers just drops the decimal part of the number. For instance $\lfloor 3.14 \rfloor = 3$. The *floor* function is part of the *math* module.

C = century ($1900's \rightarrow C = 19$)
Y = year (all four digits)
m = (15 + C - $\lfloor \frac{C}{4} \rfloor - \lfloor \frac{8C+13}{25} \rfloor$) mod 30
n = ($4 + C - \lfloor \frac{C}{4} \rfloor$) mod 7
a = Y mod 4
b = Y mod 7
c = Y mod 19
d = (19c + m) mod 30
e = (2a + 4b + 6d + n) mod 7

Easter is either March (22+d+e) or April (d+e-9). There is an exception if d = 29 and e = 6. In this case, Easter falls one week earlier on April 19. There is another exception if d = 28, e = 6, and m = 2, 5, 10, 13, 16, 21, 24, or 39. In this case, Easter falls one week earlier on April 18. Write a program that asks the user to enter a year and prints out the date of Easter in that year. (See Tattersall, Elementary Number Theory in Nine Chapters, 2nd ed., page 167)

C:\Solutions to tasks\3.8 Exercises\3.8.16.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

3.8.16.py

```
from math import floor
Y = eval(input("Enter any year: "))
C = int(Y/100)
m = ((15 + C - floor(C/4) - floor((8*C + 13)/25)))%30
n = ((4 + C - floor(C/4)))%7
a = Y%4
b = Y%7
c = Y%19
d = (19*c + m)%30
e = (2*a + 4*b + 6*d + n)%7
if d ==29 and e == 6:
   print(" 19 April")
if d ==28 and e ==  6 and (m == 2 or m == 5 or m == 10 or m == 13 /n
   or m == 16 or m == 21 or m == 24 or m == 39):
   print("18 April")
if (22 + d + e) > 31:
   print((d + e - 9), "April")
else:
   print(22 + d + e, "March")
```

17. A year is a leap year if it is divisible by 4, except that years divisible by 100 are not leap years unless they are also divisible by 400. Ask the user to enter a year, and, using the // operator, determine how many leap years there have been between 1600 and that year.

C:\Solutions to tasks\3.8 Exercises\3.8.17.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

3.8.17.py

```
year = eval(input("Enter a year: "))
c = 0
for i in range(1600, year+1):
    if i % 4 == 0 and i % 100 != 0 or i % 400 == 0:
        c = c + 1
print( "The number of the leap year is ", c)
```

18. Write a program that given an amount of change less than $1 will print out exactly how many quarters, dimes, nickels, and pennies will be needed to efficiently make that change. [Hint: the // operator may be useful.]

C:\Solutions to tasks\3.8 Exercises\3.8.18.py - Sublime Text (UNREGISTERED)

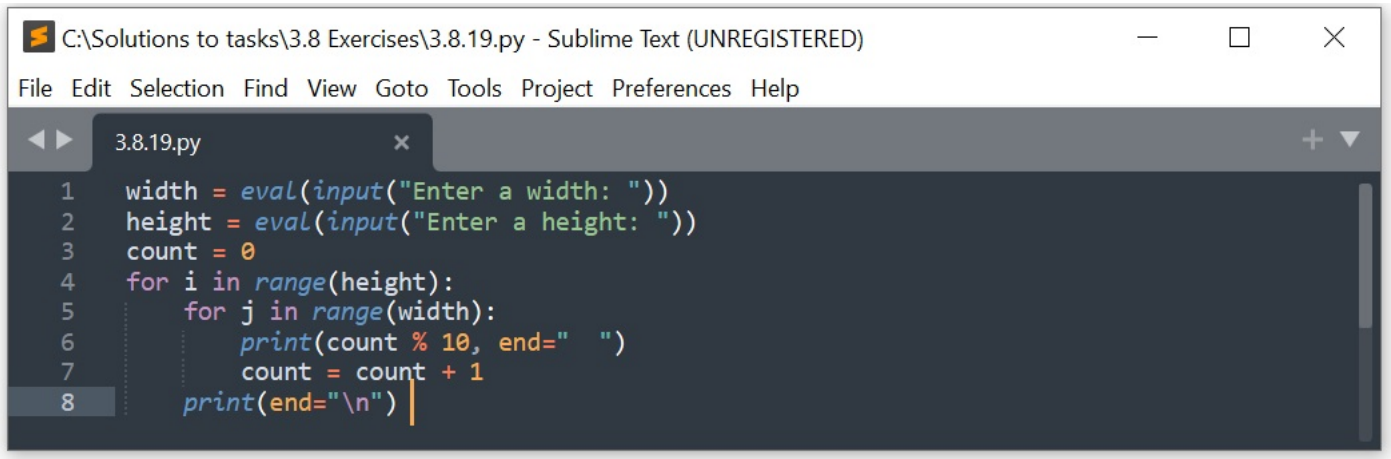
File Edit Selection Find View Goto Tools Project Preferences Help

3.8.18.py

```
c = eval(input('Enter an amount of change less than $1.00: '))
print(c // 25, " - quarters")
c = c % 25
print(c // 10, " - dimes")
c = c % 10
print(c // 5, " -  nickels")
c = c % 5
print(c // 1, " -  pennies")
```

19. Write a program that draws ″modular rectangles″like the ones below. The user specifies the width and height of the rectangle, and the entries start at 0 and increase typewriter fashion from left to right and top to bottom, but are all done mod 10. Below are examples of a 3×5 rectangle and a 4×8.

```
0 1 2 3 4
5 6 7 8 9
0 1 2 3 4

0 1 2 3 4 5 6 7
8 9 0 1 2 3 4 5
6 7 8 9 0 1 2 3
4 5 6 7 8 9 0 1
```

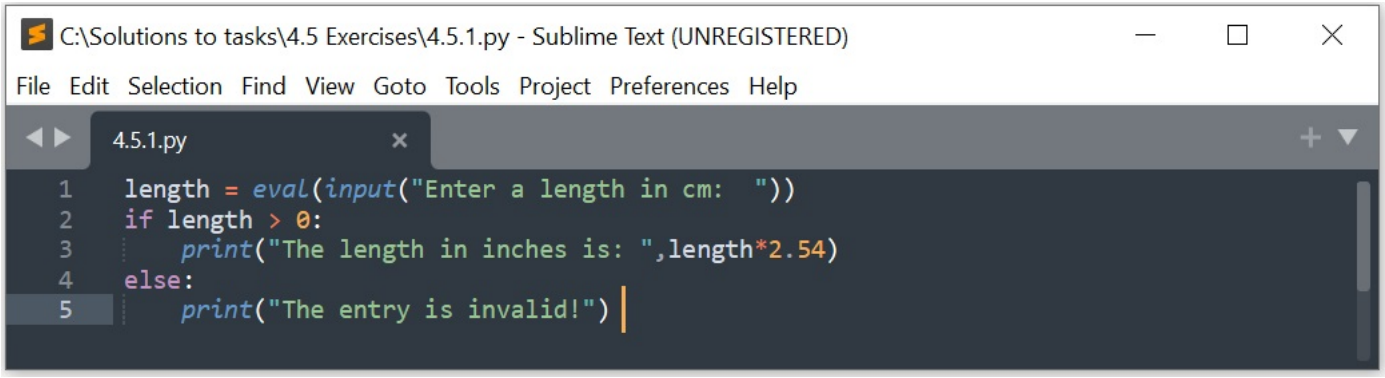


Chapter 4

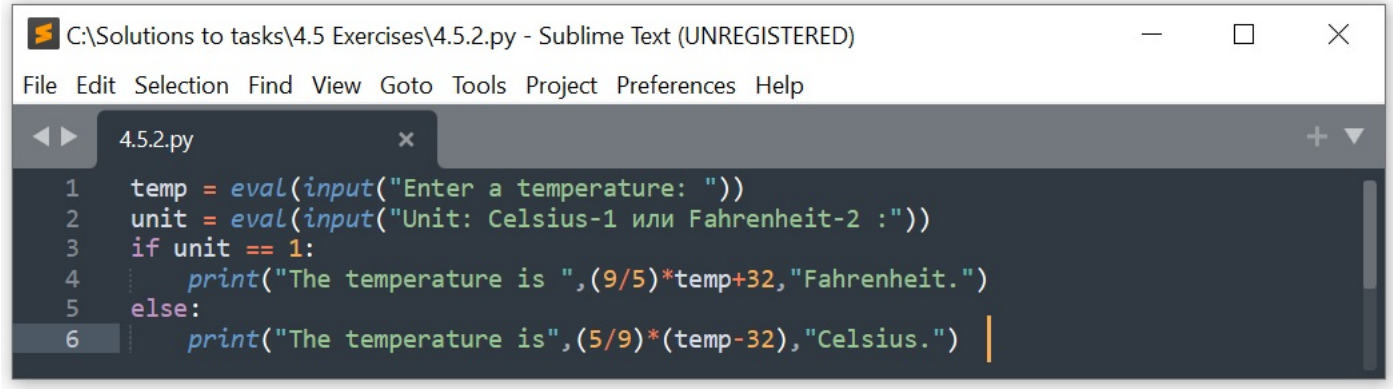
If statements

4.1 Exercises 4.5

1. Write a program that asks the user to enter a length in centimeters. If the user enters a negative length, the program should tell the user that the entry is invalid. Otherwise, the program should convert the length to inches and print out the result. There are 2.54 centimeters in an inch.



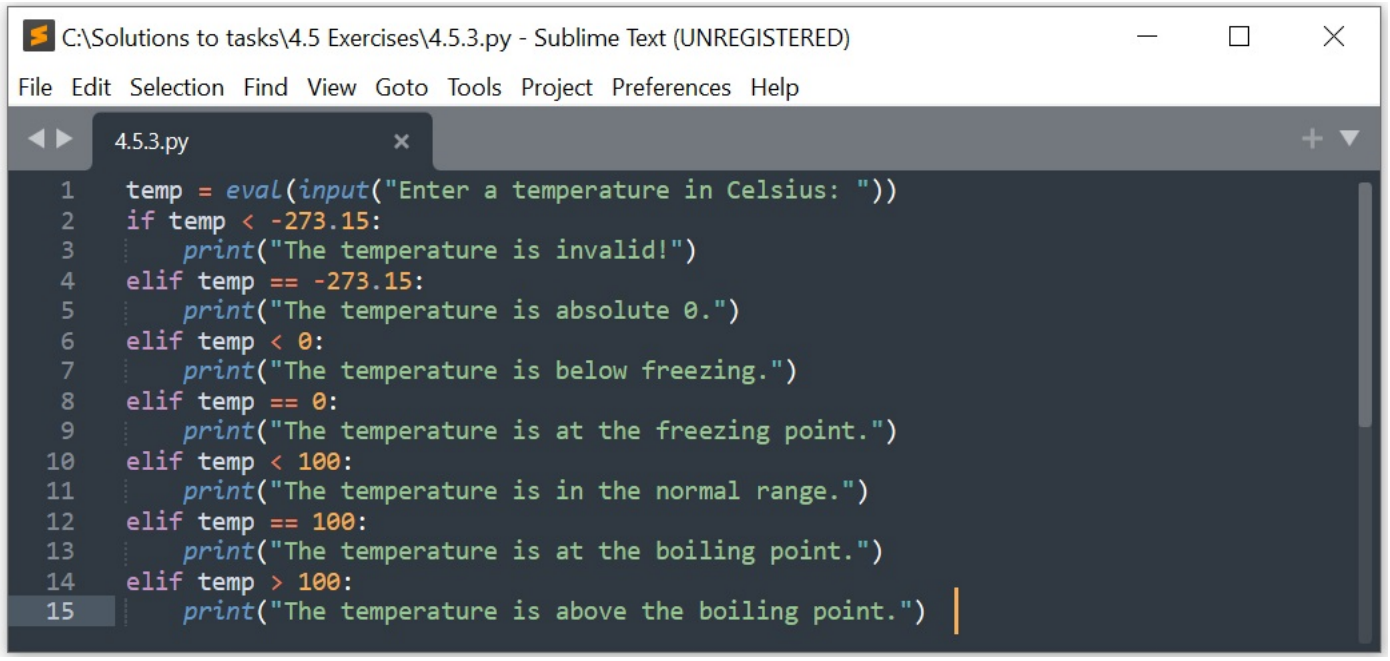
2. Ask the user for a temperature. Then ask them what units, Celsius or Fahrenheit, the temperature is in. Your program should convert the temperature to the other unit. The conversions are $F = \frac{9}{5}C + 32$ and $C = \frac{5}{9}(F - 32)$.



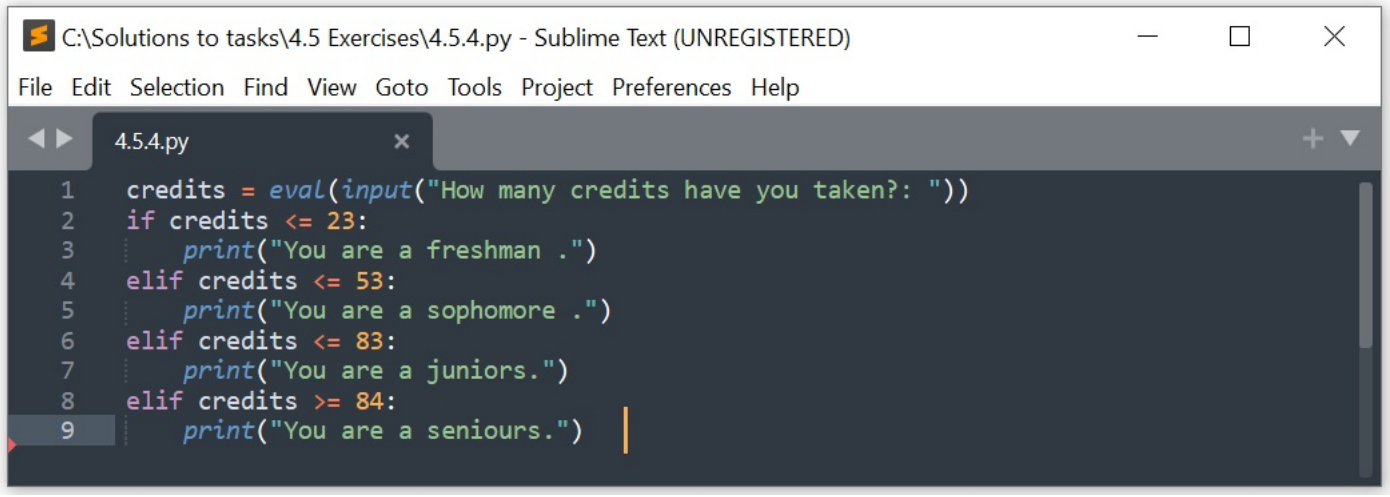
3. Ask the user to enter a temperature in Celsius. The program should print a message based on the temperature::

- If the temperature is less than -273.15, print that the temperature is invalid because it is below absolute zero.
- If it is exactly -273.15, print that the temperature is absolute 0.
- If the temperature is between -273.15 and 0, print that the temperature is below freezing.

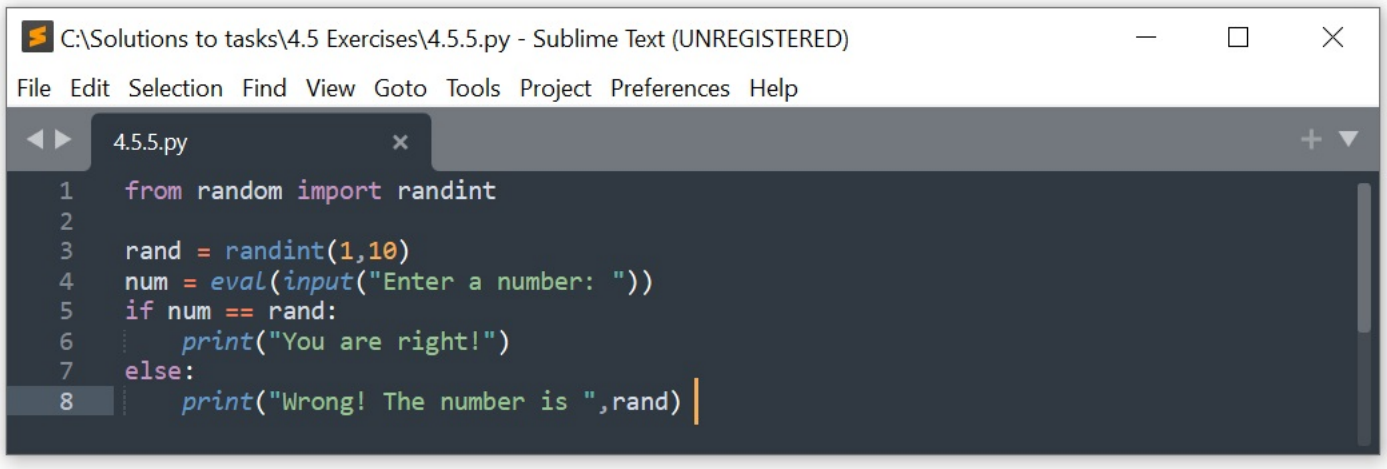
- If it is 0, print that the temperature is at the freezing point.
- If it is between 0 and 100, print that the temperature is in the normal range.
- If it is 100, print that the temperature is at the boiling point.
- If it is above 100, print that the temperature is above the boiling point.



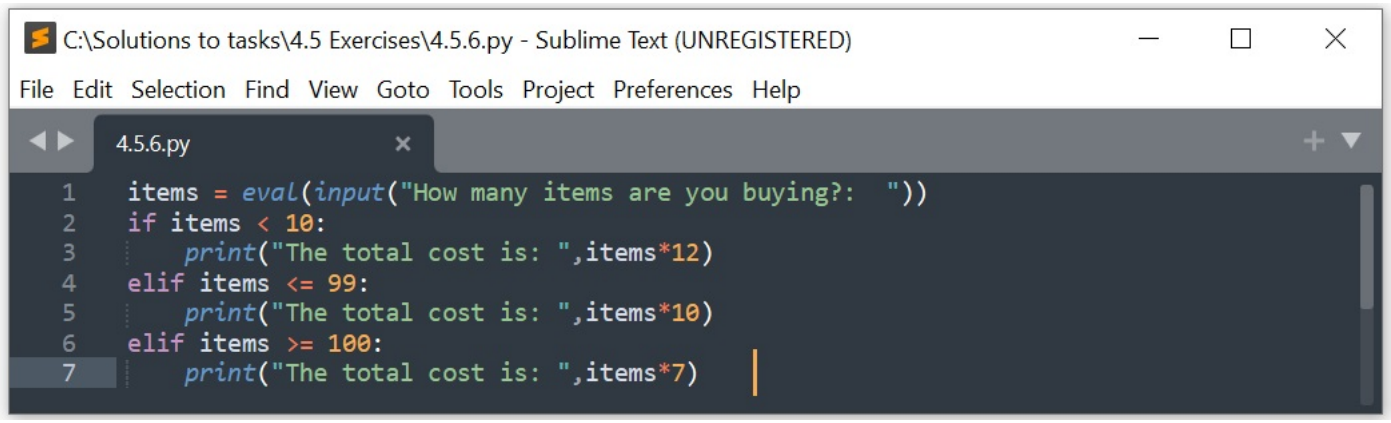
4. Write a program that asks the user how many credits they have taken. If they have taken 23 or less, print that the student is a freshman. If they have taken between 24 and 53, print that they are a sophomore. The range for juniors is 54 to 83, and for seniors it is 84 and over.



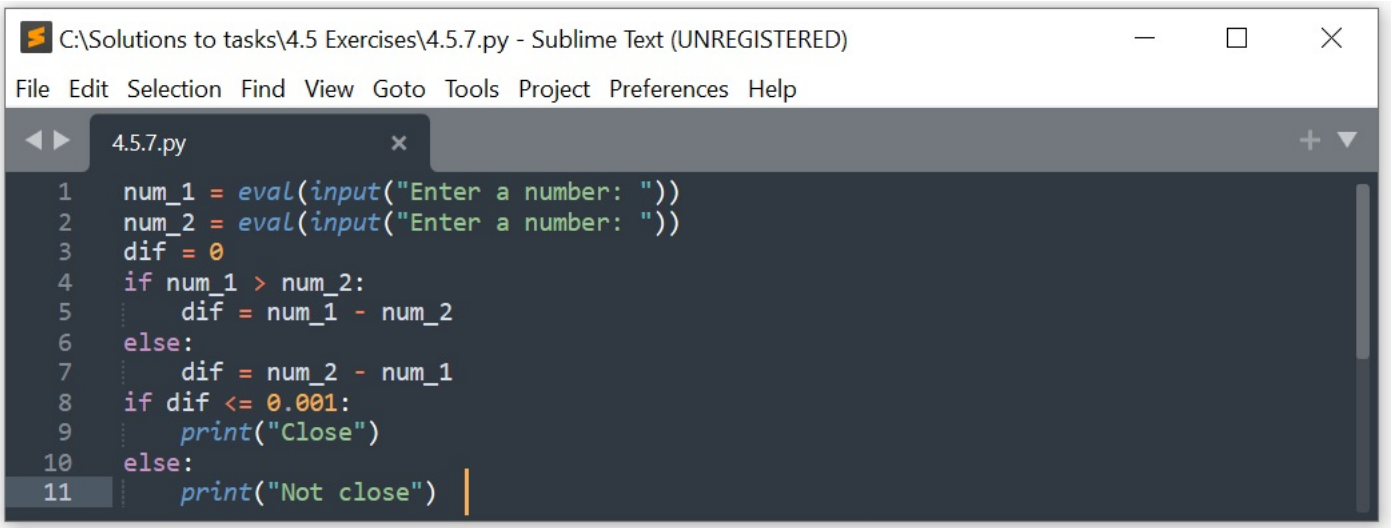
5.Generate a random number between 1 and 10. Ask the user to guess the number and print a message based on whether they get it right or not.



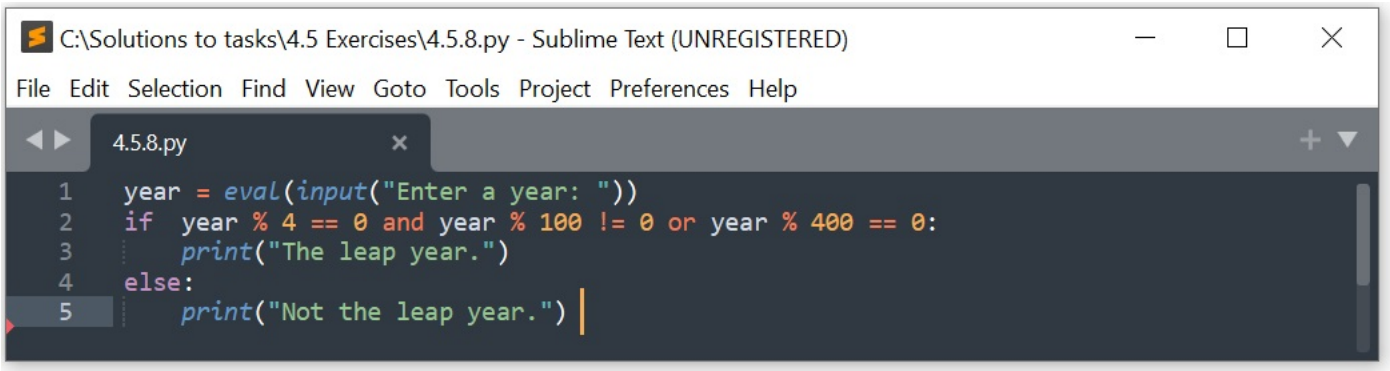
6. A store charges 12$ per item if you buy less than 10 items. If you buy between 10 and 99 items, the cost is 10$ per item. If you buy 100 or more items, the cost is 7$ per item. Write a program that asks the user how many items they are buying and prints the total cost.



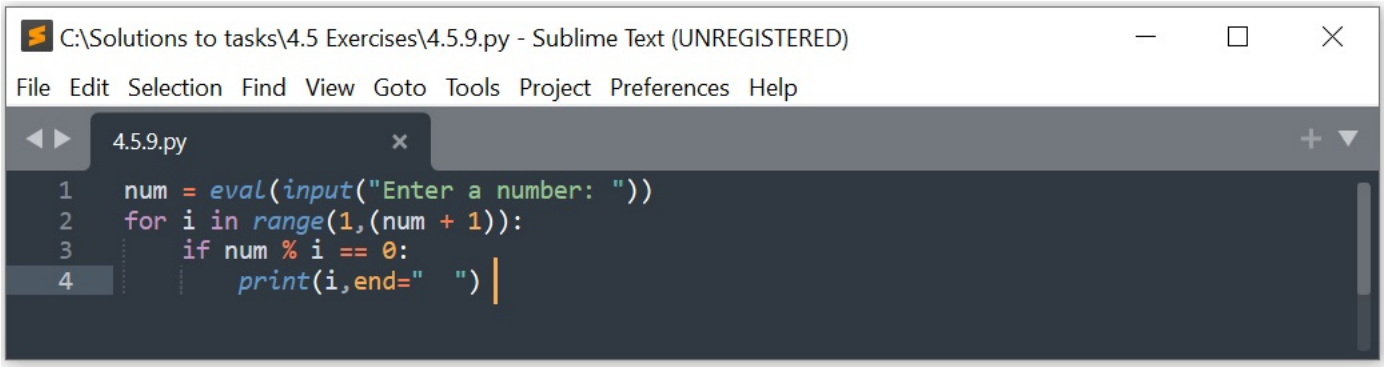
7. Write a program that asks the user for two numbers and prints *Close* if the numbers are within .001 of each other and *Not close* otherwise.



8. A year is a leap year if it is divisible by 4, except that years divisible by 100 are not leap years unless they are also divisible by 400. Write a program that asks the user for a year and prints out whether it is a leap year or not.

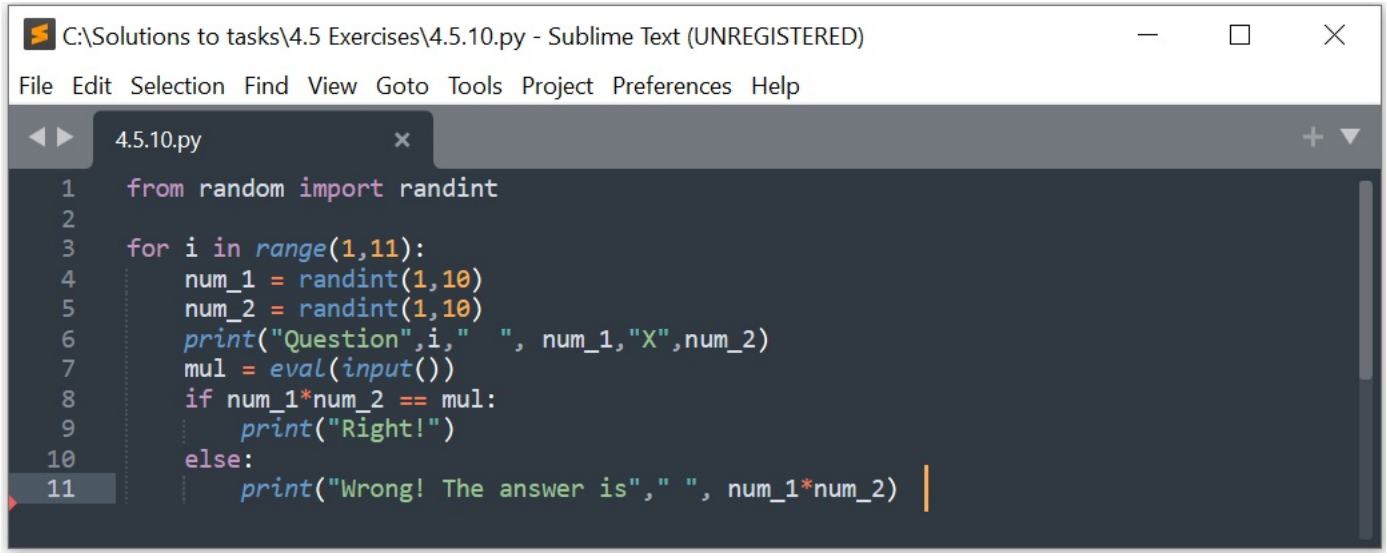


9. Write a program that asks the user to enter a number and prints out all the divisors of that number. [Hint: the % operator is used to tell if a number is divisible by something. See Section 3.2.]



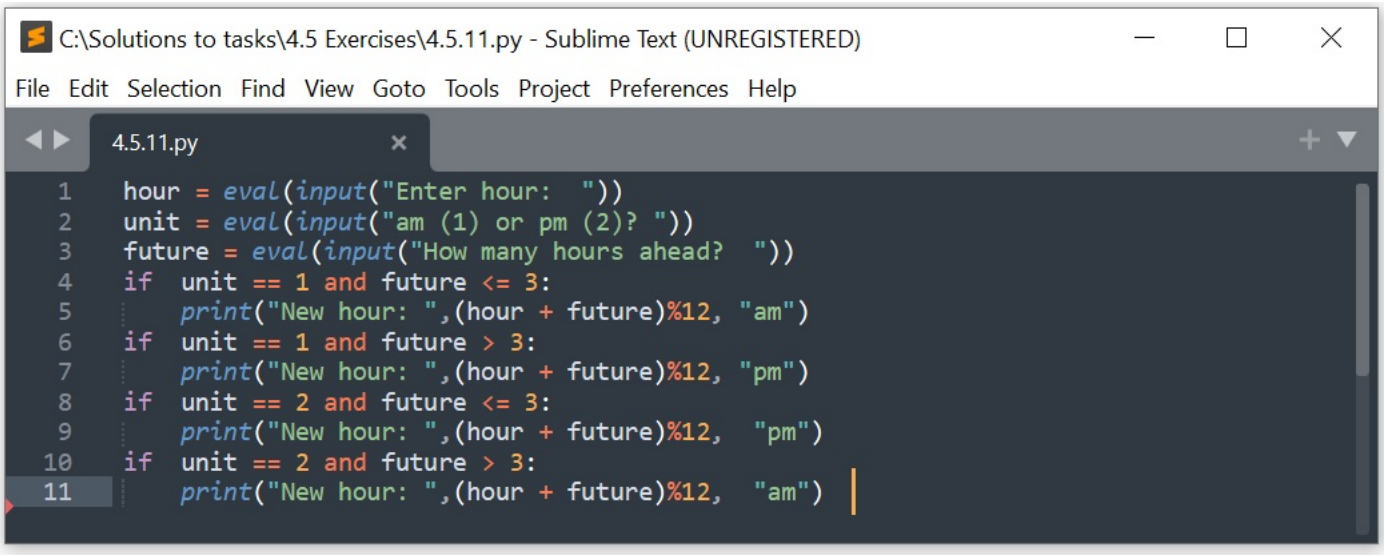
10.Write a multiplication game program for kids. The program should give the player ten randomly generated multiplication questions to do. After each, the program should tell them whether they got it right or wrong and what the correct answer is.

```
Question 1: 3 x 4 = 12
Right!
Question 2: 8 x 6 = 44
Wrong. The answer is 48.
...
...
Question 10: 7 x 7 = 49
Right!
```

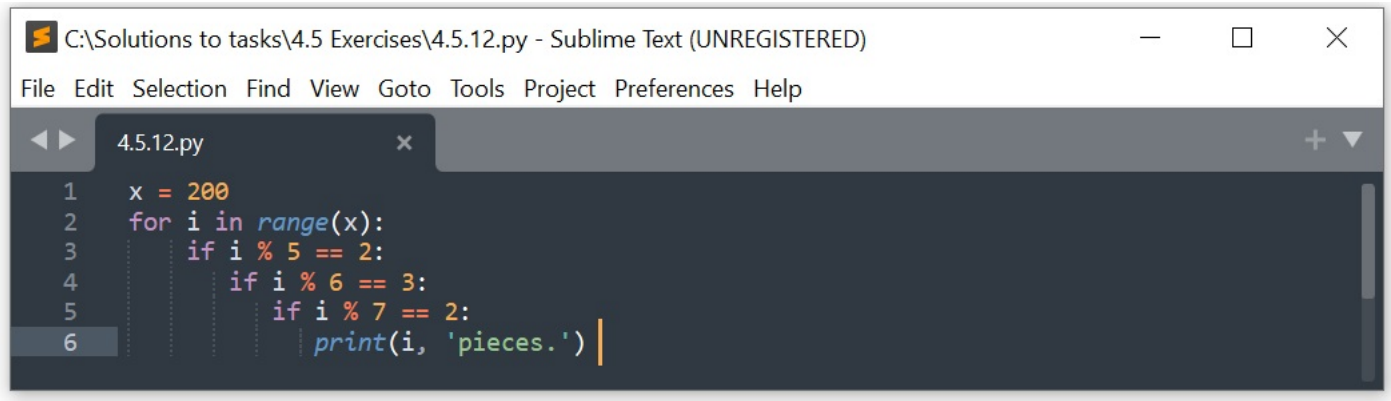


11. Write a program that asks the user for an hour between 1 and 12, asks them to enter *am* or *pm*, and asks them how many hours into the future they want to go. Print out what the hour will be that many hours into the future, printing *am* or *pm* as appropriate. An example is shown below.

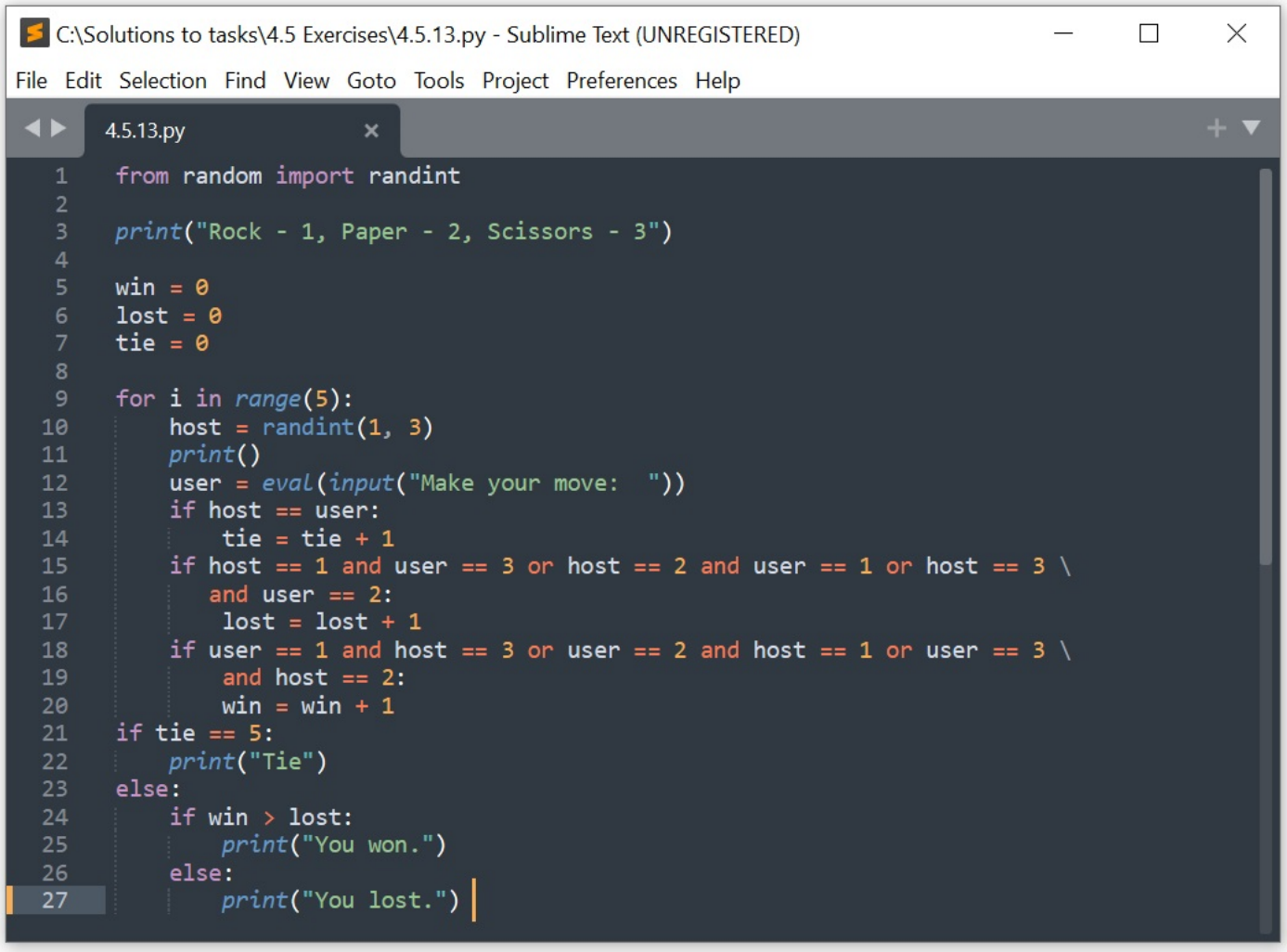
```
Enter hour: 8
am (1) or pm (2)? 1
How many hours ahead? 5
New hour: 1 pm
```



12.A jar of Halloween candy contains an unknown amount of candy and if you can guess exactly how much candy is in the bowl, then you win all the candy. You ask the person in charge the following: If the candy is divided evenly among 5 people, how many pieces would be left over? The answer is 2 pieces. You then ask about dividing the candy evenly among 6 people, and the amount left over is 3 pieces. Finally, you ask about dividing the candy evenly among 7 people, and the amount left over is 2 pieces. By looking at the bowl, you can tell that there are less than 200 pieces. Write a program to determine how many pieces are in the bowl.



13. Write a program that lets the user play Rock-Paper-Scissors against the computer. There should be five rounds, and after those five rounds, your program should print out who won and lost or that there is a tie.

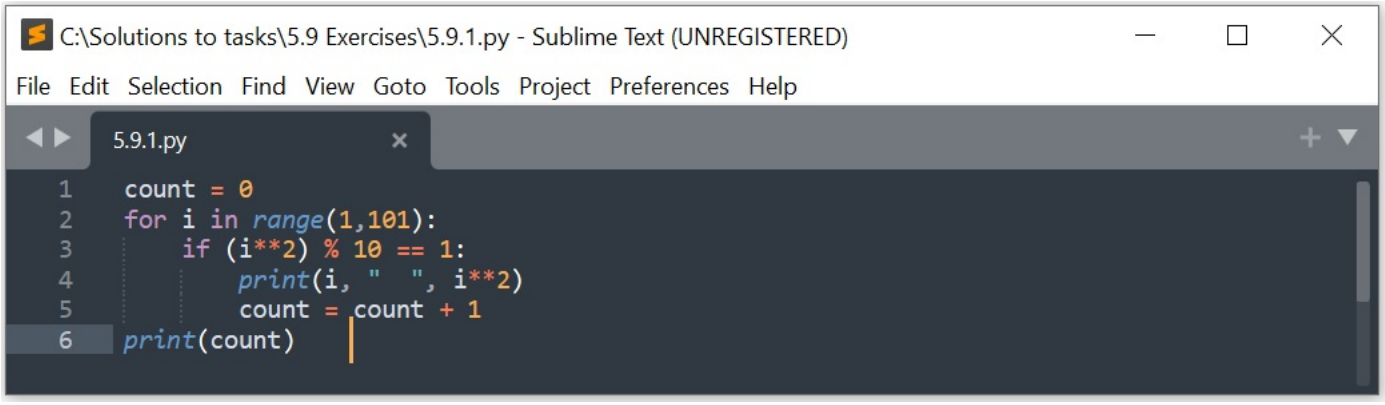

C:\Solutions to tasks\4.5 Exercises\4.5.13.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
4.5.13.py
from random import randint
print("Rock - 1, Paper - 2, Scissors - 3")
win = 0
lost = 0
tie = 0
for i in range(5):
host = randint(1, 3)
print()
user = eval(input("Make your move: "))
if host == user:
tie = tie + 1
if host == 1 and user == 3 or host == 2 and user == 1 or host == 3 \
and user == 2:
lost = lost + 1
if user == 1 and host == 3 or user == 2 and host == 1 or user == 3 \
and host == 2:
win = win + 1
if tie == 5:
print("Tie")
else:
if win > lost:
print("You won.")
else:
print("You lost.")

Chapter 5

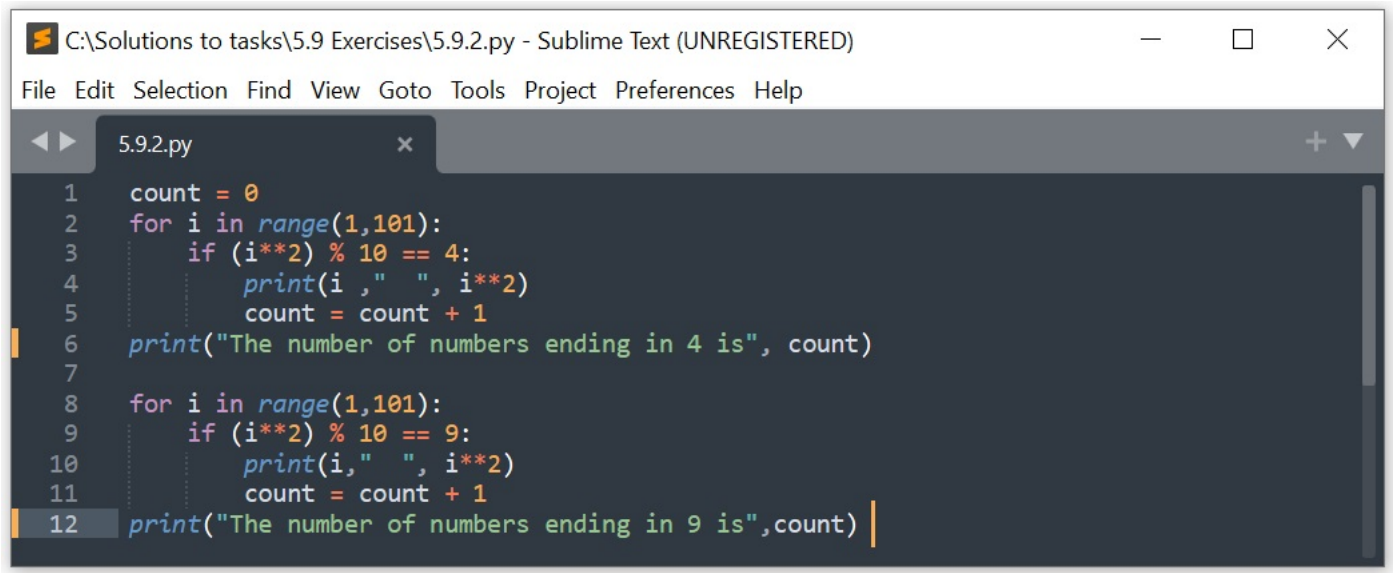
Miscellaneous Topics I

5.1 Exercises 5.9

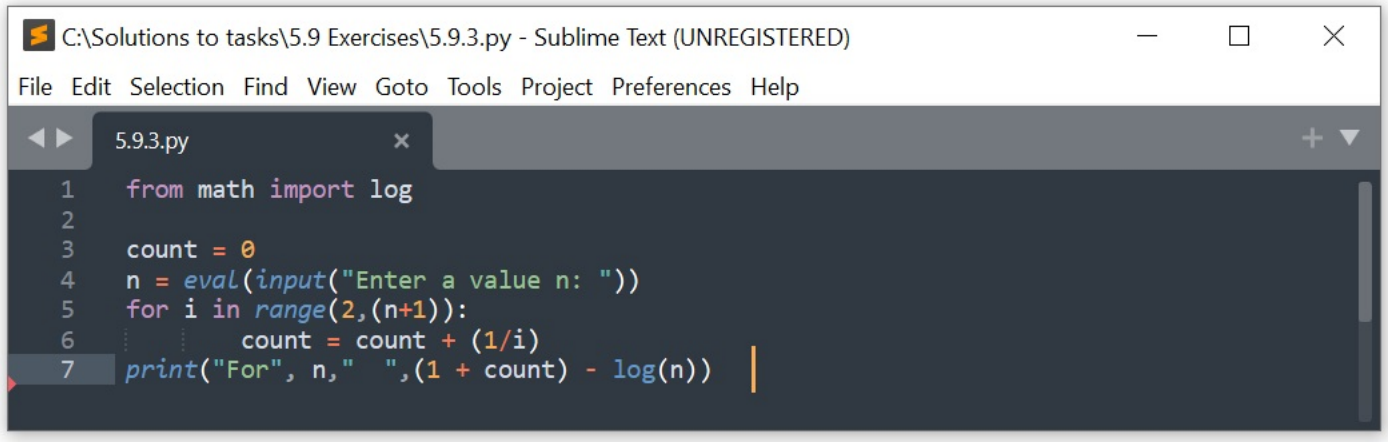
1. Write a program that counts how many of the squares of the numbers from 1 to 100 end in a 1.



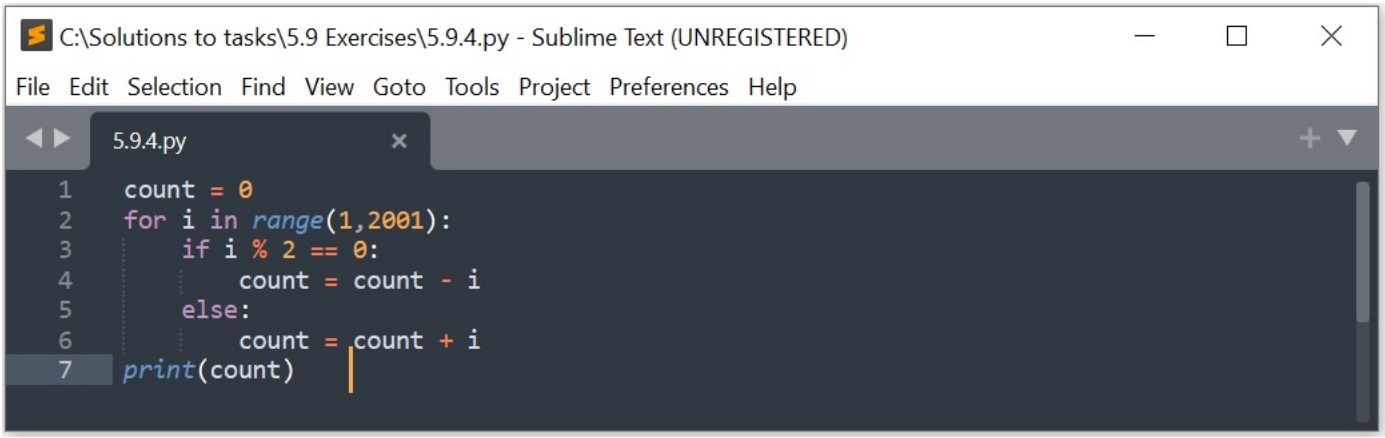
2. Write a program that counts how many of the squares of the numbers from 1 to 100 end in a 4 and how many end in a 9.



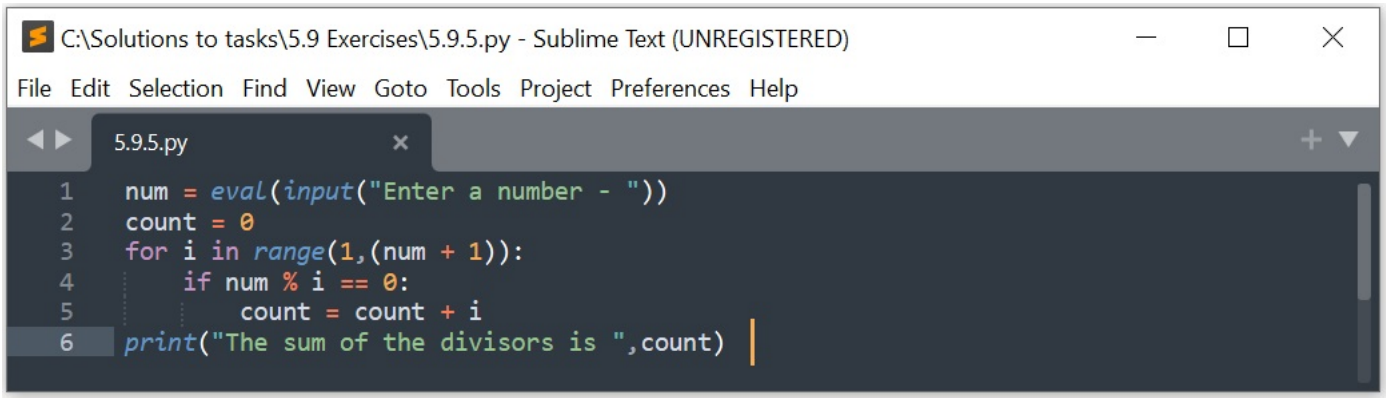
3. Write a program that asks the user to enter a value n, and then computes $(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n})-ln(n)$. The ln function is *log* in the *math* module.



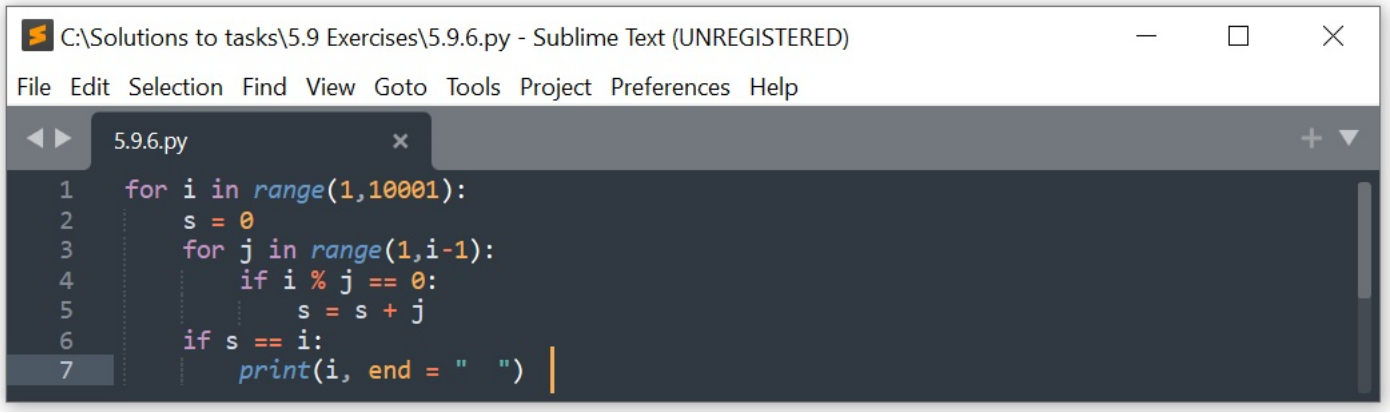
4. Write a program to compute the sum $1 - 2 + 3 - 4 + \cdots + 1999 - 2000$



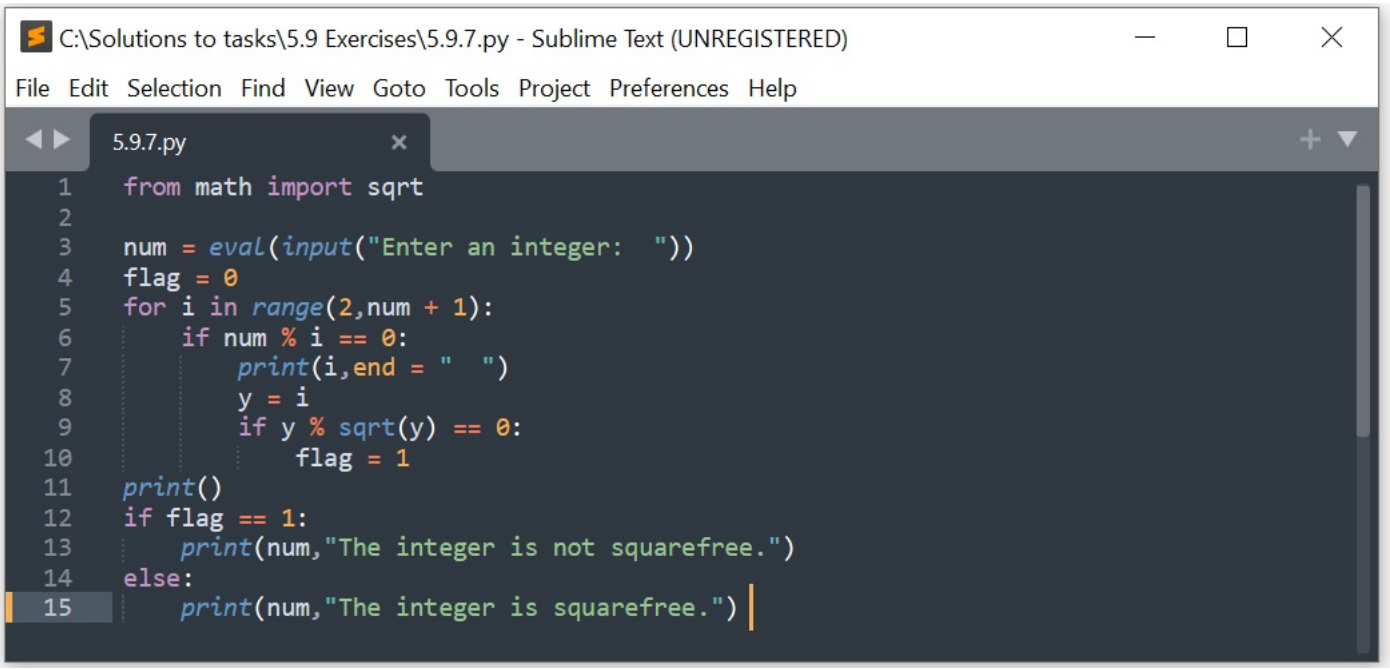
5. Write a program that asks the user to enter a number and prints the sum of the divisors of that number. The sum of the divisors of a number is an important function in number theory



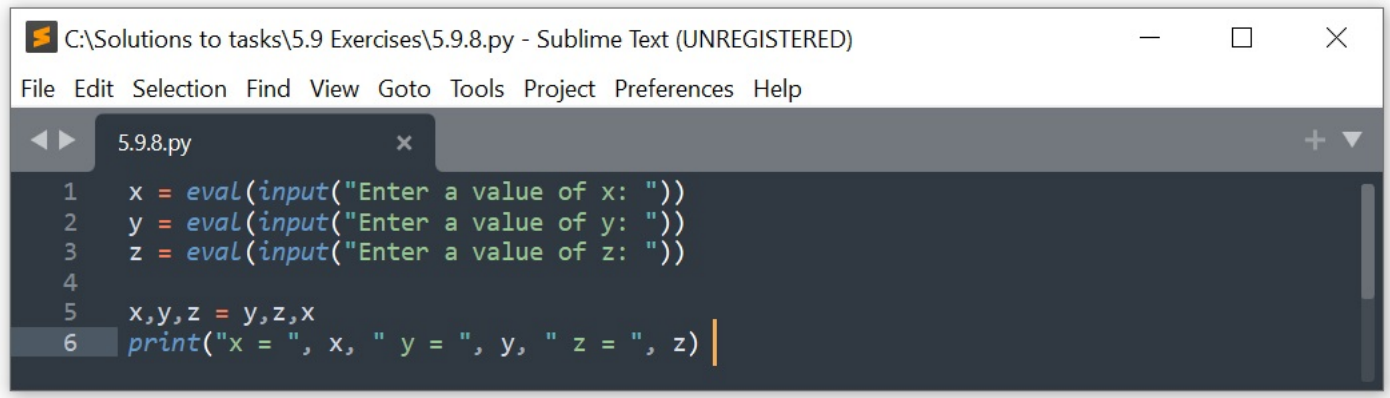
6. A number is called a *perfect number* if it is equal to the sum of all of its divisors, not including the number itself. For instance, 6 is a perfect number because the divisors of 6 are 1, 2, 3, 6 and $6 = 1 + 2 + 3$. As another example, 28 is a perfect number because its divisors are 1, 2, 4, 7, 14, 28 and $28 = 1 + 2 + 4 + 7 + 14$. However, 15 is not a perfect number because its divisors are 1, 3, 5, 15 and 15 $\neq$. Write a program that finds all four of the perfect numbers that are less than 10000.



7. An integer is called *squarefree* if it is not divisible by any perfect squares other than 1. For instance, 42 is squarefree because its divisors are 1, 2, 3, 6, 7, 21, and 42, and none of those numbers (except 1) is a perfect square. On the other hand, 45 is not squarefree because it is divisible by 9, which is a perfect square. Write a program that asks the user for an integer and tells them if it is squarefree or not.



8. Write a program that swaps the values of three variables x, y, and z, so that x gets the value of y, y gets the value of z, and z gets the value of x.



9.Write a program to count how many integers from 1 to 1000 are not perfect squares, perfect cubes, or perfect fifth powers.

C:\Solutions to tasks\5.9 Exercises\5.9.9.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.9.py

```
s = 0
for i in range(1, 1001):
    if i % round(i**(1./2), 4) != 0:
        s = s + 1
print("x^2 - ",s)
print()
c = 0
for i in range(1, 1001):
    if i % round(i**(1./3), 4) != 0:
        c = c + 1
print("x^3 - ",c)
print()
f = 0
for i in range(1, 1001):
    if i % round(i**(1./5), 4) != 0:
        f = f + 1
print("x^5 - ",f)
```

10. Ask the user to enter 10 test scores. Write a program to do the following:

(a). Print out the highest and lowest scores.

C:\Solutions to tasks\5.9 Exercises\5.9.10.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.10.a.py

```
score = eval(input("Enter a test score:  "))
highest = score
lowest = score

for i in range(9):
    score = eval(input("Enter a test score:  "))
    if score > highest:
        highest = score
    if score < lowest:
        lowest = score
print("The highest score - ",highest )
print("The lowest - ",lowest)
```

(b). Print out the average of the scores.

C:\Solutions to tasks\5.9 Exercises\5.9.10.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.10.b.py

```
s = 0
for i in range(10):
    score = (eval(input("Enter a test score: ")))
    s = s + score
print("The average of the scores - ",s/10)
```

(c). Print out the second largest score.

C:\Solutions to tasks\5.9 Exercises\5.9.10.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.10.c.py

```
a = eval(input("Enter a test score - "))
b = eval(input("Enter a test score - "))
if a < b:
    high_1 = b
    high_2 = a
else:
    high_1 = a
    high_2 = b
for i in range(8):
    num = eval(input("Enter a test score -"))
    if  high_2 < num < high_1:
        high_2 = num
    elif num > high_1:
        high_2 = high_1
        high_1 = num
print("The second largest score -",high_2)
```

(d). If any of the scores is greater than 100, then after all the scores have been entered, print a message warning the user that a value over 100 has been entered.

C:\Solutions to tasks\5.9 Exercises\5.9.10.d.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.10.d.py

```
flag = 0
for i in range(10):
    score = eval(input("Enter a test score:  "))
    if score > 100:
        flag = 1
print()
if flag == 1:
    print("The value over 100 has been entered.")
```

(e). Drop the two lowest scores and print out the average of the rest of them.

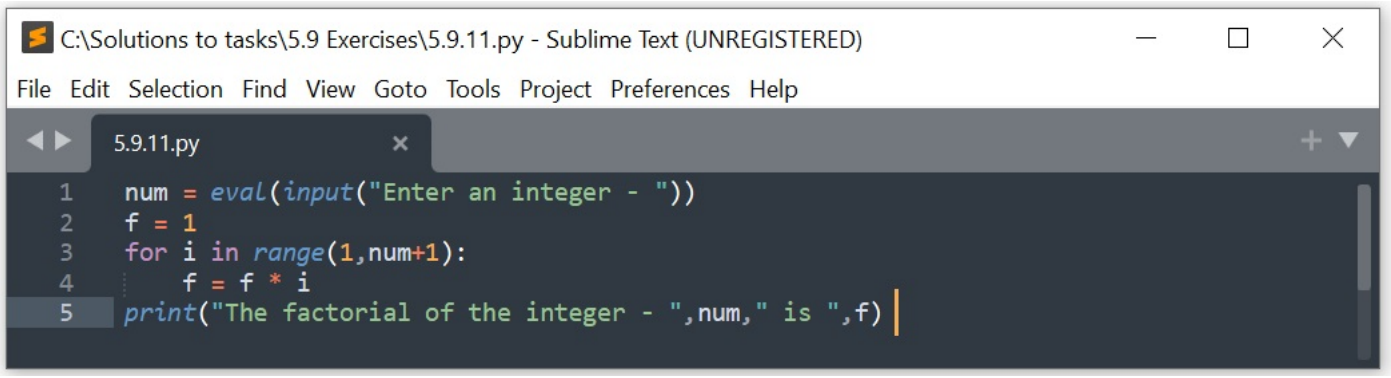
C:\Solutions to tasks\5.9 Exercises\5.9.10.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

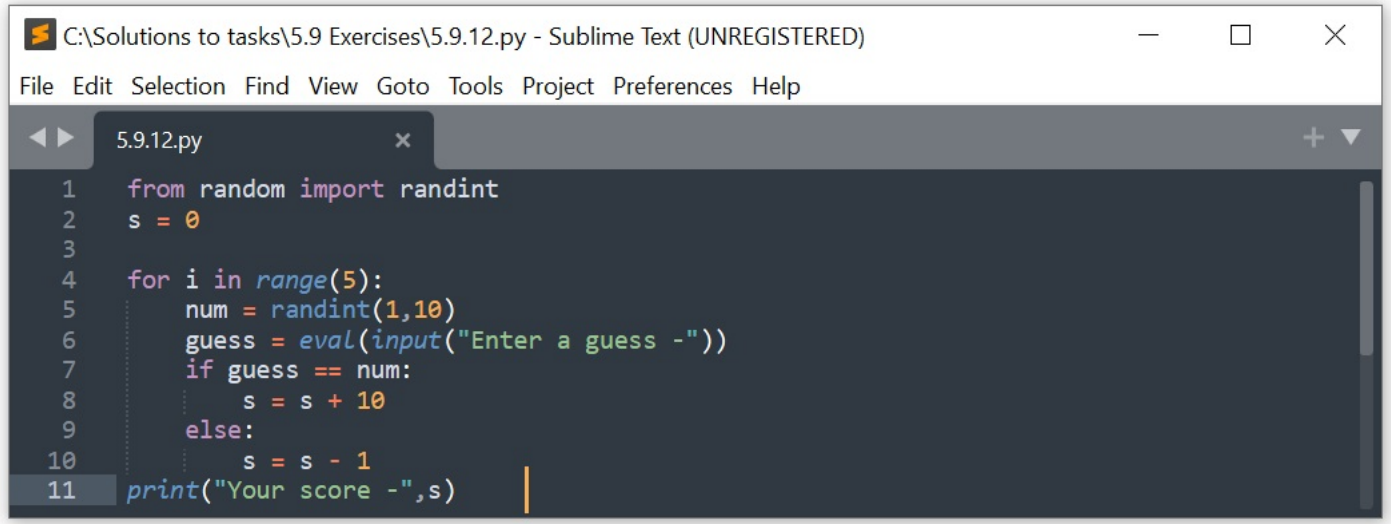
5.9.10.e.py

```
a = eval(input('Enter a test score: '))
b = eval(input('Enter a test score: '))
if a < b:
    low_1 = a
    low_2 = b
else:
    low_1 = b
    low_2 = a
s = a + b
for i in range(8):
    a = eval(input("Enter a test score: "))
    s = s + a
    if low_1 < a < low_2:
        low_2 = a
    elif a < low_1:
        low_2 = low_1
        low_1 = a
print("The average without the two lowest scores - ",(s-low_1-low_2)/8,)
```

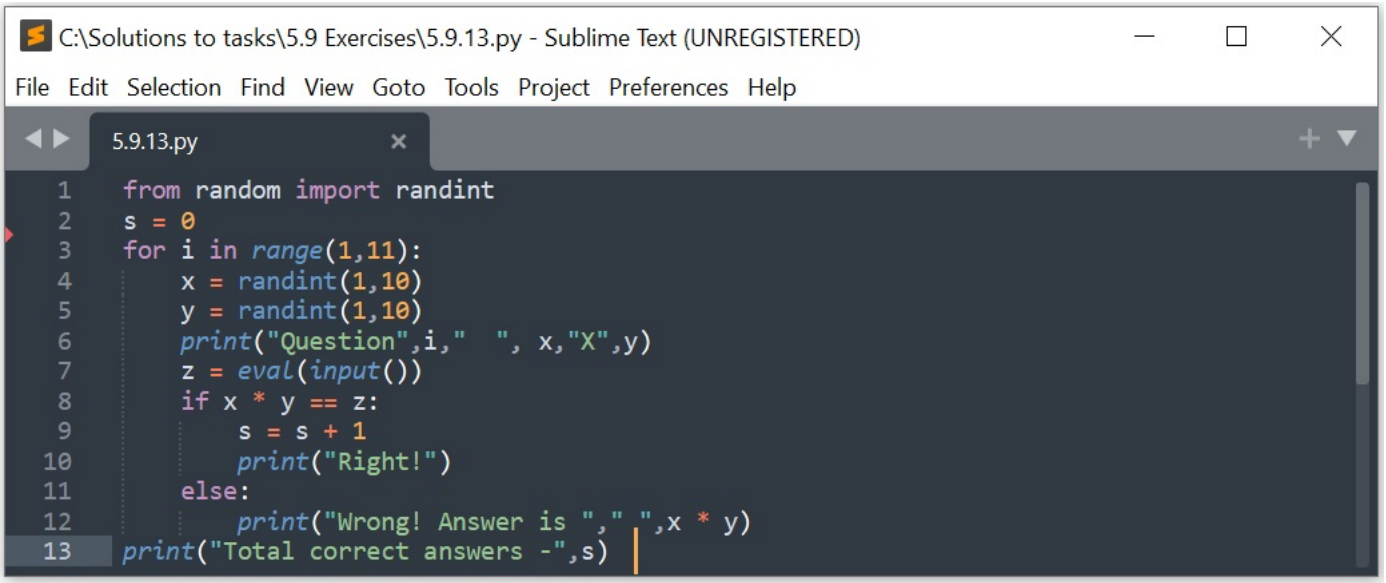
11. Write a program that computes the factorial of a number. The factorial, n, of a number n is the product of all the integers between 1 and n, including n. For instance, $5 = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$. [Hint: Try using a multiplicative equivalent of the summing technique.]



12. Write a program that asks the user to guess a random number between 1 and 10. If they guess right, they get 10 points added to their score, and they lose 1 point for an incorrect guess. Give the user five numbers to guess and print their score after all the guessing is done.



13. In the last chapter there was an exercise that asked you to create a multiplication game for kids. Improve your program from that exercise to keep track of the number of right and wrong answers. At the end of the program, print a message that varies depending on how many questions the player got right.



14. This exercise is about the well-known Monty Hall problem. In the problem, you are a contestant on a game show. The host, Monty Hall, shows you three doors. Behind one of those doors is a prize, and behind the other two doors are goats. You pick a door. Monty Hall, who knows behind which door the prize lies, then opens up one of the doors that doesn't contain the prize. There are now two doors left, and Monty gives you the opportunity to change your choice. Should you keep the same door, change doors, or does it not matter?

(a).Write a program that simulates playing this game 10000 times and calculates what percentage of the time you would win if you switch and what percentage of the time you would win by not switching.

C:\Solutions to tasks\5.9 Exercises\5.9.14.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.14.a.py

```
# Igor Antonenko

from random import randint
win_no_change = 0
win_change = 0
for _ in range(10000):
  car = randint(1,3)
  player = randint(1,3)
  # no change
  if player == car:
    win_no_change = win_no_change + 1
  # change
  elif player != car:
    win_change = win_change + 1
print("Wins by not switching - ",round((win_no_change/10000)*100, 2),"%")
print("Wins by switching - ", round((win_change/10000)*100, 2),"%")
```

(b). Try the above but with four doors instead of three. There is still only one prize, and Monty still opens up one door and then gives you the opportunity to switch.

C:\Solutions to tasks\5.9 Exercises\5.9.14.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

5.9.14.b.py

```
from random import choice

N_doors = int(input("How many doors are (>= 3? "))

wins_change = 0
wins_no_change = 0
doors = list(range(1, N_doors+1))
for K in range(1, 10000):
    host_pick = choice(doors)
    player_pick = choice(doors)
    show = choice([i for i in doors if i != host_pick and i != player_pick])
    if host_pick == player_pick:
      wins_no_change += 1
    player_pick = choice([i for i in doors if i != show and i != player_pick])
    if player_pick == host_pick:
      wins_change += 1
print(f'Wins by not switching {wins_no_change/N_Trials:.2%}')
print(f'Wins by switching {wins_change/N_Trials:.2%}')
```

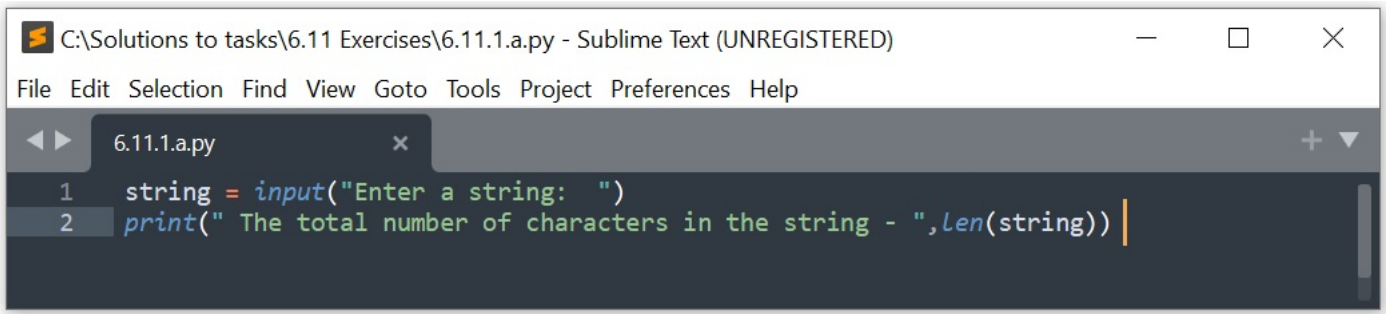
Chapter 6

String

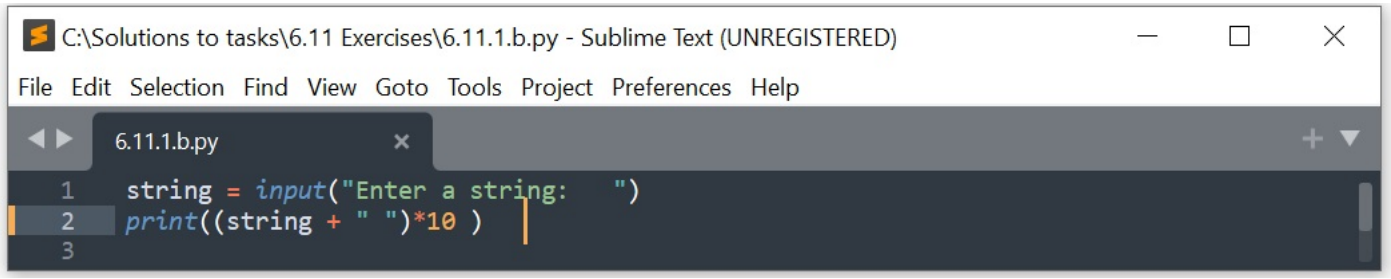
6.1 Exercises 6.11

1. Write a program that asks the user to enter a string. The program should then print the following:

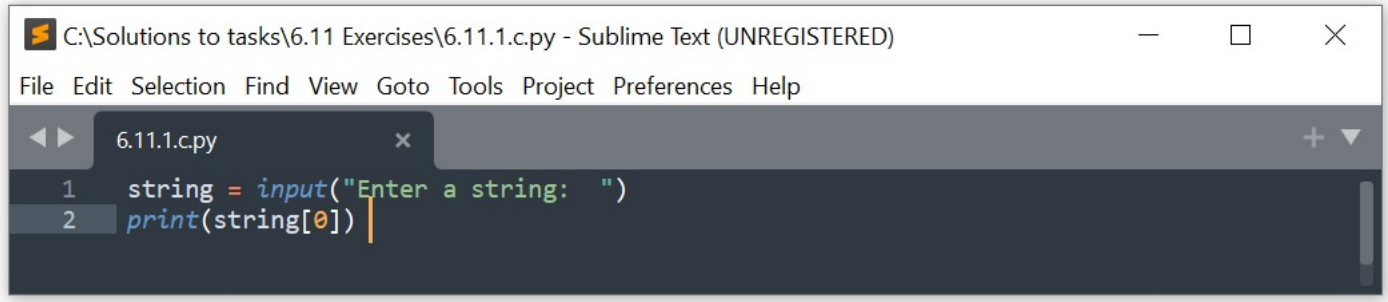
(a). The total number of characters in the string



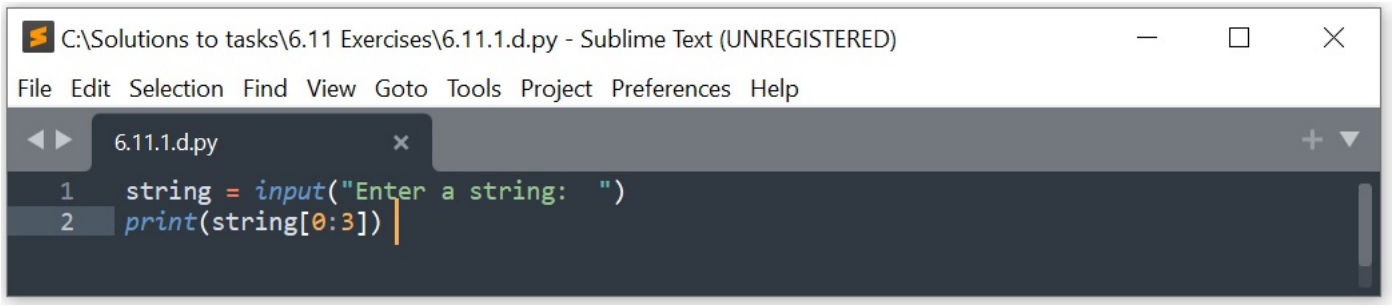
(b). The string repeated 10 times



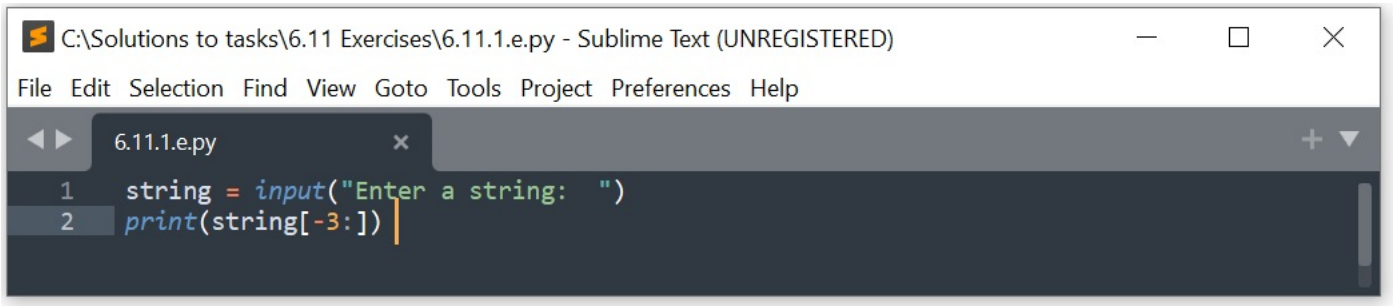
(c). The first character of the string (remember that string indices start at 0)



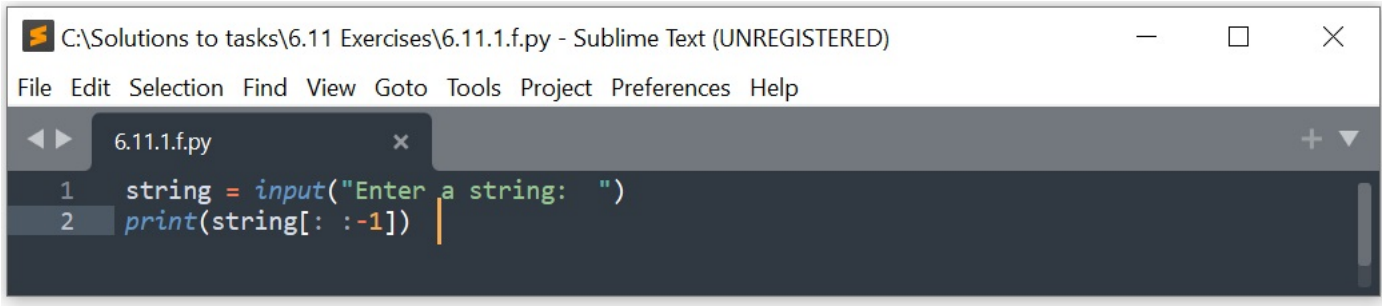
(d). The first three characters of the string



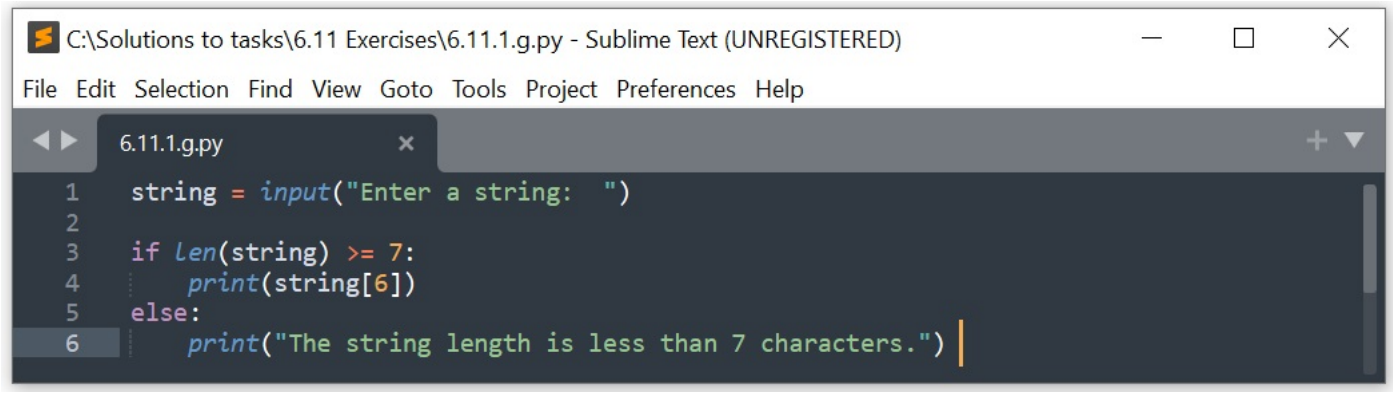
(e).The last three characters of the string



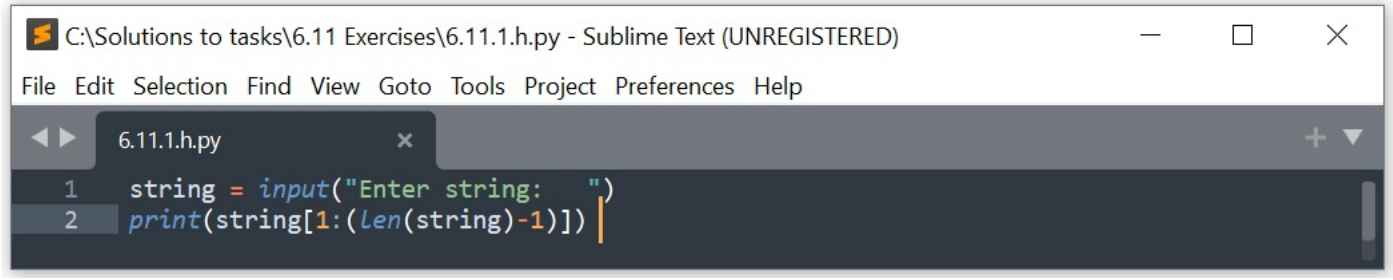
(f). The string backwards



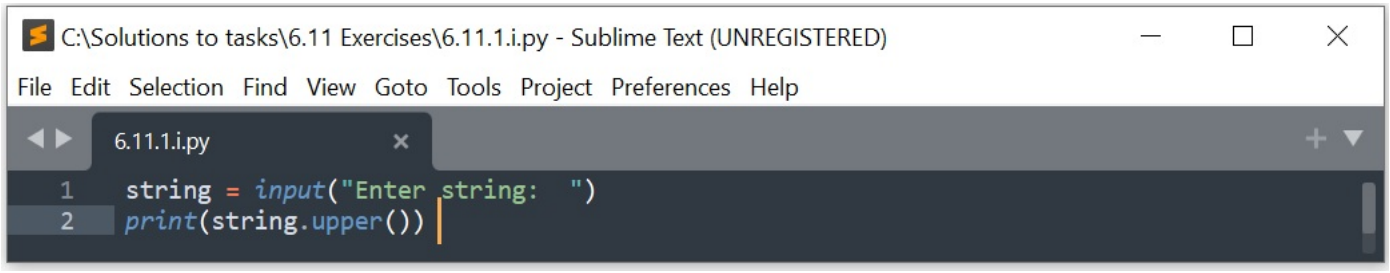
(g). The seventh character of the string if the string is long enough and a message otherwise



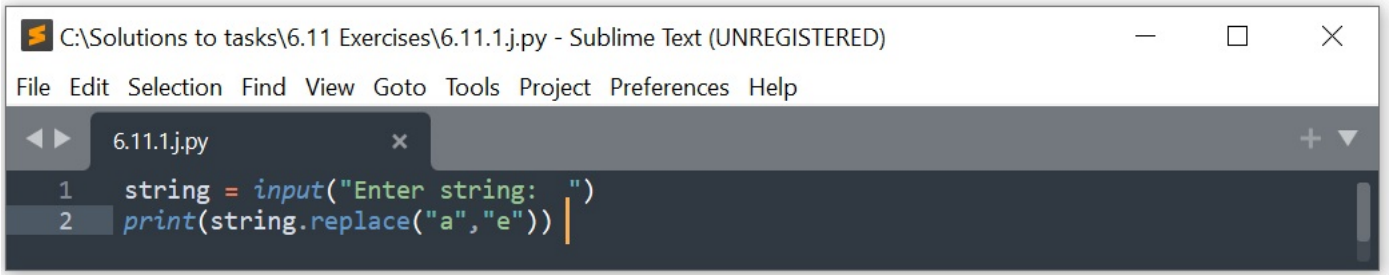
(h).The string with its first and last characters removed



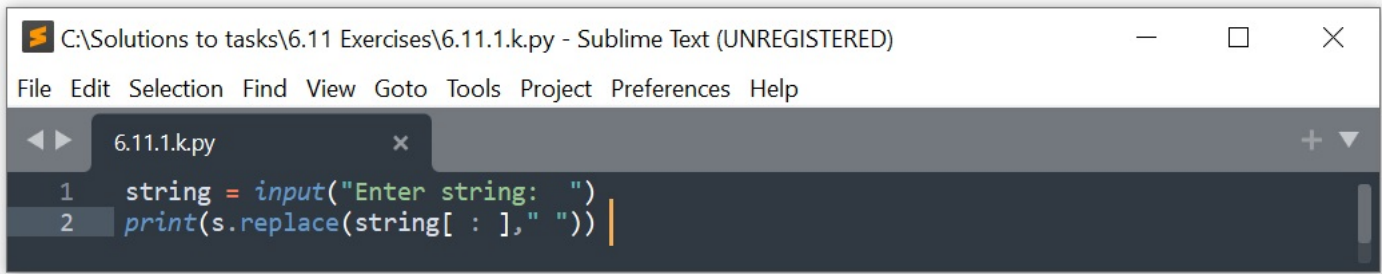
(i). The string in all caps



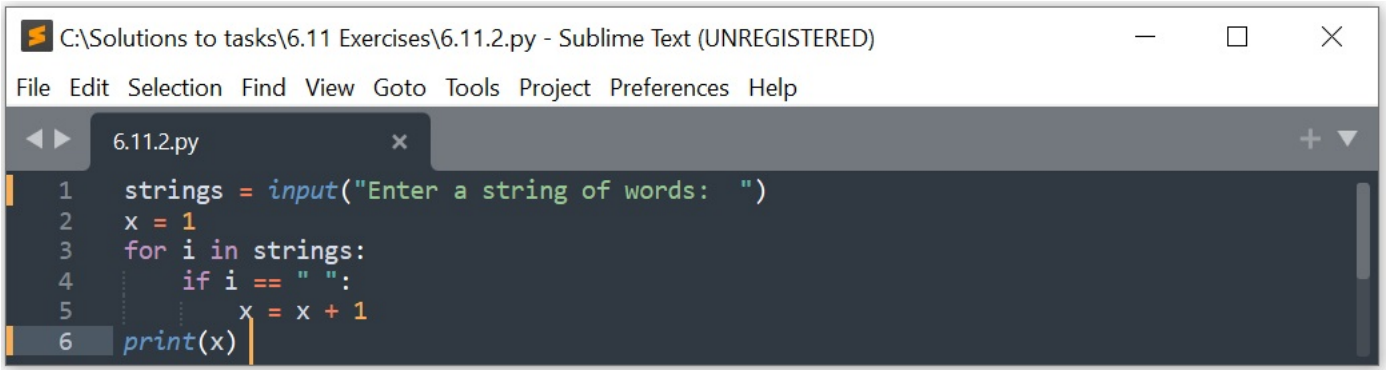
(j). The string with every a replaced with an e.



(k). The string with every letter replaced by a space



2. A simple way to estimate the number of words in a string is to count the number of spaces in the string. Write a program that asks the user for a string and returns an estimate of how many words are in the string.



3. People often forget closing parentheses when entering formulas. Write a program that asks the user to enter a formula and prints out whether the formula has the same number of opening and closing parentheses.

C:\Solutions to tasks\6.11 Exercises\6.11.3.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.3.py

```
formula = input("Enter a formula:   ")
x = 0
y = 0
for i in formula:
    if i == "(":
        x = x + 1
    if i == ")":
        y = y + 1
if x == y:
    print("The formula is right.")
else:
    print("The formula is wrong.")
```

4. Write a program that asks the user to enter a word and prints out whether that word contains any vowels.

C:\Solutions to tasks\6.11 Exercises\6.11.4.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.4.py

```
string = input("Enter a word:  ")
x = " "

for i in string:
    if i in 'aeiou':
        x = x + i
print("The word contains the following vowels -",x)
```

5. Write a program that asks the user to enter a string. The program should create a new string called *new_string* from the user's string such that the second character is changed to an asterisk and three exclamation points are attached to the end of the string. Finally, print *new_string*. Typical output is shown below:

Enter your string: Qbert
Q*ert!!!

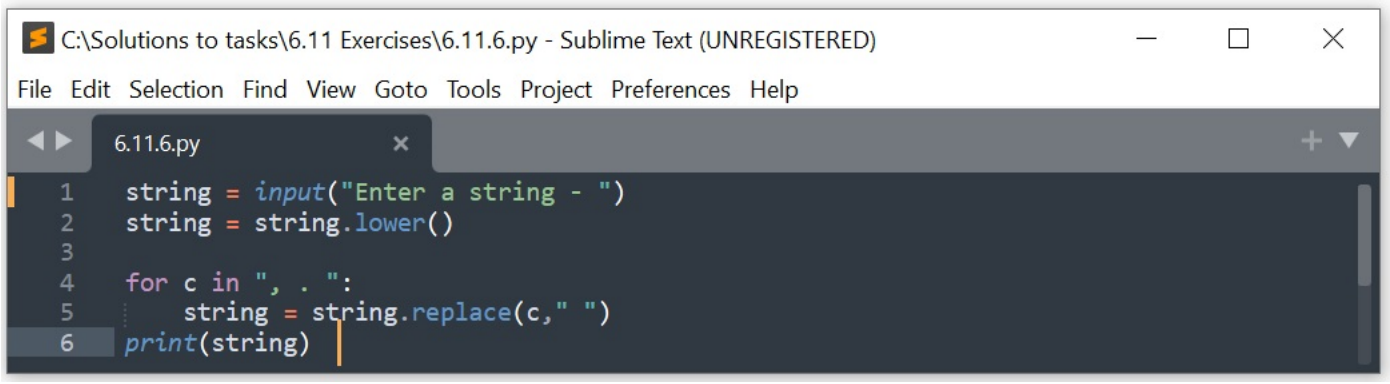
C:\Solutions to tasks\6.11 Exercises\6.11.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

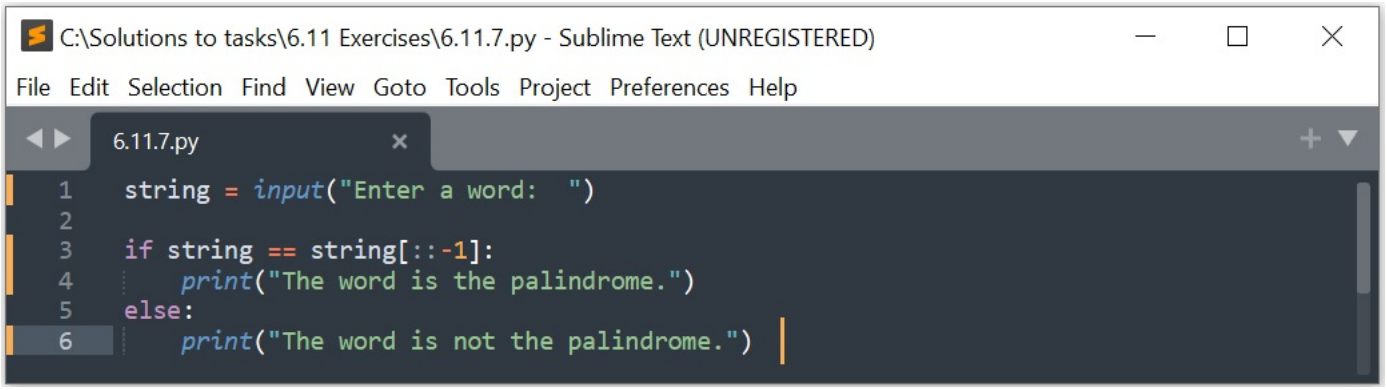
6.11.5.py

```
string = input("Enter a string:   ")

string_new = ""
for i in range(len(string)):
    if i != 1:
        string_new = string_new + string[i]
    else:
        string_new = string_new + "*"
print(string_new + "!!!")
```

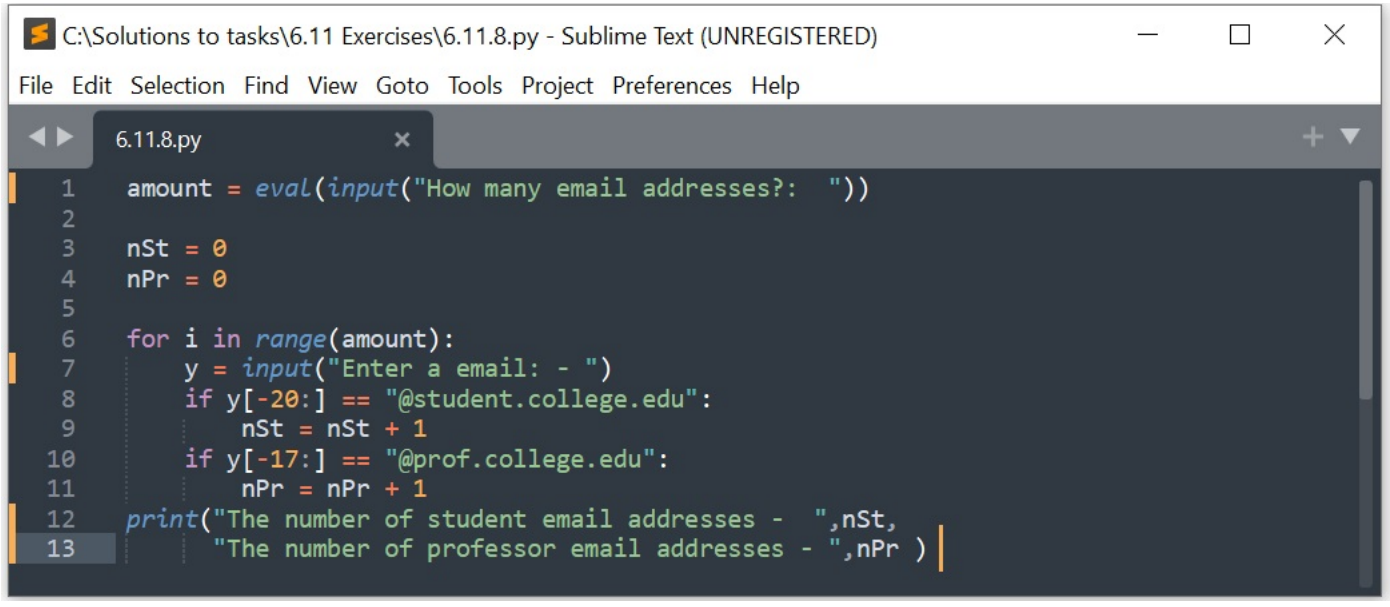
6. Write a program that asks the user to enter a string s and then converts s to lowercase, removes all the periods and commas from s, and prints the resulting string.



7. Write a program that asks the user to enter a word and determines whether the word is a palindrome or not. A palindrome is a word that reads the same backwards as forwards.

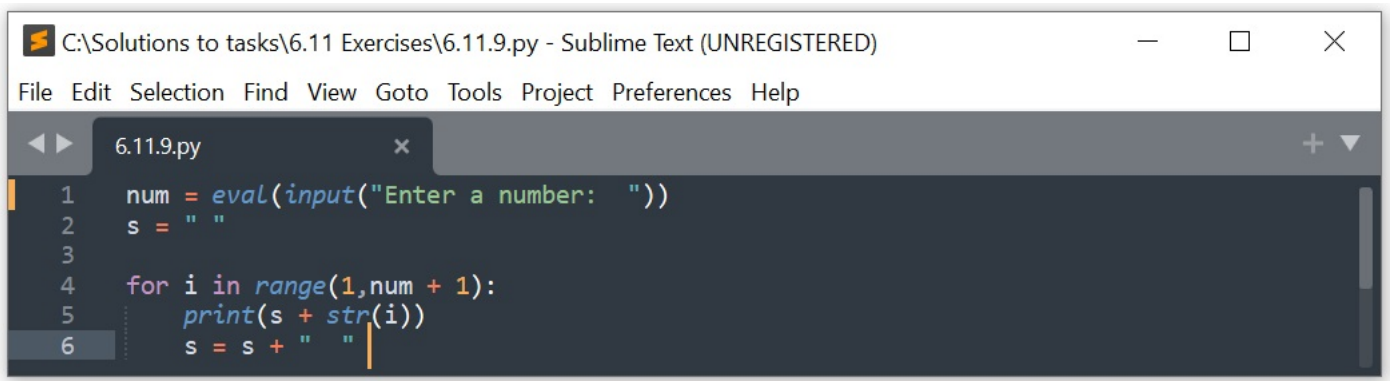


8. At a certain school, student email addresses end with *@student.college.edu*, while professor email addresses end with *@prof.college.edu*. Write a program that first asks the user how many email addresses they will be entering, and then has the user enter those addresses. After all the email addresses are entered, the program should print out a message indicating either that all the addresses are student addresses or that there were some professor addresses entered



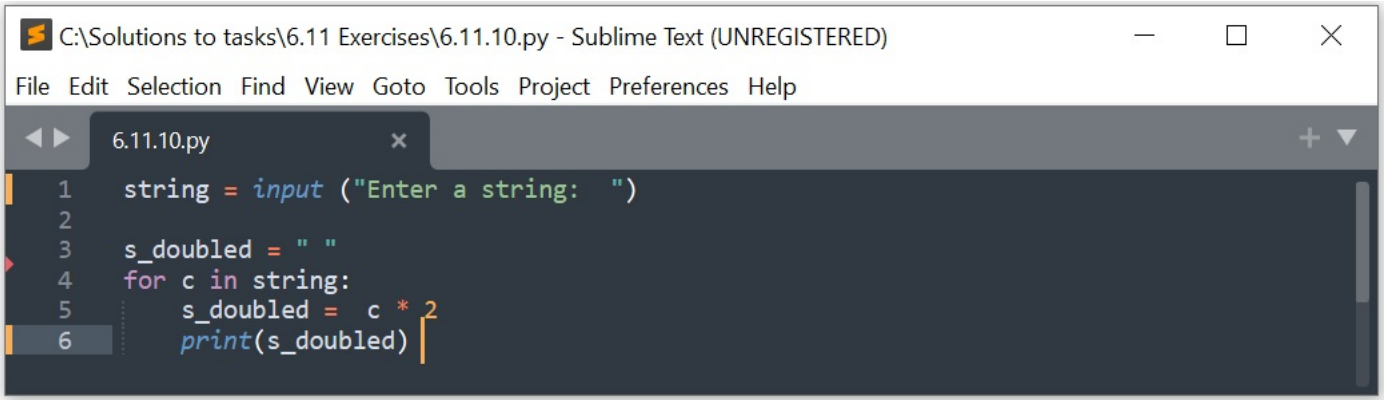
9. Ask the user for a number and then print the following, where the pattern ends at the number that the user enters.

1
2
3
4



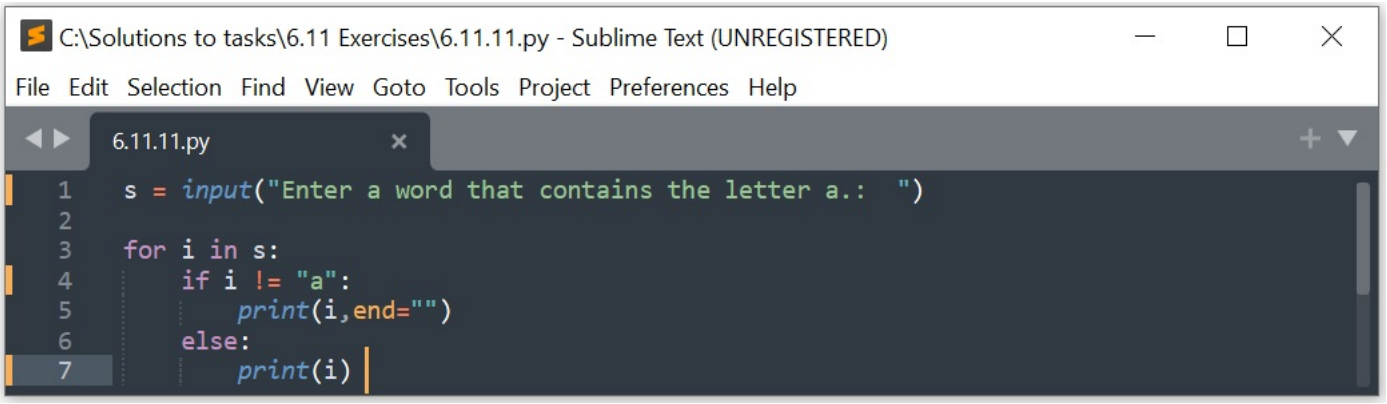
10. Write a program that asks the user to enter a string, then prints out each letter of the string doubled and on a separate line. For instance, if the user entered *HEY*, the output would be

HH
EE
YY



11. Write a program that asks the user to enter a word that contains the letter *a*. The program should then print the following two lines: On the first line should be the part of the string up to and including the first *a*, and on the second line should be the rest of the string. Sample output is shown below:

Enter a word: buffalo
buffa
lo



12.Write a program that asks the user to enter a word and then capitalizes every other letter of that word. So if the user enters *rhinoceros*, the program should print *rHiNoCeRoS*.

C:\Solutions to tasks\6.11 Exercises\6.11.12.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.12.py

```
1  s = input("Enter a word:  ")
2  a = ""
3
4  for i in range(len(s)):
5      if i % 2 != 0:
6          a = a + s[i].upper()
7      else:
8          a = a + s[i]
9  print(a)
```

13. Write a program that asks the user to enter two strings of the same length. The program should then check to see if the strings are of the same length. If they are not, the program should print an appropriate message and exit. If they are of the same length, the program should alternate the characters of the two strings. For example, if the user enters *abcde* and *ABCDE* the program should print out *AaBbCcDdEe*.

C:\Solutions to tasks\6.11 Exercises\6.11.13.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.13.py

```
1  string_1 = input("Enter the first string:  ")
2  string_2 = input("Enter the second string of the same length: ")
3  string = ""
4
5  if len(string_1) != len(string_2):
6      print("Strings of different lengths")
7  else:
8      for i in range(len(string_1)):
9          string = string_1[i] + string_2[i]
10         print(string,end = "")
```

14. Write a program that asks the user to enter their name in lowercase and then capitalizes the first letter of each word of their name.

C:\Solutions to tasks\6.11 Exercises\6.11.14.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.14.py

```
1  # Antonenko Igor
2  name = input("Enter their full name in lowercase:")
3  n =  name[0].upper()
4  for i in range(1, len(name)):
5      if  name[i-1] != " ":
6          n = n + name[i]
7      else:
8          n = n + name[i].upper()
9  print(n)
```

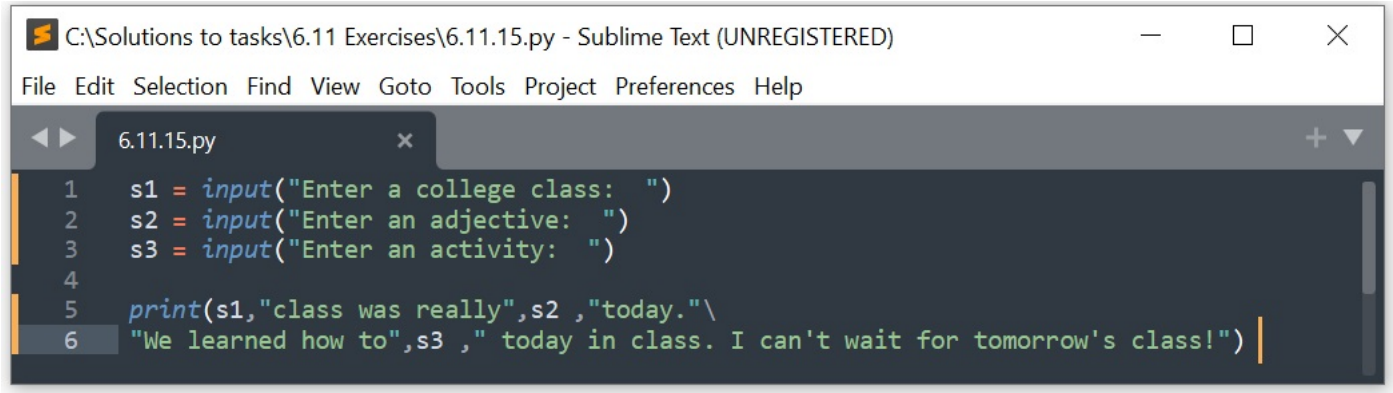
15. When I was a kid, we used to play this game called *Mad Libs*. The way it worked was a friend

would ask me for some words and then insert those words into a story at specific places and read the story. The story would often turn out to be pretty funny with the words I had given since I had no idea what the story was about. The words were usually from a specific category, like a place, an animal, etc.

For this problem you will write a *Mad Libs* program. First, you should make up a story and leave out some words of the story. Your program should ask the user to enter some words and tell them what types of words to enter. Then print the full story along with the inserted words. Here is a small example, but you should use your own (longer) example:

Enter a college class: CALCULUS
Enter an adjective: HAPPY
Enter an activity: PLAY BASKETBALL

CALCULUS class was really HAPPY today. We learned how to
PLAY BASKETBALL today in class. I can't wait for tomorrow's
class!



16.Companies often try to personalize their offers to make them more attractive. One simple way to do this is just to insert the person's name at various places in the offer. Of course, companies don't manually type in every person's name; everything is computer-generated. Write a program that asks the user for their name and then generates an offer like the one below. For simplicity's sake, you may assume that the person's first and last names are one word each.

Enter name: George Washington

Dear George Washington,

I am pleased to offer you our new Platinum Plus Rewards card
at a special introductory APR of 47.99%. George, an offer
like this does not come along every day, so I urge you to call
now toll-free at 1-800-314-1592. We cannot offer such a low
rate for long, George, so call right away.

C:\Solutions to tasks\6.11 Exercises\6.11.16.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.16.py

```
s = input("Enter their name: ")
s1 = " "
for i in s:
    if i !=" ":
        s1 = s1 +i
    if i == " ":
        print("Dear",s,",")
        print()
        print(
"I am pleased to offer you our new Platinum Plus Rewards card at a special"
" introductory APR of 47.99%.",s1,", an offer like this does not come along every"
" day, so I urge you to call now toll-free at  1-800-314-1592. We cannot offer"
" such a low rate for long,", s1,", so call right away.")
```

17. Write a program that generates the 26-line block of letters partially shown below. Use a loop containing one or two **print** statements.

abcdefghijklmnopqrstuvwxyz
bcdefghijklmnopqrstuvwxyza
cdefghijklmnopqrstuvwxyzab
. . .
yzabcdefghijklmnopqrstuvwx
zabcdefghijklmnopqrstuvwxy

C:\Solutions to tasks\6.11 Exercises\6.11.17.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.17.py

```
s = "abcdefghijklmnopqrstuvwxyz"

for i in range(len(s)):
    print(s[i: : ] + s[ :i: ],end = " ")
    print()
```

18. The goal of this exercise is to see if you can mimic the behavior of the in operator and the count and index methods using only variables, for loops, and if statements.

(a) Without using the in operator, write a program that asks the user for a string and a letter and prints out whether or not the letter appears in the string.

C:\Solutions to tasks\6.11 Exercises\6.11.18.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.18.a.py

```
string = input("Enter a string:   ")

for i in range(len(string)):
    letter = input("Enter a letter:  ")
    if string[i] == letter:
        print("The letter is in the word.")
    else:
        print("The letter is not in the word.")
```

(b) Without using the *count* method, write a program that asks the user for a string and a letter and counts how many occurrences there are of the letter in the string.

C:\Solutions to tasks\6.11 Exercises\6.11.18.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.18.b.py

```
1  string = input("Enter a string:   ")
2  letter = input("Enter a letter:   ")
3
4  x = 0
5  for i in range(len(string)):
6      if string[i] == letter:
7          x = x + 1
8  print("The letter ",letter,"occurs in the string ",x,"time(s).")
```

(c) Without using the *index* method, write a program that asks the user for a string and a letter and prints out the index of the first occurrence of the letter in the string. If the letter is not in the string, the program should say so.

C:\Solutions to tasks\6.11 Exercises\6.11.18.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

6.11.18.c.py

```
1  string = input("Enter a string:   ")
2  letter = input("Enter a letter:    ")
3
4  index = " "
5  for i in range(len(string)):
6      if string[i] == letter:
7          index = index + str(i)
8  if index == " ":
9      print("The letter is not in the string.")
10 else:
11     print("The index of the first occurrence of the letter - ", index[1])
```

19. Write a program that asks the user for a large integer and inserts commas into it according to the standard American convention for commas in large numbers. For instance, if the user enters *1000000*, the output should be *1,000,000*.

C:\Solutions to tasks\6.11 Exercises\6.11.19.py - Sublime Text (UNREGISTERED)

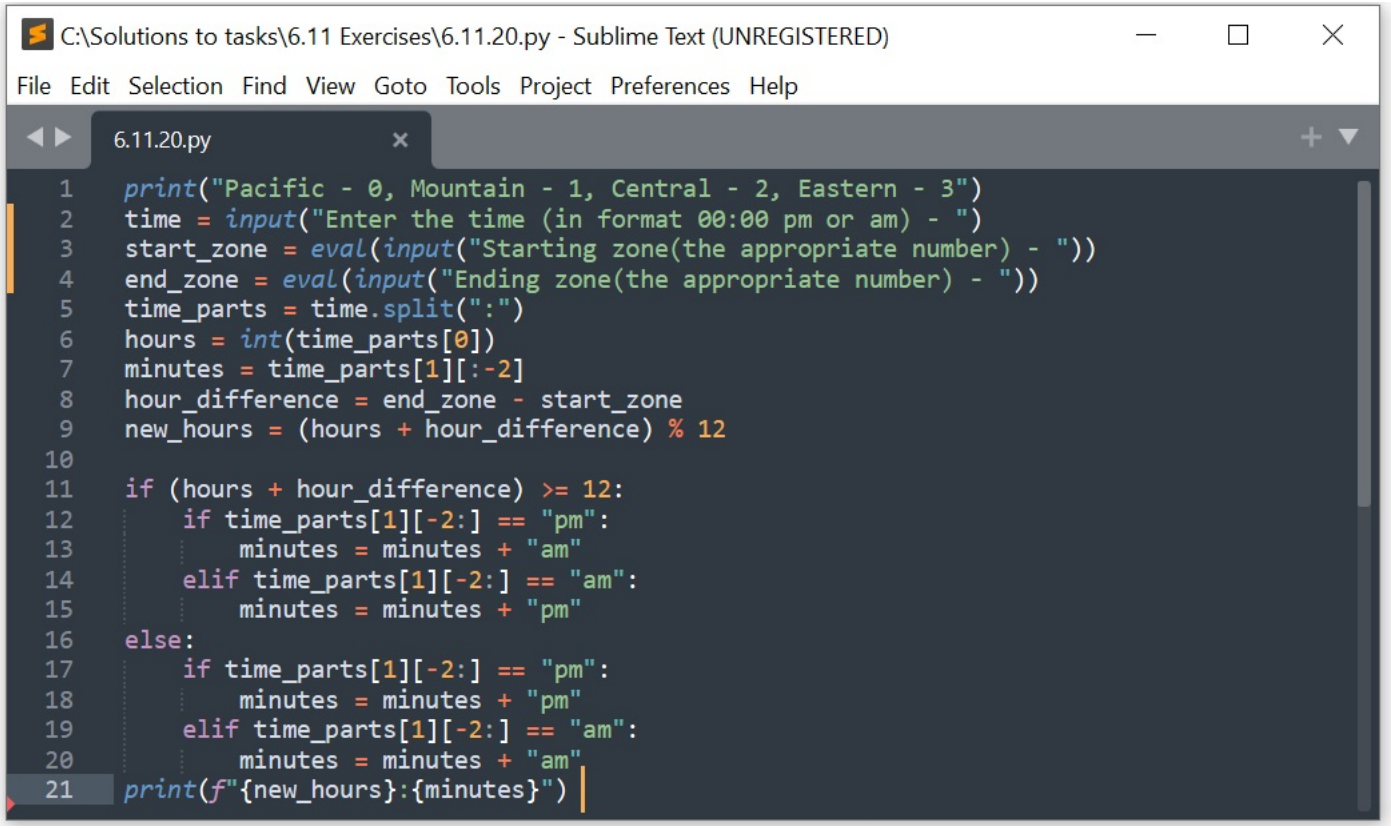
File Edit Selection Find View Goto Tools Project Preferences Help

6.11.19.py

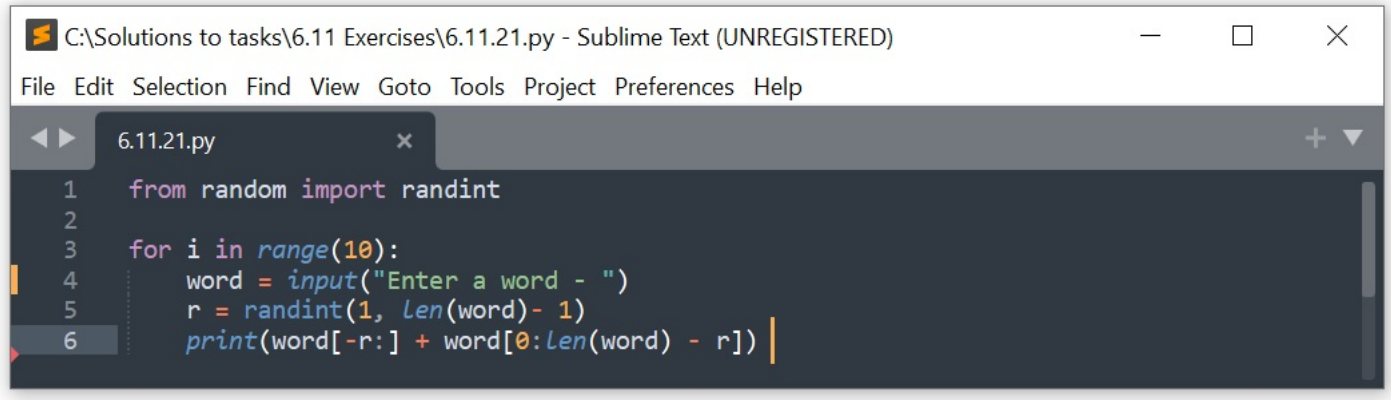
```
1  num = input("Enter a large integer:  ")
2  numreverse = (n[: :-1])
3  c = 0
4  s = " "
5  for i in range(len(nreverse)):
6      c = c + 1
7      if c % 3 == 0:
8          s = s + numreverse[i] + ","
9      else:
10         s  = s + numreverse[i]
11 num = s[: :-1]
12 if num[0] == ",":
13     print(num[1:])
14 else:
15     print(num)
```

20. Write a program that converts a time from one time zone to another. The user enters the time in the usual American way, such as *3:48pm* or *11:26am*. The first time zone the user enters is that of the original time and the second is the desired time zone. The possible time zones are *Eastern*, *Central*, *Mountain*, or *Pacific*.

Time: 11:48pm
Starting zone: Pacific
Ending zone: Eastern
2:48am



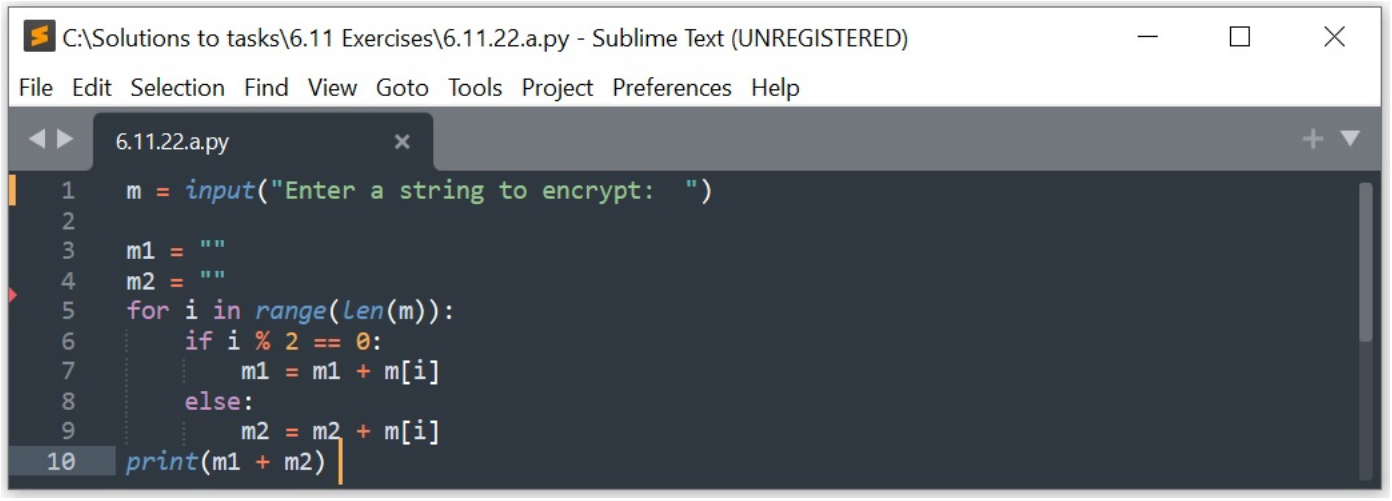
21.An anagram of a word is a word that is created by rearranging the letters of the original. For instance, two anagrams of *idle* are *deli* and *lied*. Finding anagrams that are real words is beyond our reach until Chapter 12. Instead, write a program that asks the user for a string and returns a random anagram of the string—in other words, a random rearrangement of the letters of that string.



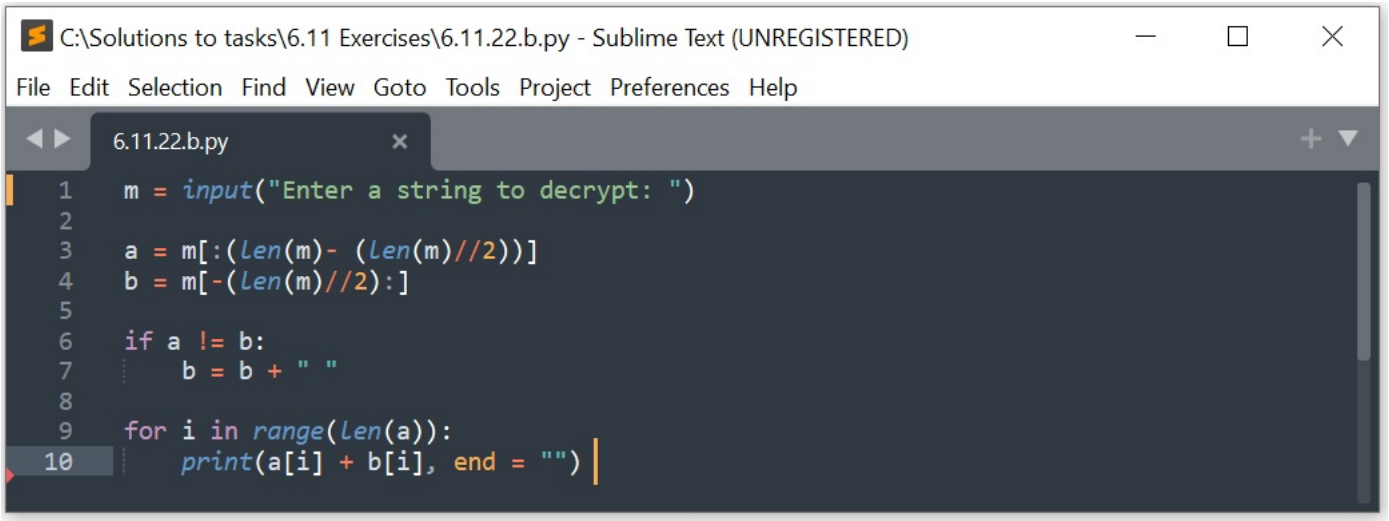
22. A simple way of encrypting a message is to rearrange its characters. One way to rearrange the characters is to pick out the characters at even indices, put them first in the encrypted string, and follow them by the odd characters. For example, the string *message* would be encrypted as *msaeesg* because the even characters are *m*, *s*, *a*, *e* (at indices 0, 2, 4, and 6) and the odd characters are *e*, *s*, *g*

(at indices 1, 3, and 5).

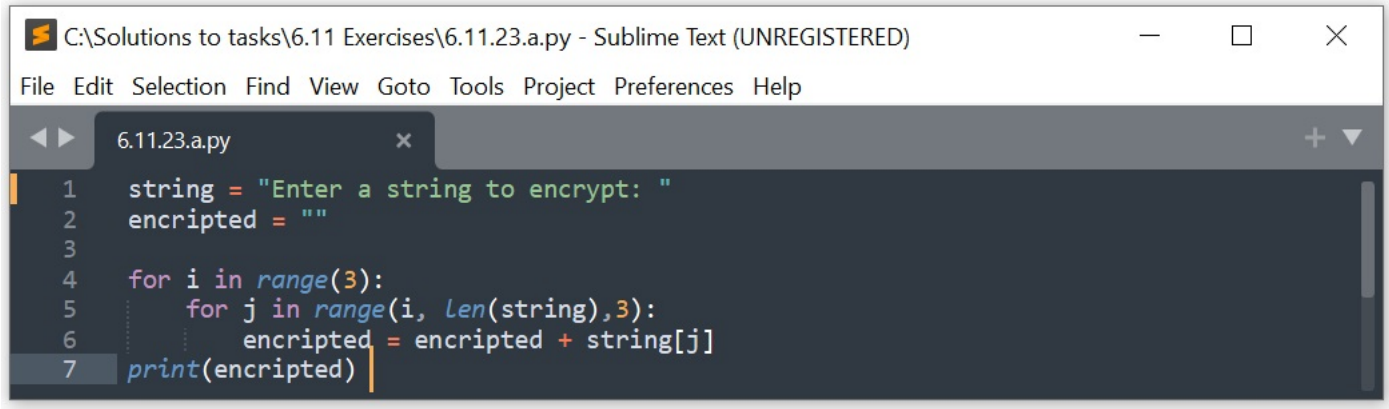
(a) Write a program that asks the user for a string and uses this method to encrypt the string.



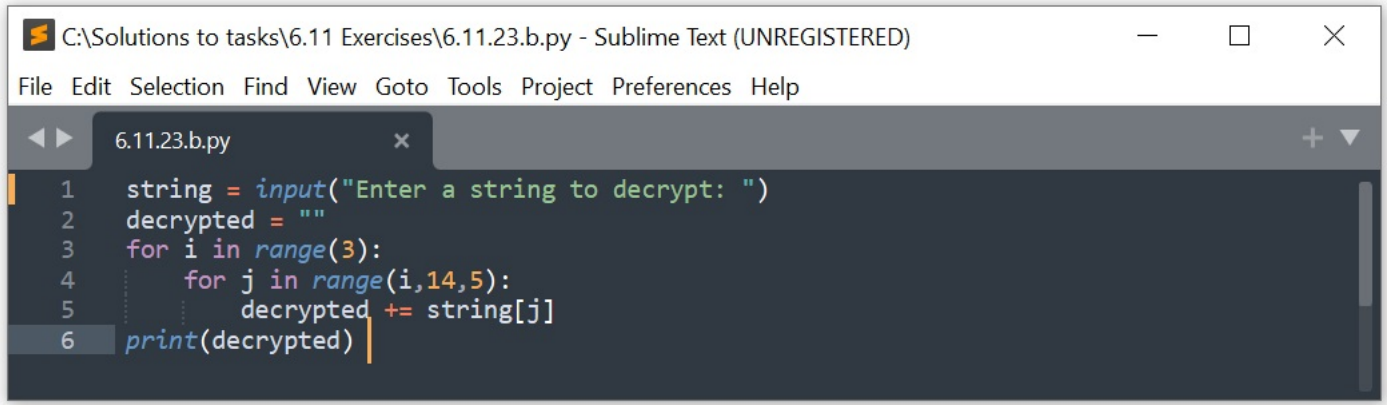
(b) Write a program that decrypts a string that was encrypted with this method.



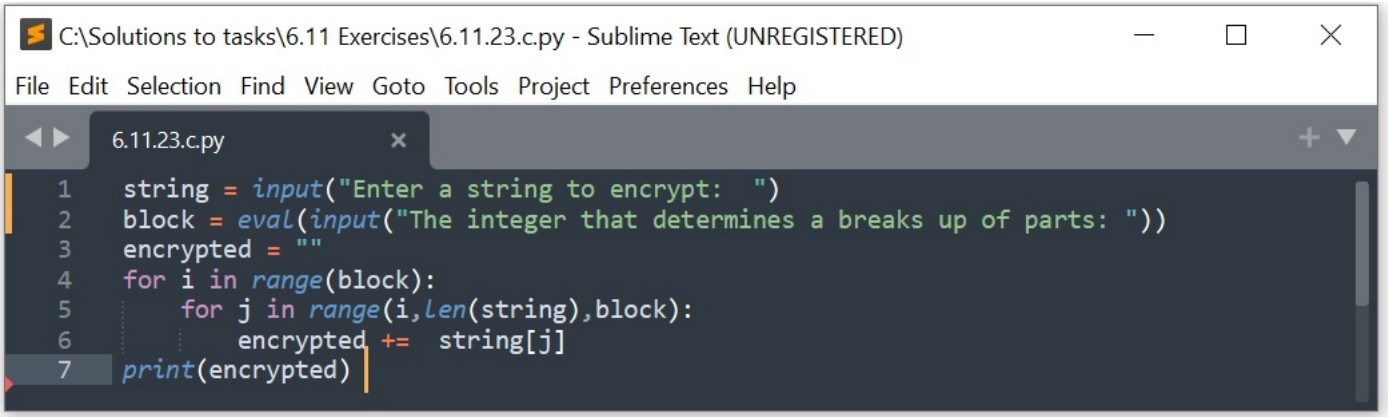
23. A more general version of the above technique is the *rail fence cipher*, where instead of breaking things into evens and odds, they are broken up by threes, fours or something larger. For instance, in the case of threes, the string *secret message* would be broken into three groups. The first group is *sr sg*, the characters at indices 0, 3, 6, 9 and 12. The second group is *eemse*, the characters at indices 1, 4, 7, 10, and 13. The last group is *ctea*, the characters at indices 2, 5, 8, and 11. The encrypted message is *sr sgeemsectea*.
(a). Write a program the asks the user for a string and uses the rail fence cipher in the threes case to encrypt the string.



(b). Write a decryption program for the threes case.

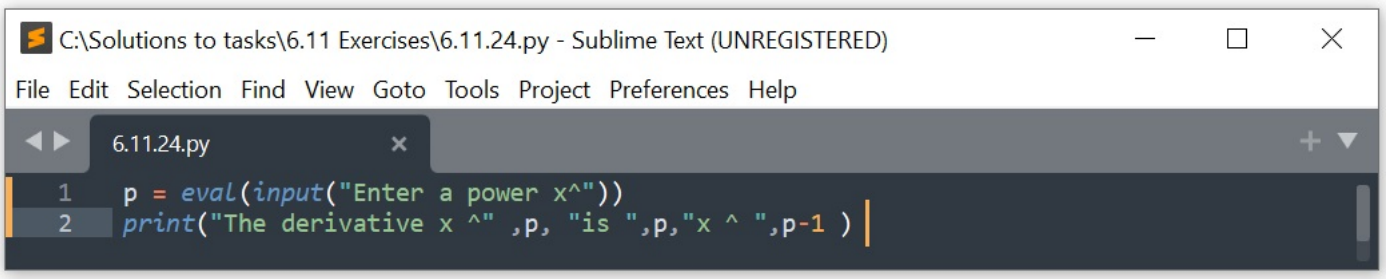


(c). Write a program that asks the user for a string, and an integer determining whether to break things up by threes, fours, or whatever. Encrypt the string using the rail-fence cipher.

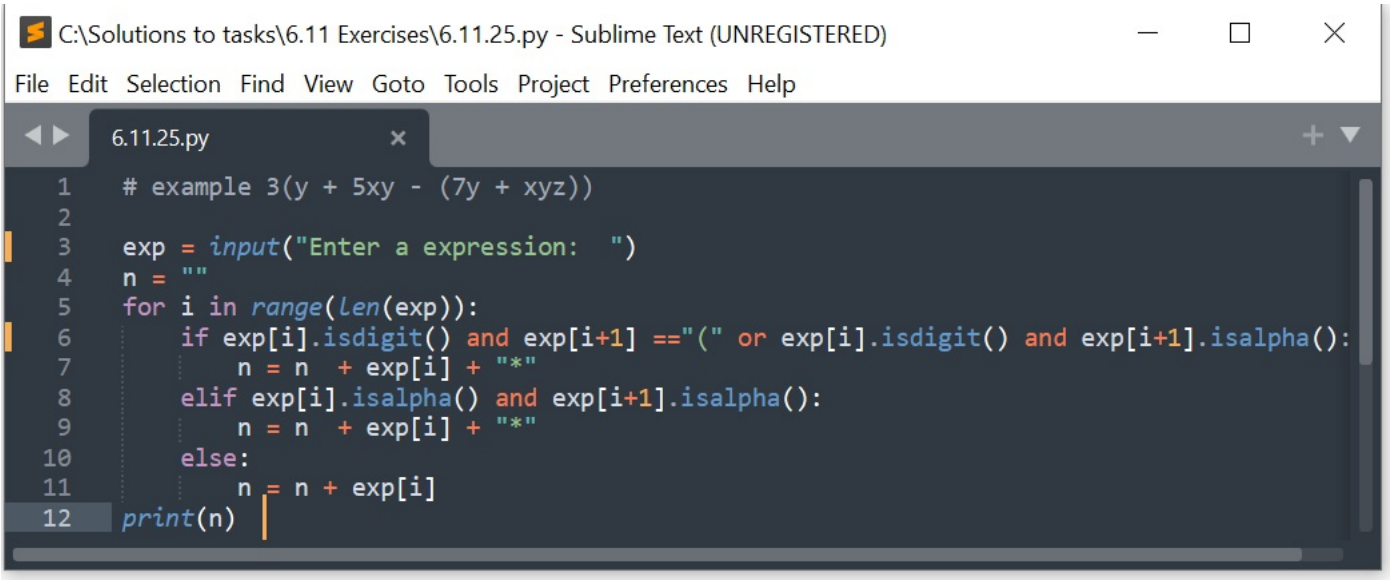


(d). Write a decryption program for the general case.

24. In calculus, the derivative of x^4 is $4x^3$. The derivative of x^5 is $5x^4$. The derivative of x^6 is $6x^5$. This pattern continues. Write a program that asks the user for input like x^3 or x^{25} and prints the derivative. For example, if the user enters x^3, , the program should print out $3x^2$.



25. In algebraic expressions, the symbol for multiplication is often left out, as in $3x + 4y$ or $3(x + 5)$. Computers prefer those expressions to include the multiplication symbol, like $3 * x + 4 * y$ or $3 * (x + 5)$. Write a program that asks the user for an algebraic expression and then inserts multiplication symbols where appropriate.


C:\Solutions to tasks\6.11 Exercises\6.11.25.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
6.11.25.py
# example 3(y + 5xy - (7y + xyz))
exp = input("Enter a expression: ")
n = ""
for i in range(len(exp)):
if exp[i].isdigit() and exp[i+1] =="(" or exp[i].isdigit() and exp[i+1].isalpha():
n = n + exp[i] + "*"
elif exp[i].isalpha() and exp[i+1].isalpha():
n = n + exp[i] + "*"
else:
n = n + exp[i]
print(n)

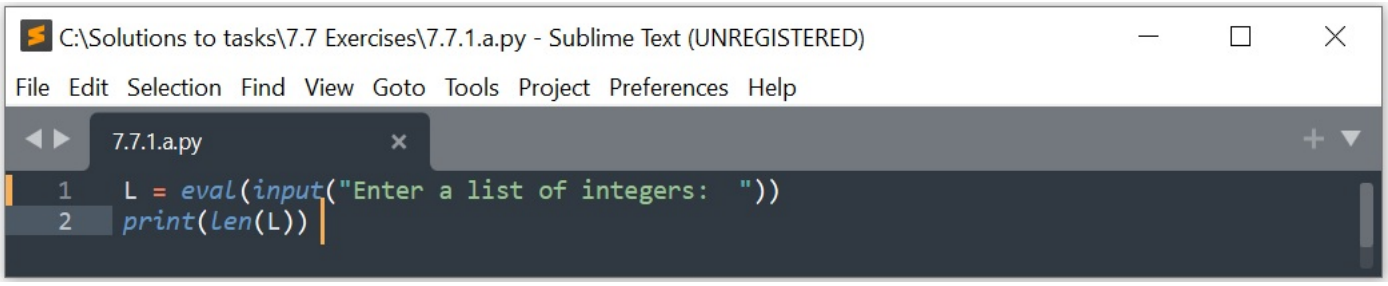
Chapter 7

Lists

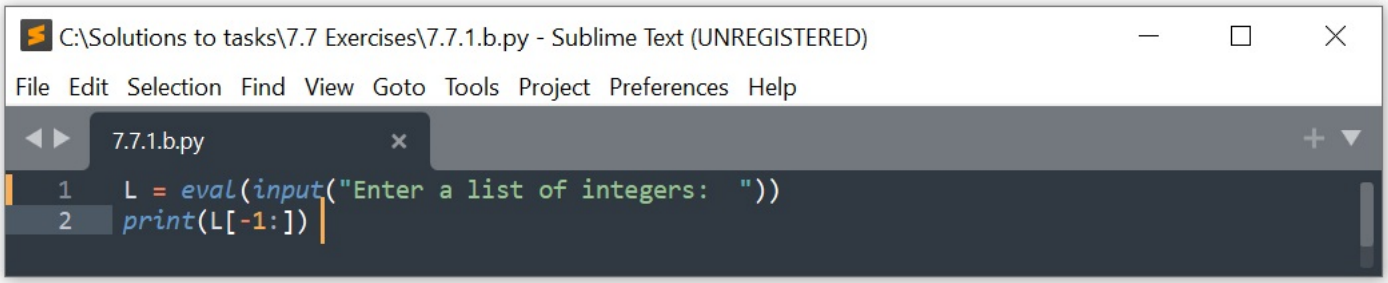
7.1 Exercises 7.7

1. Write a program that asks the user to enter a list of integers. Do the following:

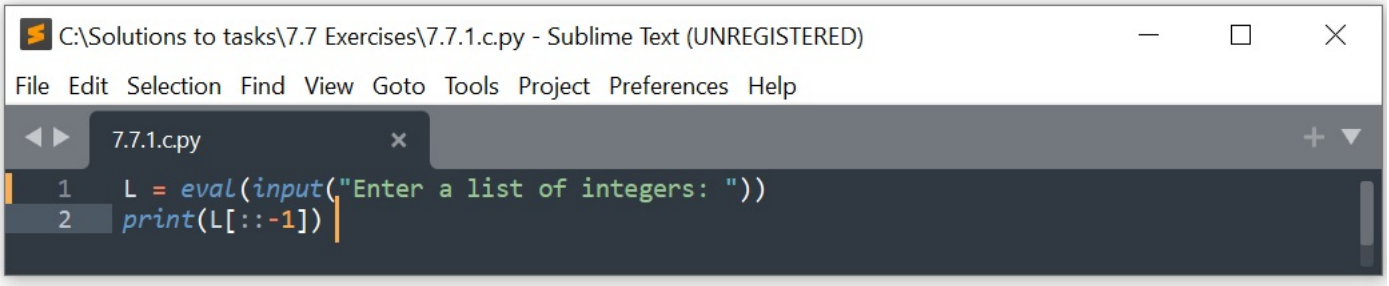
(a) Print the total number of items in the list.



(b) Print the last item in the list.



(c) Print the list in reverse order



(d) Print *Yes* if the list contains a 5 and *No* otherwise.

C:\Solutions to tasks\7.7 Exercises\7.7.1.d.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.1.d.py

```
1  L = eval(input("Enter a list of integers: "))
2  flag = 0
3
4  for i in L:
5      if i == 5:
6          flag = 1
7  if flag == 1:
8      print("Yes")
9  else:
10     print("No")
```

(e) Print the number of fives in the list.

C:\Solutions to tasks\7.7 Exercises\7.7.1.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.1.e.py

```
1  L = eval(input("Enter a list of integers: "))
2
3  c = 0
4  for i in L:
5      if i == 5:
6          c = c + 1
7  print("The number of fives is ",c)
```

(f) Remove the first and last items from the list, sort the remaining items, and print the result.

C:\Solutions to tasks\7.7 Exercises\7.7.1.f.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.1.f.py

```
1  L = eval(input("Enter a list of integers: "))
2
3  del L[0]
4  del L[-1:]
5  L.sort()
6  print(L)
```

(g) Print how many integers in the list are less than 5.

C:\Solutions to tasks\7.7 Exercises\7.7.1.g.py - Sublime Text (UNREGISTERED)

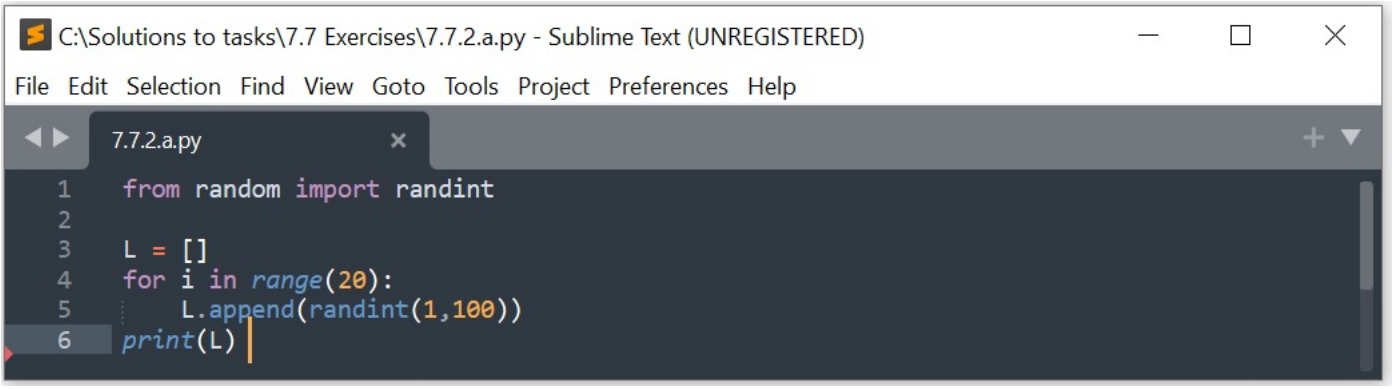
File Edit Selection Find View Goto Tools Project Preferences Help

7.7.1.g.py

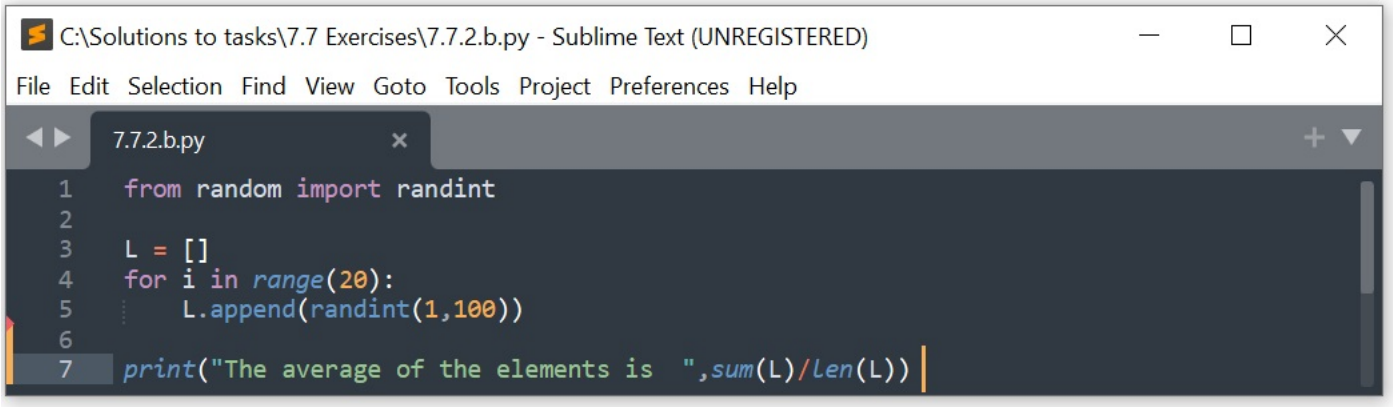
```
1  L = eval(input("Enter a list of integers: "))
2  c = 0
3  for i in L:
4      if i < 5:
5          c = c + 1
6  print("The number of integers less than 5 is ",c)
```

2. Write a program that generates a list of 20 random numbers between 1 and 100.

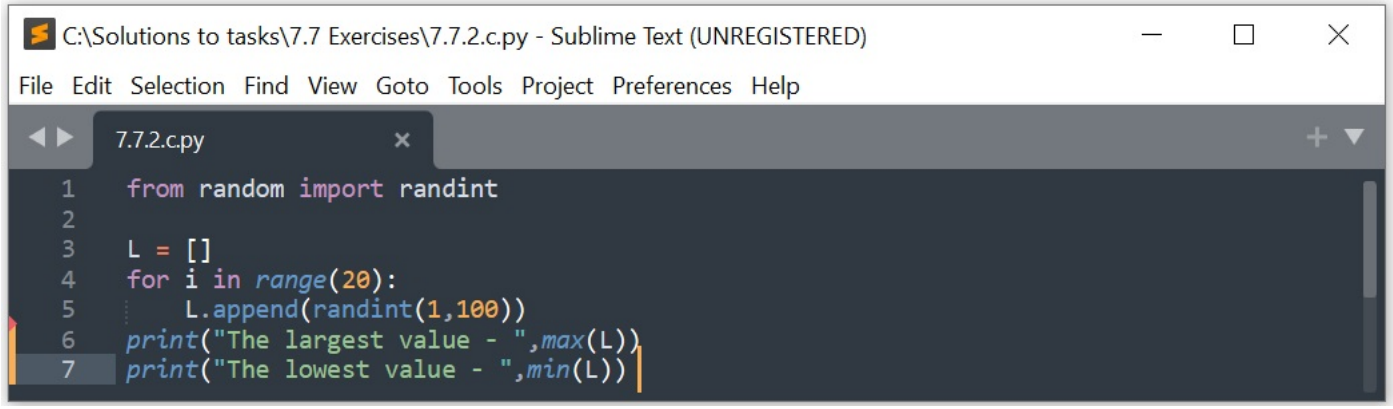
(a) Print the list.



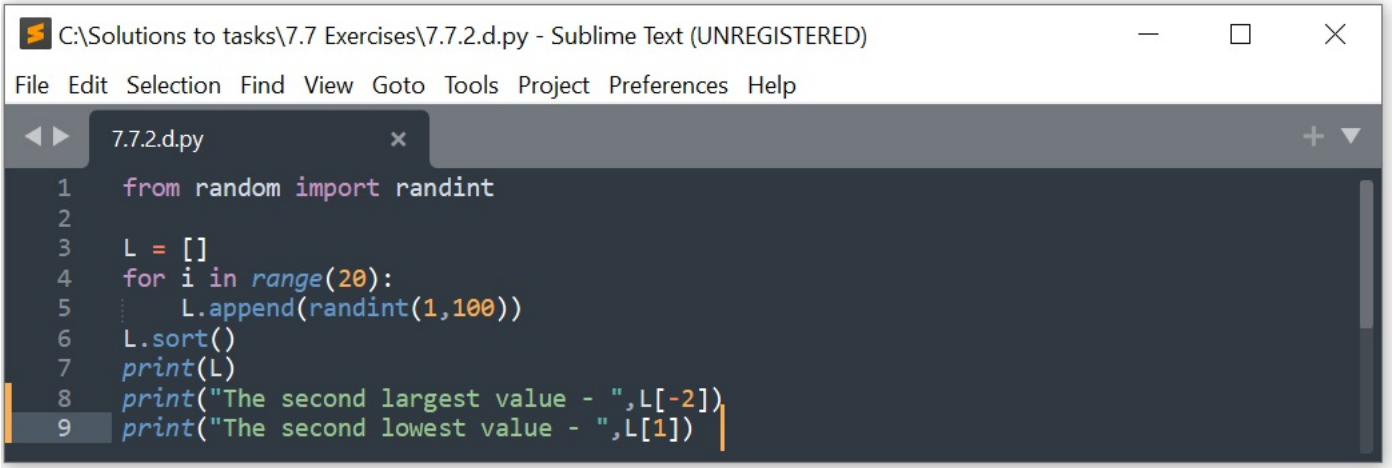
(b) Print the average of the elements in the list.



(c) Print the largest and smallest values in the list.



(d) Print the second largest and second smallest entries in the list



(e) Print how many even numbers are in the list.

C:\Solutions to tasks\7.7 Exercises\7.7.2.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.2.e.py

```
1  from random import randint
2
3  L= []
4  for i in range(20):
5      L.append(randint(1,100))
6  c = 0
7  for i in L:
8      if i % 2 == 0:
9          c = c + 1
10 print("The number of even numbers is ",c)
```

3. Start with the list $[8, 9, 10]$. Do the following:

(a). Set the second entry (index 1) to 17
(b). Add 4, 5, and 6 to the end of the list.
(c). Remove the first entry from the list.
(d). Sort the list.
(e). Double the list.
(f). Insert 25 at index 3.
The final list should equal $[4, 5, 6, 25, 10, 17, 4, 5, 6, 10, 17]$

C:\Solutions to tasks\7.7 Exercises\7.7.3.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.3.py

```
1  L = [8,9,10]
2  L[1] = 17
3  L.append(4)
4  L.append(5)
5  L.append(6)
6  del L[0]
7  L.sort()
8  M = L*2
9  M.insert(3,25)
10 print(M)
```

4. Ask the user to enter a list containing numbers between 1 and 12. Then replace all of the entries in the list that are greater than 10 with 10.

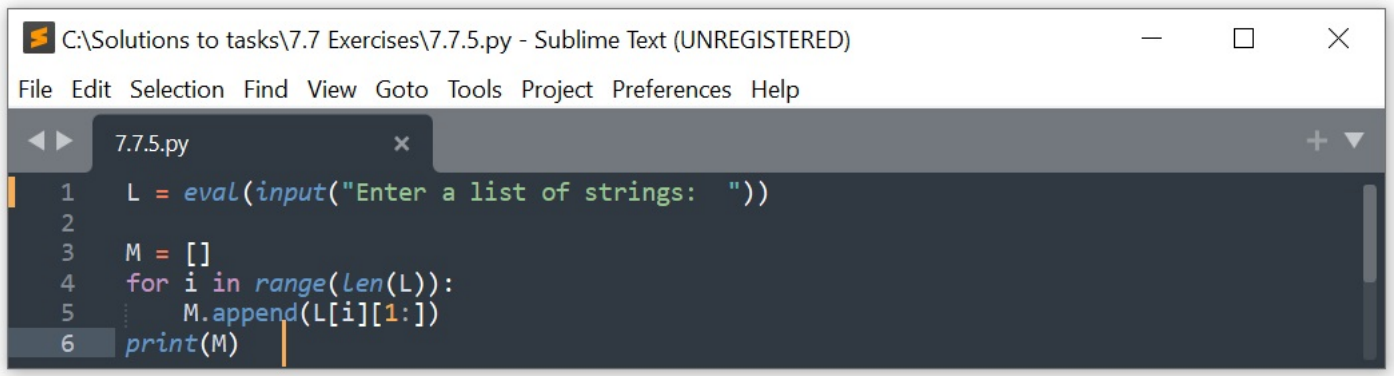
C:\Solutions to tasks\7.7 Exercises\7.7.4.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.4.py

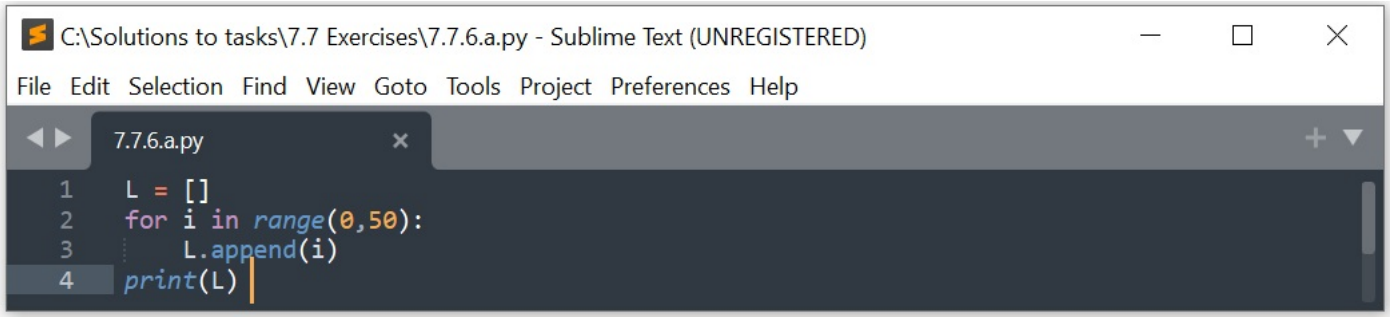
```
1  L = eval(input("Enter a list containing numbers between 1 and 12: "))
2
3  M = []
4  for i in L:
5      if i > 10:
6          i = 10
7      M.append(i)
8  print(M)
```

5.Ask the user to enter a list of strings. Create a new list that consists of those strings with their first characters removed.

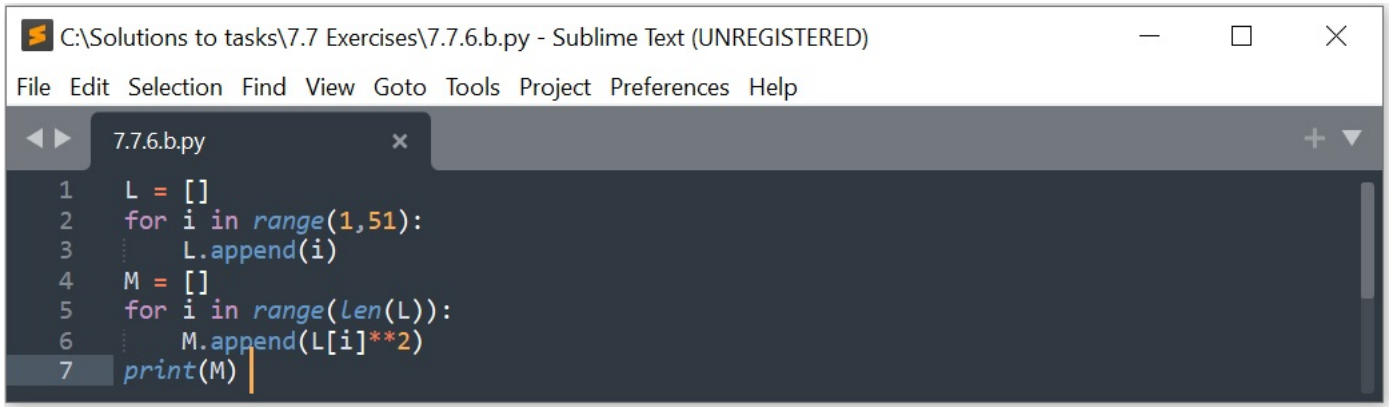


6. Create the following lists using a for loop.

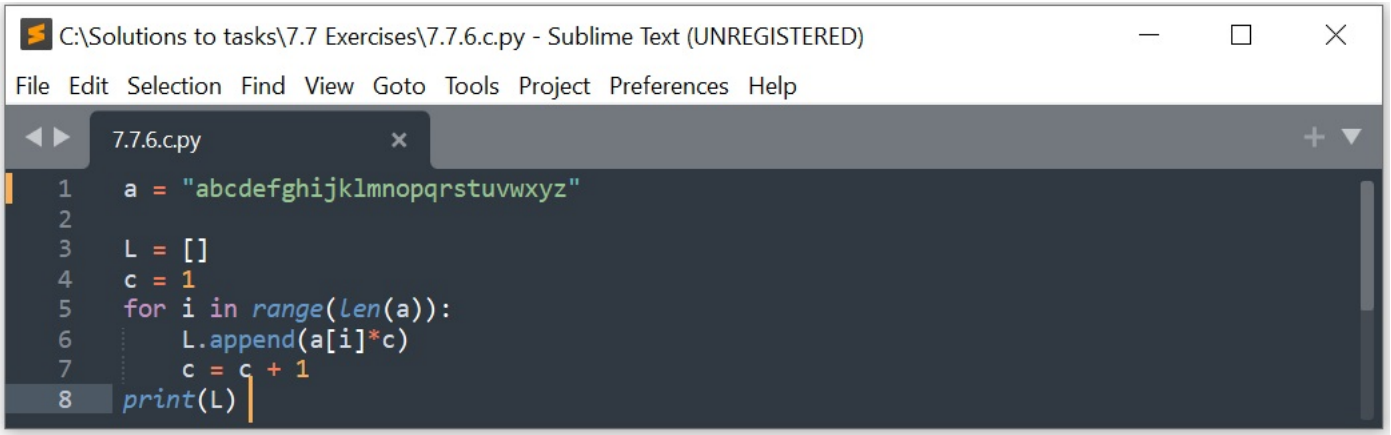
(a) A list consisting of the integers 0 through 49.



(b) A list containing the squares of the integers 1 through 50.



(c) The list ['a','bb','ccc','dddd', . . .], that ends with 26 copies of the letter z.



7. Write a program that takes any two lists L and M of the same size and adds their elements together

to form a new list N whose elements are sums of the corresponding elements in L and M. For instance, if L = [3, 1, 4] and M = [1, 5, 9] , then N should equal [4,6,13]. [4, 6, 13].

C:\Solutions to tasks\7.7 Exercises\7.7.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.7.py

```
1  L = eval(input("Enter a first list:   "))
2  M = eval(input("Enter a second list of the same length:  "))
3
4  N = []
5  for i in range(len(M)):
6      N.append(L[i] + M[i])
7  print(N)
```

8. Write a program that asks the user for an integer and creates a list that consists of the factors of that integer.

C:\Solutions to tasks\7.7 Exercises\7.7.8.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.8.py

```
1  num = eval(input("Enter an integer:  "))
2
3  N = []
4  for i in range(1,num + 1):
5      if num % i == 0:
6          N.append(i)
7  print("The factors of the integer ", N)
```

9.When playing games where you have to roll two dice, it is nice to know the odds of each roll. For instance, the odds of rolling a 12 are about 3%, and the odds of rolling a 7 are about 17%. You can compute these mathematically, but if you don't know the math, you can write a program to do it. To do this, your program should simulate rolling two dice about 10,000 times and compute and print out the percentage of rolls that come out to be $2, 3, 4, \ldots, 12$.

C:\Solutions to tasks\7.7 Exercises\7.7.9.py - Sublime Text (UNREGISTERED)

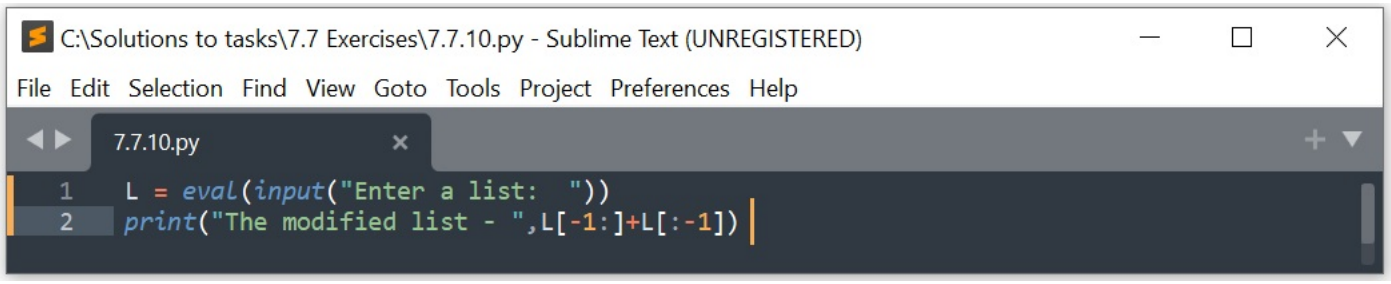
File Edit Selection Find View Goto Tools Project Preferences Help

7.7.9.py

```
1  from random import randint
2  L = []
3  N = [2,3,4,5,6,7,8,9,10,11,12]
4  x = 10000
5  for i in range(x):
6      dice1 = randint(1,6)
7      dice2 = randint(1,6)
8      L.append(dice1 + dice2)
9  for j in N:
10     if j in L:
11         L.count(j)
12         print(j,"-", round(((L.count(j)/x)*100), 3))
```

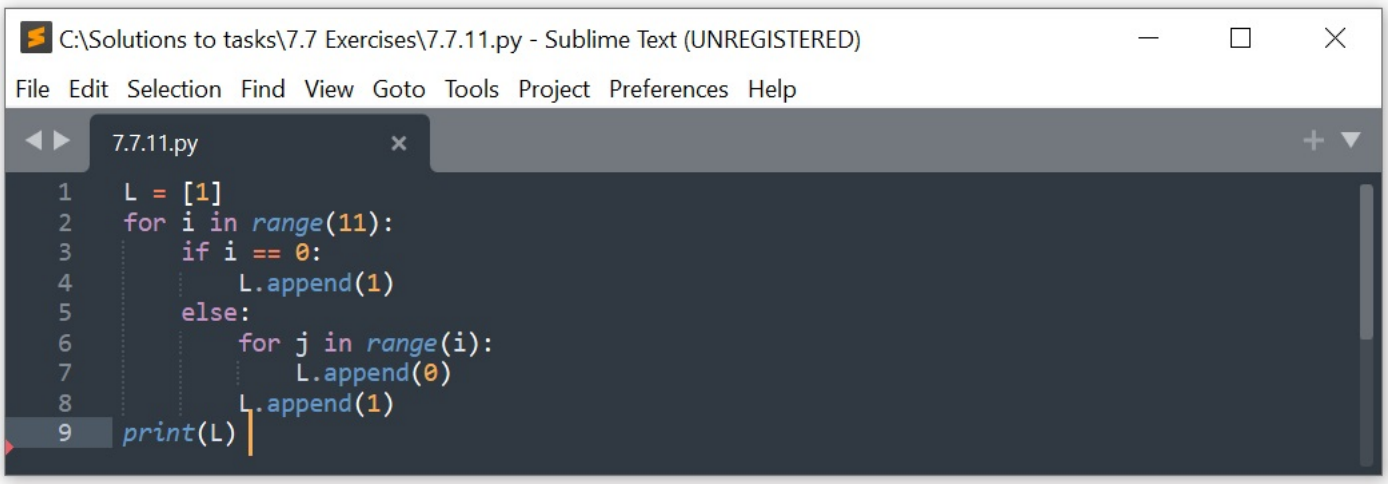
10. Write a program that rotates the elements of a list so that the element at the first index moves to the second index, the element in the second index moves to the third index, etc., and the element in

the last index moves to the first index.

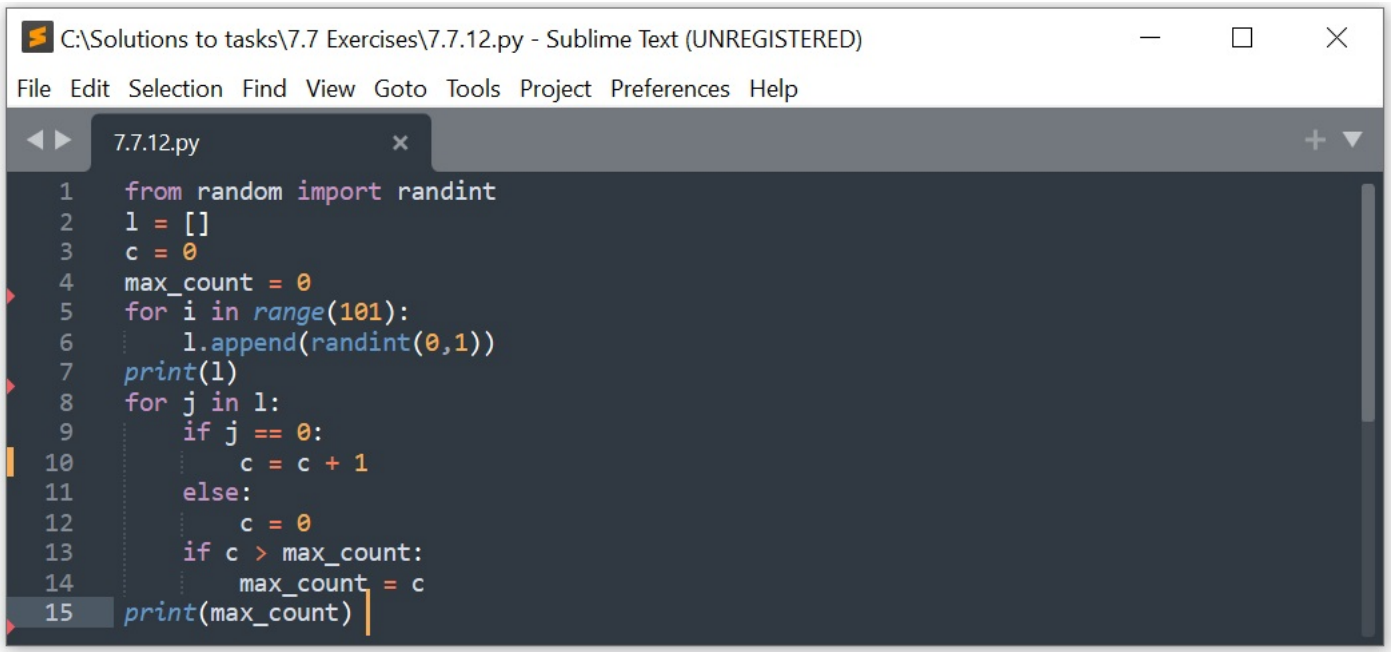


11. Using a for loop, create the list below, which consists of ones separated by increasingly many zeroes. The last two ones in the list should be separated by ten zeroes.

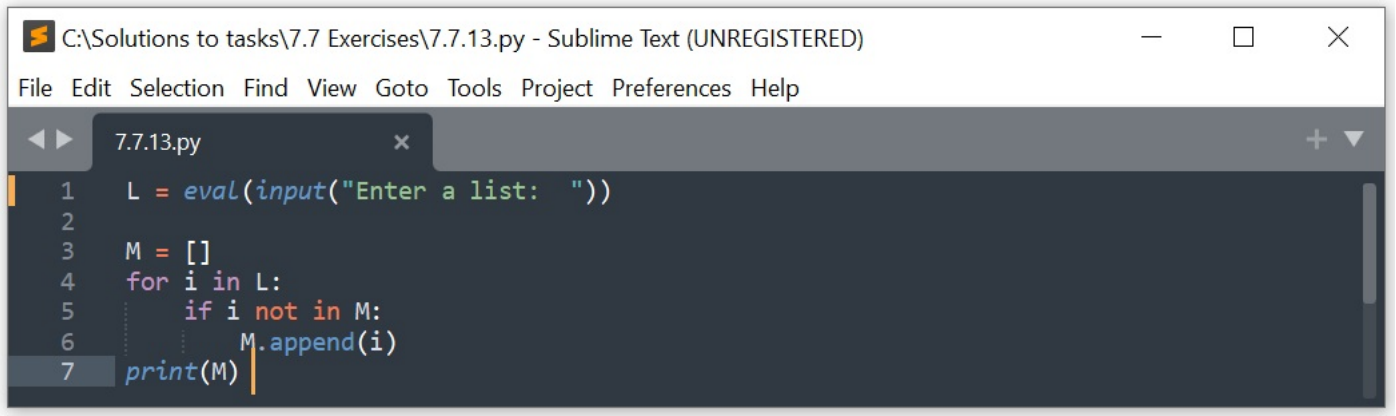
$$[1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, \ldots]$$



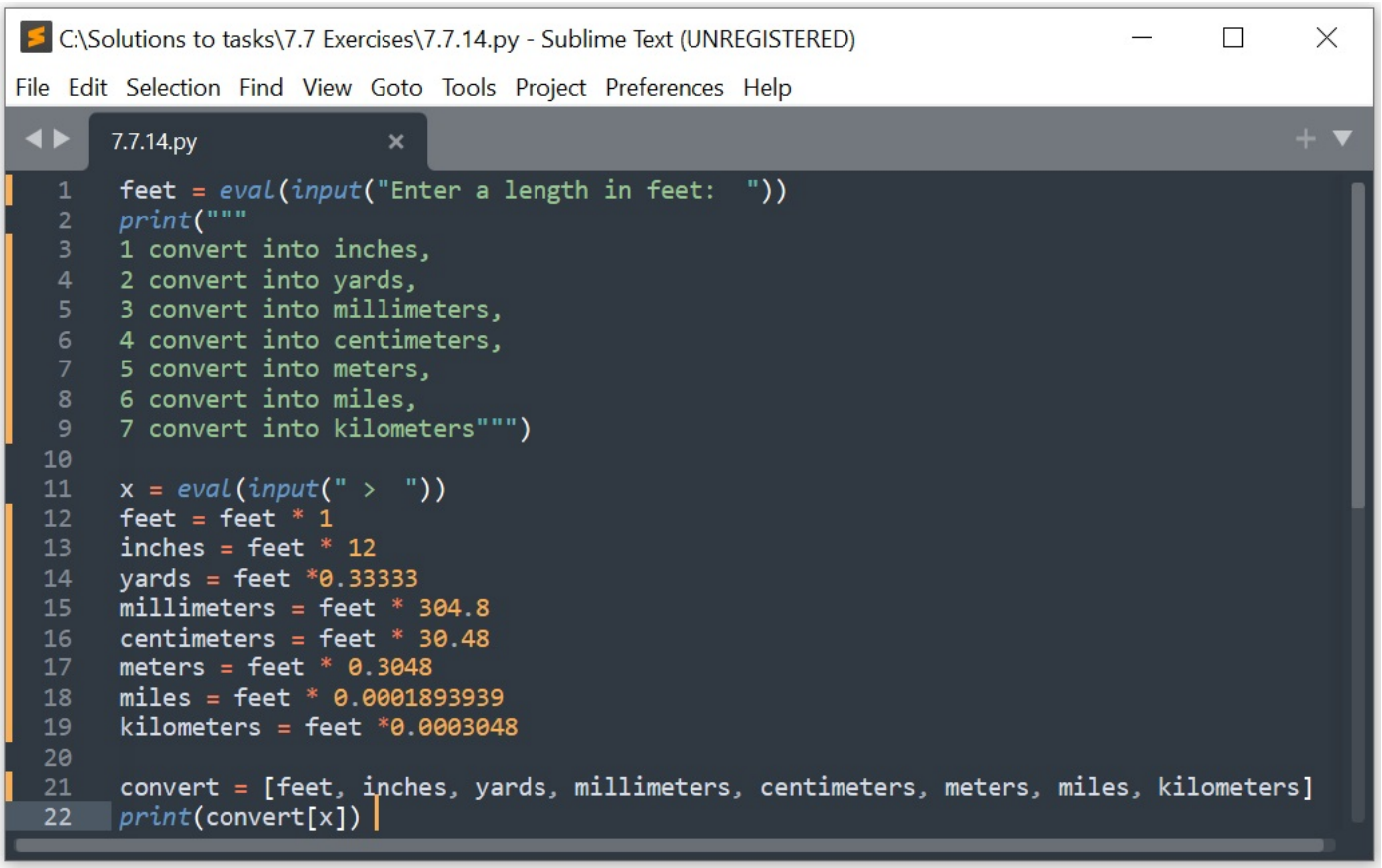
12.. Write a program that generates 100 random integers that are either 0 or 1. Then find the longest *run* of zeros, the largest number of zeros in a row. For instance, the longest run of zeros in $[1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0]$ is 4.



13. Write a program that removes any repeated items from a list so that each item appears at most once. For instance, the list $[1, 1, 2, 3, 4, 3, 0, 0]$ would become $[1, 2, 3, 4, 0]$.



14.Write a program that asks the user to enter a length in feet. The program should then give the user the option to convert from feet into inches, yards, miles, millimeters, centimeters, meters, or kilometers. Say if the user enters a 1, then the program converts to inches, if they enter a 2, then the program converts to yards, etc. While this can be done with if statements, it is much shorter with lists and it is also easier to add new conversions if you use lists.



15. . There is a provably unbreakable cipher called a one-time pad. The way it works is you shift each character of the message by a random amount between 1 and 26 characters, wrapping around the alphabet if necessary. For instance, if the current character is y and the shift is 5, then the new character is d. Each character gets its own shift, so there needs to be as many random shifts as there are characters in the message. As an example, suppose the user enters *secret*. The program should generate a random shift between 1 and 26 for each character. Suppose the randomly generated shifts are $1, 3, 2, 10, 8$ and 2. The encrypted message would be *thebmv*.

(a) Write a program that asks the user for a message and encrypts the message using the one-time pad. First convert the string to lowercase. Any spaces and punctuation in the string should be left unchanged. For example, *Secret*!!! becomes *thebmv*!!! using the shifts above.

C:\Solutions to tasks\7.7 Exercises\7.7.15.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.15.a.py

```
# written by Antonenko Igor(Орск)
# one-time pad (The Vigenère cipher)

from random import randint
message = input("Enter a message: - ")
alphabet = "abcdefghijklmnopqrstuvwxyz"
secret = []
key = []
num = []
for i in range(len(message)):
    char = message[i].lower()
    if char in alphabet:
        shift = randint(0,25)
        num.append(shift)
        key.append(alphabet[shift])
        enc = (alphabet.index(char) + shift)%25
        secret.append(alphabet[enc])
    else:
        secret.append(char)
print("The encrypted message: ", "".join(secret))
print("The key: ","".join(key))
```

(b) Write a program to decrypt a string encrypted as above.

C:\Solutions to tasks\7.7 Exercises\7.7.15.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

7.7.15.b.py

```
# written by Antonenko Igor(Orsk)
secret = input("Enter a encrypted message: - ")
key = input("Enter a key: - ")

alphabet = "abcdefghijklmnopqrstuvwxyz"
decrypted = []
for i in range(len(secret)):
    letter = secret[i]
    if letter in alphabet:
        dec_shift = ((alphabet.index(letter) + 25) - alphabet.index(key[i]))%25
        decrypted.append(alphabet[dec_shift])
    else:
        decrypted.append(letter)
print("The decrypted message: ", "".join(decrypted))
```

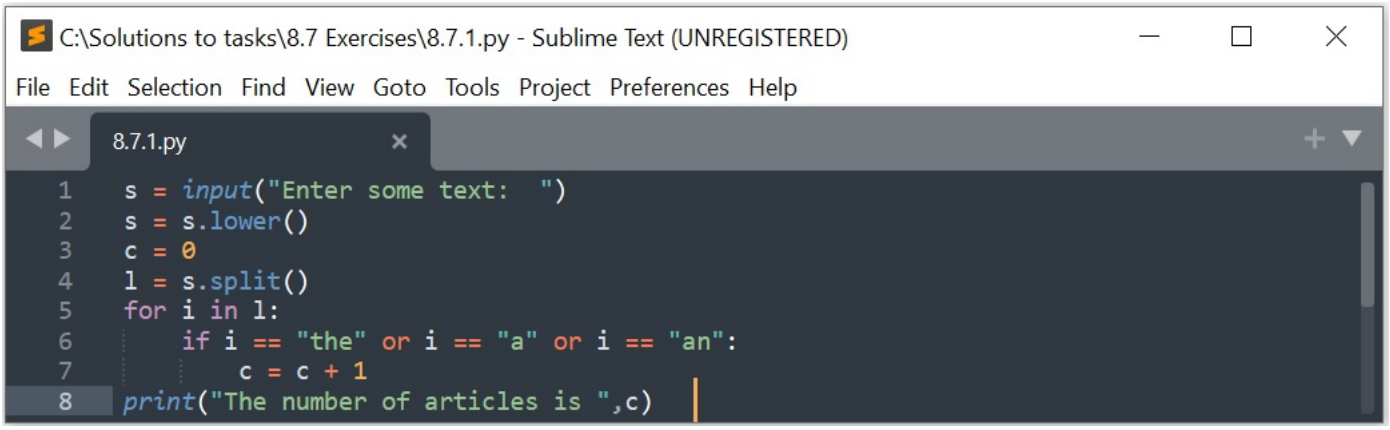
The reason it is called a one-time-pad is that the list of random shifts should only be used once. It becomes easily breakable if the same random shifts are used for more than one message. Moreover, it is only provably unbreakable if the random numbers are truly random, and the numbers generated by *randint* are not truly random. For this problem, just use *randint*, but for cryptographically safe random numbers, see Section 22.8.

Chapter 8

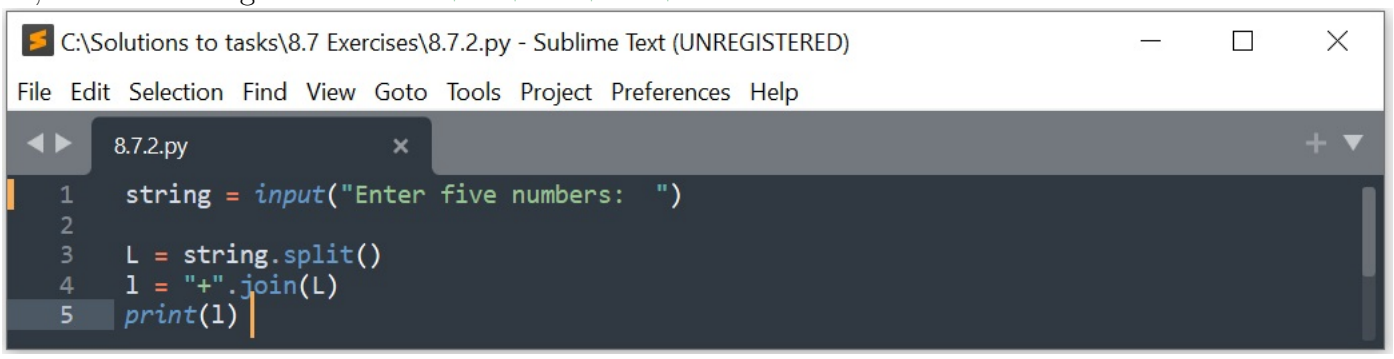
More with Lists

8.1 Exercises 8.7

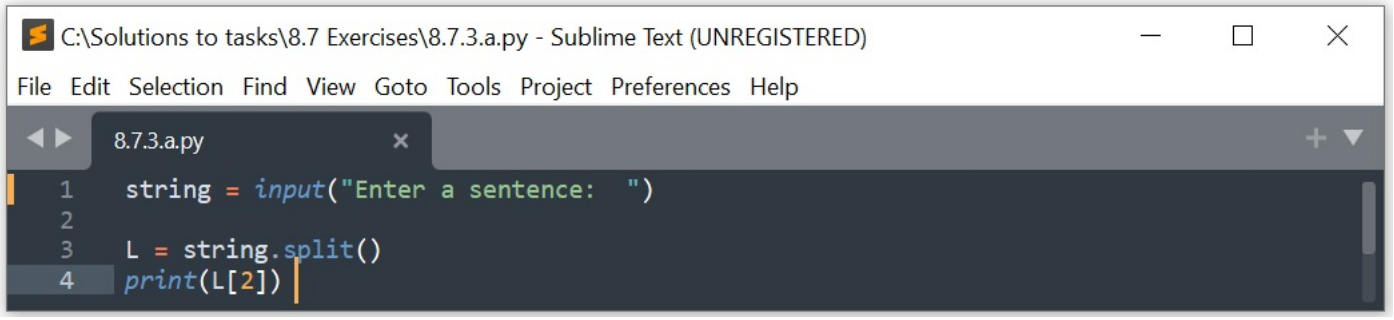
1.Write a program that asks the user to enter some text and then counts how many articles are in the text. Articles are the words 'a', 'an' and 'the'.



2.Write a program that allows the user to enter five numbers (read as strings). Create a string that consists of the user's numbers separated by plus signs. For instance, if the user enters 2, 5, 11, 33, and 55, then the string should be '2 + 5 + 11 + 33 + 55'.



3.(a) Ask the user to enter a sentence and print out the third word of the sentence.



(b) Ask the user to enter a sentence and print out every third word of the sentence.

C:\Solutions to tasks\8.7 Exercises\8.7.3.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.3.b.py

```
1  string = input("Enter a sentence:  ")
2
3  L = string.split()
4  c = 2
5  for i in range(len(L)):
6      if c < len(L):
7          print(L[c])
8          c = c + 3
```

4.(a) Write a program that asks the user to enter a sentence and then randomly rearranges the words of the sentence. Don't worry about getting punctuation or capitalization correct.

C:\Solutions to tasks\8.7 Exercises\8.7.4.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.4.a.py

```
1  from random import shuffle
2
3  string = input("Enter a sentence:  ")
4  L = string.split()
5  shuffle(L)
6  print(" ".join(L))
```

(b) Do the above problem, but now make sure that the sentence starts with a capital, that the original first word is not capitalized if it comes in the middle of the sentence, and that the period is in the right place.

C:\Solutions to tasks\8.7 Exercises\8.7.4.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.4.b.py

```
1  from random import shuffle
2
3  string = input("Enter a sentence:  ")
4  string = string.lower()
5  string = string.replace("."," ")
6  string = string.replace(","," ")
7  L = string.split()
8  shuffle(L)
9  l = " ".join(L)
10 print(l[0].upper() + l[1:] + ".")
```

5. Write a simple quote-of-the-day program. The program should contain a list of quotes, and when the user runs the program, a randomly selected quote should be printed.

C:\Solutions to tasks\8.7 Exercises\8.7.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.5.py

```
1  from random import choice
2
3  L = ["Success comes to those who act.","Don't stop yet\
4  you won't be proud.","Dreams won't work until you do."]
5  l = choice(L)
6  print(l)
```

6.Write a simple lottery drawing program. The lottery drawing should consist of six different numbers between 1 and 48.

C:\Solutions to tasks\8.7 Exercises\8.7.6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.6.py

```
1  from random import randint
2  L = []
3  for i in range(6):
4      L.append(randint(1,48))
5  print(L)
```

7.Write a program that estimates the average number of drawings it takes before the user's numbers are picked in a lottery that consists of correctly picking six different numbers that are between 1 and 10. To do this, run a loop 1000 times that randomly generates a set of user numbers and simulates drawings until the user's numbers are drawn. Find the average number of drawings needed over the 1000 times the loop runs.

C:\Solutions to tasks\8.7 Exercises\8.7.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.7.py

```
1  from random import sample
2
3  user_numbers = sample(range(1,11),6)
4  user_numbers.sort()
5
6  total_count = 0
7  for _ in range(1000):
8      winning_numbers = sample(range(1,11),6)
9      winning_numbers.sort()
10     count = 1
11
12     while user_numbers != winning_numbers:
13         winning_numbers = sample(range(1,11),6)
14         winning_numbers.sort()
15         count += 1
16
17     total_count += count
18     average_count = total_count/1000
19 print("The average number of drawings  - ", round(average_count,2))
```

8.Write a program that simulates drawing names out of a hat. In this drawing, the number of hat entries each person gets may vary. Allow the user to input a list of names and a list of how many entries

each person has in the drawing, and print out who wins the drawing.

C:\Solutions to tasks\8.7 Exercises\8.7.8.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.8.py

```
from random import choice,shuffle

names = input("Enter the names separeted by commas:  ")
entries = eval(input("Enter the entries each person separeted by commas:  "))
L = names.split(",")
X = list(entries)
M = []
for i in range(len(X)):
    for _ in range(X[i]):
        M.append(L[i])
    shuffle(M)
print("The lottery wins ",choice(M))
```

9. Write a simple quiz game that has a list of ten questions and a list of answers to those questions. The game should give the player four randomly selected questions to answer. It should ask the questions one-by-one, and tell the player whether they got the question right or wrong. At the end it should print out how many out of four they got right.

C:\Solutions to tasks\8.7 Exercises\8.7.9.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.9.py

```
from random import sample

Questions = ["The capital of the Russian Federation ",
             "State in the USA has only one neighbor ",
             'The first man in space ',
             'The kind of precious metal is yellow ',
             'The second most important city in Russia ',
             'The river divides Orsk into Europe and Asia ',
             'The capital of the USA ',
             'The cold pole in Russia ',
             'The name of the monetary unit of Russia ',
             'The capital of France ']
Answers = ['Moscow','Maine','Gagarin','Golden','Saint-Petersburg','Ural',\
'Washington','Oymyakon','Rubel','Paris']
c = 0
L = []
s =sample(Questions,4)
for j in range(len(s)):
    x = Questions.index(s[j])
    L.append(Answers[x])
for i in range(len(s)):
    answer = input(s[i])
    if answer == L[i]:
        c = c + 1
        print("Right")
    else:
        print("Wrong")
print("Total correct answers:  ",c)
```

10. Write a censoring program. Allow the user to enter some text and your program should print out

the text with all the curse words starred out. The number of stars should match the length of the curse word. For the purposes of this program, just use the 'curse'words *darn, dang, freakin, heck* and *shoot*. Sample output is below:

> Enter some text: Oh shoot, I thought I had the dang problem figured out. Darn it. Oh well, it was a heck of a freakin try.
>
> Oh ⋆⋆⋆⋆⋆, I thought I had the ⋆⋆⋆⋆ problem figured out. ⋆⋆⋆⋆ it. Oh well, it was a ⋆⋆⋆⋆ of a ⋆⋆⋆⋆⋆ try.

C:\Solutions to tasks\8.7 Exercises\8.7.10.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.10.py

```
"""
Oh shoot, I thought I had the dang problem
figured out. Darn it. Oh well, it was a heck of a freakin try.
"""
s = input("Enter some text: ")
w =["shoot", "darn", "dang","freakin", "heck"]
n = ""
for i in s:
    if i[-1:] == "," or i[-1:] == ".":
        n = n + i[:-1] + " " +i[-1:]
    else:
        n = n  + i
s = n.split()
for i in s:
    if i.lower() in w:
        print("*"*len(i),end = " ")
    else:
        print(i, end = " ")
```

11. Section 8.3 described how to use the *shuffle* method to create a random anagram of a string. Use the *choice* method to create a random anagram of a string.

C:\Solutions to tasks\8.7 Exercises\8.7.11.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.11.py

```
from random import choice

word = input("Enter a word: ")
W = list(word)
A = []
for i in range(len(W)):
    letter = choice(W)
    A.append(letter)
    W.remove(letter)
print("".join(A))
```

12. Write a program that gets a string from the user containing a potential telephone number. The program should print *Valid* if it decides the phone number is a real phone number, and *Invalid* otherwise. A phone number is considered valid as long as it is written in the form *abc-def-hijk* or *1-abc-def-hijk*. The dashes must be included, the phone number should contain only numbers and dashes, and the number of digits in each group must be correct. Test your program with the output shown below.

```
Enter a phone number:  1-301-447-5820
Valid
Enter a phone number:   301-447-5820
Valid
Enter a phone number:   301-4477-5820
Invalid
Enter a phone number:   3X1-447-5820
Invalid
Enter a phone number:   3014475820
Invalid
```

C:\Solutions to tasks\8.7 Exercises\8.7.12.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.12.py

```
# Igor Antonenko
s = input("Enter a phone number.")

if s[:2] =="1-" and len(s) == 14:
    s = s[2:]
else:
    s = s
w = "Wrong!"
if s[3] == "-" and s[7] == "-":
    s = s.split("-")
    if len(s[0]) == 3 and len(s[1]) == 3 and len(s[2]) == 4:
        if "".join(s).isdigit():
            print("Right.")
        else:
            print(w)
    else:
        print(w)
else:
    print(w)
```

13. Let L be a list of strings. Write list comprehensions that create new lists from L for each of the following.

(a). A list that consists of the strings of s with their first characters removed.

C:\Solutions to tasks\8.7 Exercises\8.7.13.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.13.a.py

```
L = ["red","blue","green"]
M = [i[1:] for i in L]
print(M)
```

(b). A list of the lengths of the strings of s.

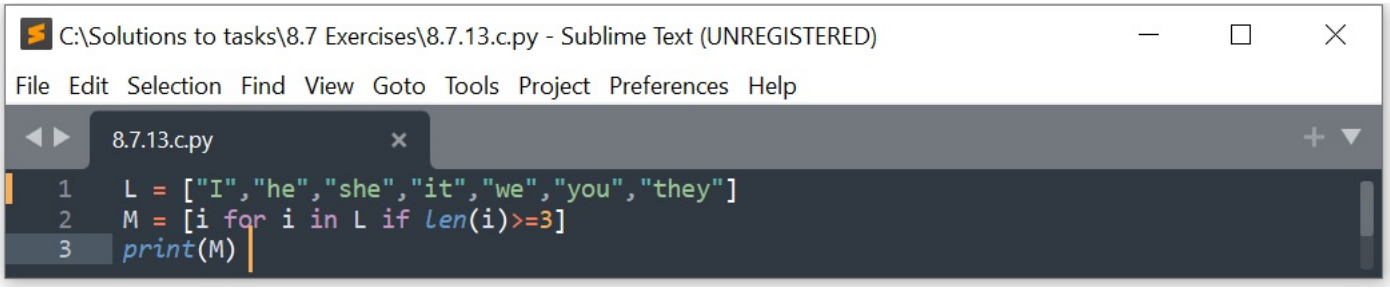
C:\Solutions to tasks\8.7 Exercises\8.7.13.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

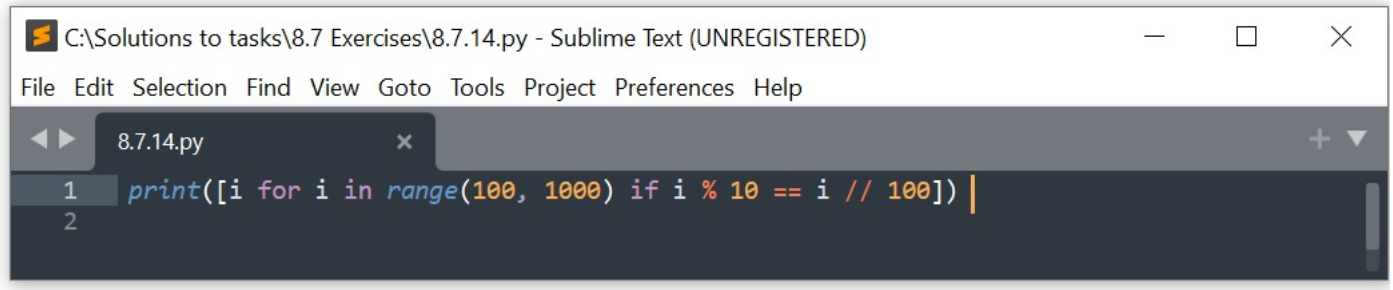
8.7.13.b.py

```
L = ["red","blue","green"]
M = [len(i) for i in L]
print(M)
```

(c). A list that consists of only those strings of s, that are at least three characters long.

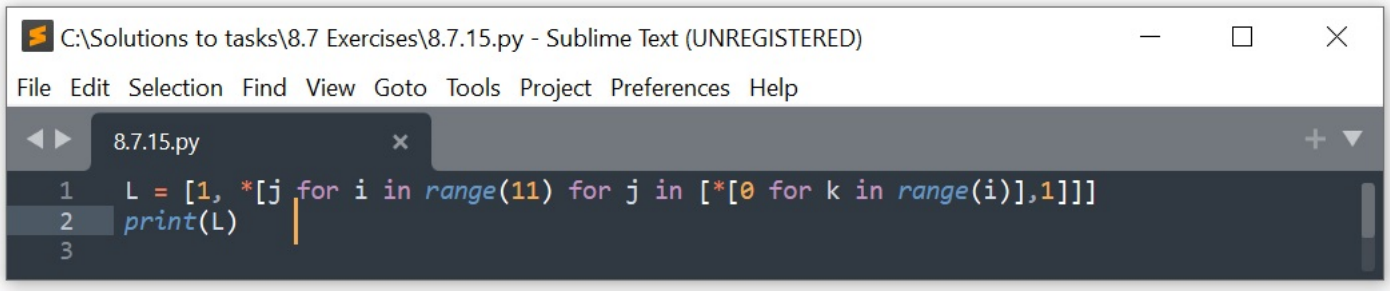


14. Use a list comprehension to produce a list that consists of all palindromic numbers between 100 and 1000.

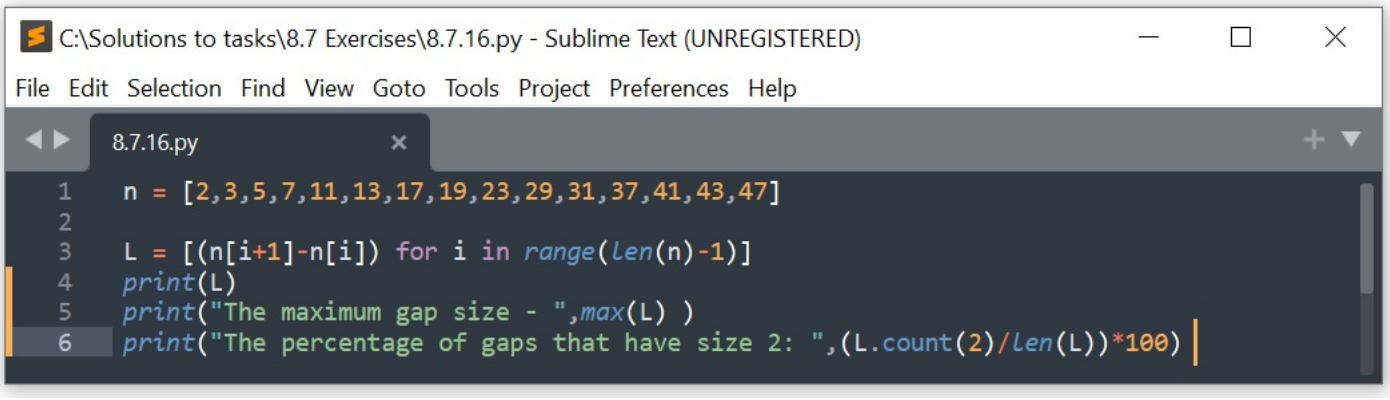


15. to create the list below, which consists of ones separated by increasingly many zeroes. The last two ones in the list should be separated by ten zeroes.

$[\ 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1 \ldots\ ]$



16. Let L = $[\ 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47\ ]$. Use a list comprehension to produce a list of the gaps between consecutive entries in L. Then find the maximum gap size and the percentage of gaps that have size 2



17. Write a program that finds the average of all of the entries in a 4×4 list of integers.

C:\Solutions to tasks\8.7 Exercises\8.7.17.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.17.py

```
1  from random import randint
2  L = []
3  for i in range(4):
4      M =[]
5      for j in range(4):
6          M.append(randint(0,9))
7      L.append(M)
8  print(L)
9  for i in range(len(L)):
10     for j in range(len(L[i])):
11         print(L[i][j], end="  ")
12     print()
13 s = 0
14 for i in range(len(L)):
15     for j in range(len(L[i])):
16         s = s + L[i][j]
17 print("The average of all of the entries ",s/16)
```

18. Write a program that creates a 10×10 list of random integers between 1 and 100. Then do the following:

(a). Print the list.

C:\Solutions to tasks\8.7 Exercises\8.7.18.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.18.a.py

```
1  from random import randint
2
3  L = []
4  for i in range(10):
5      M = []
6      for j in range(10):
7          M.append(randint(1,100))
8      L.append(M)
9  print(L)
10 print()
11
12 for i in range(len(L)):
13     for j in range(len(L[i])):
14         print(L[i][j],end = "  ")
15     print()
```

(b). Find the largest value in the third row.

C:\Solutions to tasks\8.7 Exercises\8.7.18.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.18.b.py

```
from random import randint

L = []
for i in range(10):
    M = []
    for j in range(10):
        M.append(randint(1,100))
    L.append(M)
print(L)
print()
for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j],end = " ")
    print()
print("The largest value in the third row: ", max(L[2]))
```

(c). Find the smallest value in the sixth column

C:\Solutions to tasks\8.7 Exercises\8.7.18.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.18.c.py

```
from random import randint
L = []
for i in range(10):
    M = []
    for j in range(10):
        M.append(randint(1,100))
    L.append(M)
print(L)
print()
for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j],end = " ")
    print()
C = []
for i in range(len(L)):
    C.append(L[i][5])
print(C)
print("The smallest value in the sixth column: ",min(C))
```

19. Write a program that creates and prints an 8×8 list whose entries alternate between 1 and 2 in a checkerboard pattern, starting with 1 in the upper left corner.

C:\Solutions to tasks\8.7 Exercises\8.7.19.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.19.py

```
L = []
c = 0
for i in range(8):
    M = []
    for j in range(8):
        t = c + j
        if t % 2 == 1:
            M.append(2)
        else:
            M.append(1)
    c = c + 1
    L.append(M)

for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j],end = "  ")
    print()
```

20. Write a program that checks to see if a 4×4 list is a magic square. In a magic square, every row, column, and the two diagonals add up to the same value.

C:\Solutions to tasks\8.7 Exercises\8.7.20.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.20.py

```
print("""The example of the magic square
      [[8,11,14,1],
      [13,2,7,12],
      [3,16,9, 6],
      [10,5,4,15]]""")

L = eval(input("Enter a 4x4 list to check "))

D1 = []
for i in range(4):
    D1.append(L[i][i])

D2 = []
for i in range(4):
     D2.append(L[3-i][0+i])

if sum(D1) == sum(D2):
     if sum(L[0]) == sum(L[1]) == sum (L[2]) == sum(L[3]):
          V = []
          for i in range(4):
               v = []
               for j in range(4):
                    v.append(L[j][i])
               V.append(v)
          if sum(V[0]) == sum(V[1]) == sum (V[2]) == sum(V[3]):
               print("The magic square.")
          else:
               print("No the magic square!")
     else:
          print("No the magic square!")
else:
     print("No the magic square!")
```

21. Write a program that asks the user to enter a length. The program should ask them what unit the length is in and what unit they would like to convert it to. The possible units are inches, yards, miles, millimeters, centimeters, meters, and kilometers. While this can be done with 25 if statements, it is shorter and easier to add on to if you use a two-dimensional list of conversions, so please use lists for this problem.

C:\Solutions to tasks\8.7 Exercises\8.7.21.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

8.7.21.py

```
conversion_rates = [
# into inches,yards,miles, millimeters,centimeters,meters,kilometers
[1, 1/36, 1/63360, 25.4, 2.54, 0.0254, 0.0000254],  # from inches
[36, 1, 1/1760, 914.4, 91.44, 0.9144, 0.0009144],   # from yards
[63360, 1760, 1, 1609344, 160934.4, 1609.344, 1.60934],  # from miles
[0.0393701, 0.00109361, 0.000000621371, 1, 0.1, 0.001, 0.000001],#from millimeters
[0.393701, 0.0109361, 0.00000621371, 10, 1, 0.01, 0.00001],  # from centimeters
[39.3701, 1.09361, 0.000621371, 1000, 100, 1, 0.001],  # from meters
[39370.1, 1093.61, 0.621371, 1000000, 100000, 1000, 1]  # from kilometers
]
length = float(input("Enter a length: "))
from_unit = input("What unit the length is (inches,yards,miles,millimeters,\
    centimeters,meters,kilometers): ")
to_unit = input("What unit you would like to convert(inches, yards, miles,\
 millimeters,centimeters, meters, kilometers): ")
units = ["inches","yards","miles","millimeters","centimeters","meters","kilometers"]
from_index = units.index(from_unit)
to_index = units.index(to_unit)
converted_length = length * conversion_rates[from_index][to_index]
print(f"{length}   is {converted_length} ")
```

22. The following is useful as part of a program to play *Battleship*. Suppose you have a 5×5 list that consists of zeroes and ones. Ask the user to enter a row and a column. If the entry in the list at that row and column is a one, the program should print *Hit* and otherwise it should print *Miss*.

C:\Solutions to tasks\8.7 Exercises\8.7.22.py - Sublime Text (UNREGISTERED)

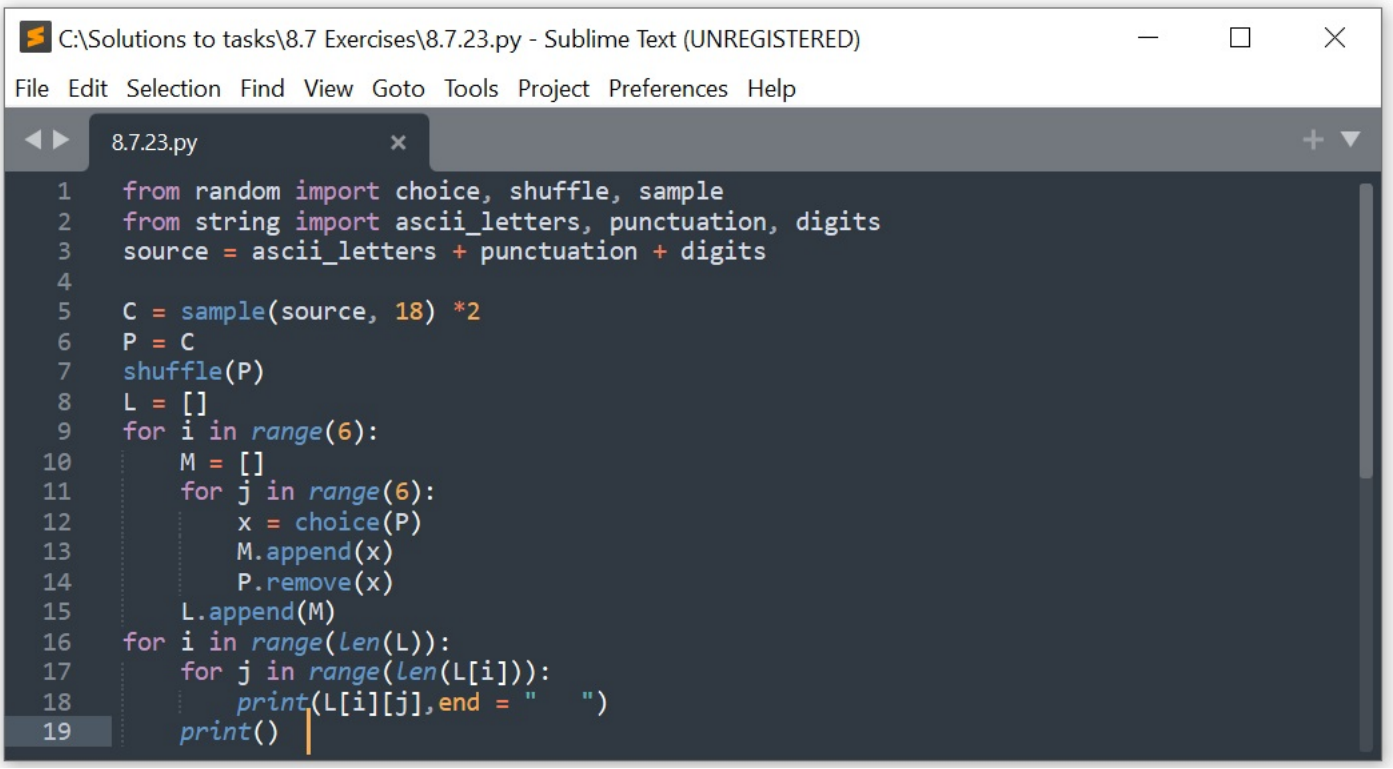
File Edit Selection Find View Goto Tools Project Preferences Help

8.7.22.py

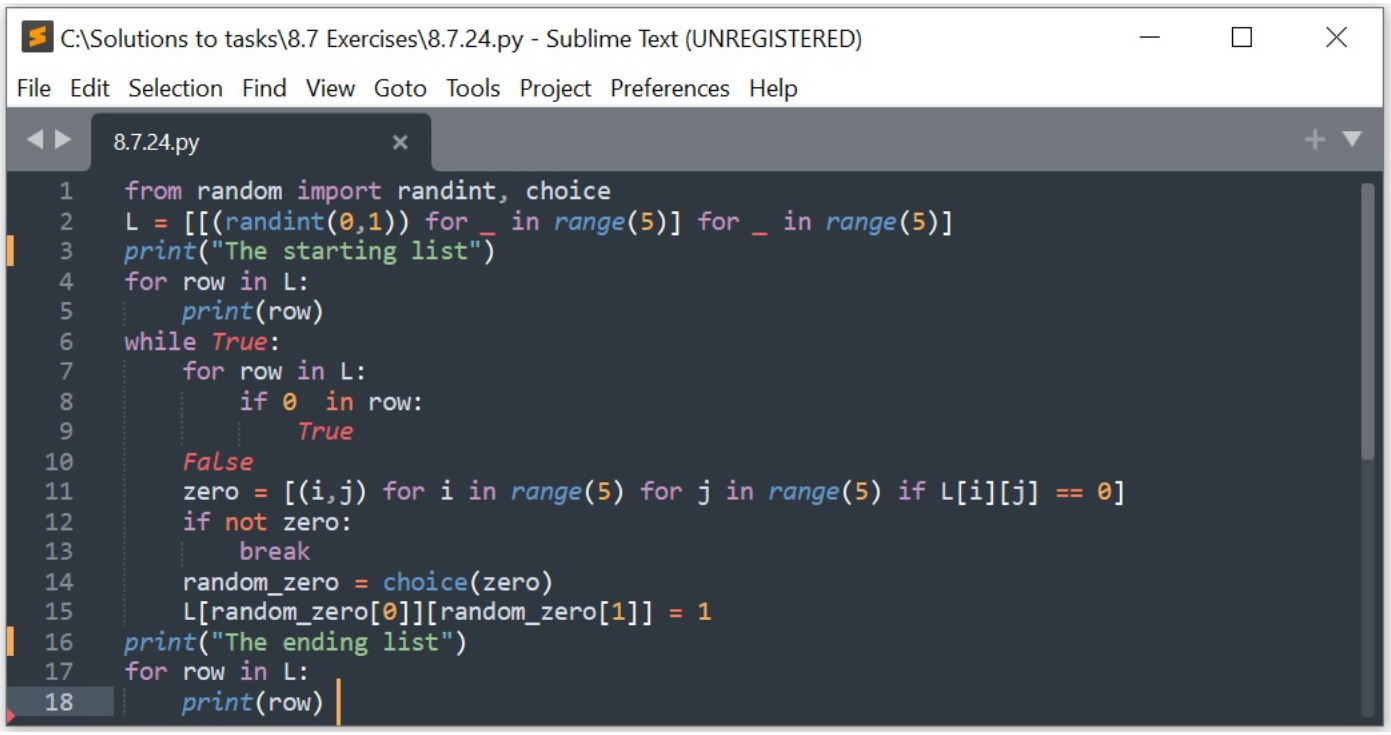
```
from random import randint

L = []
for i in range(5):
    M = []
    for j in range(5):
        M.append(randint(0,1))
    L.append(M)
i = eval(input("Enter a row:  "))
j = eval(input("Enter a column: "))
if L[i-1][j-1] == 1:
    print("Hit")
else:
    print("Miss")
```

23. This exercise is useful in creating a *Memory* game. Randomly generate a 6×6 list of assorted characters such that there are exactly two of each character. An example is shown below.

```
@  5  #  A  A  !
5  0  b  @  $  z
$  N  x  !  N  z
0  -  +  #  b  :
-  :  +  c  c  x
```

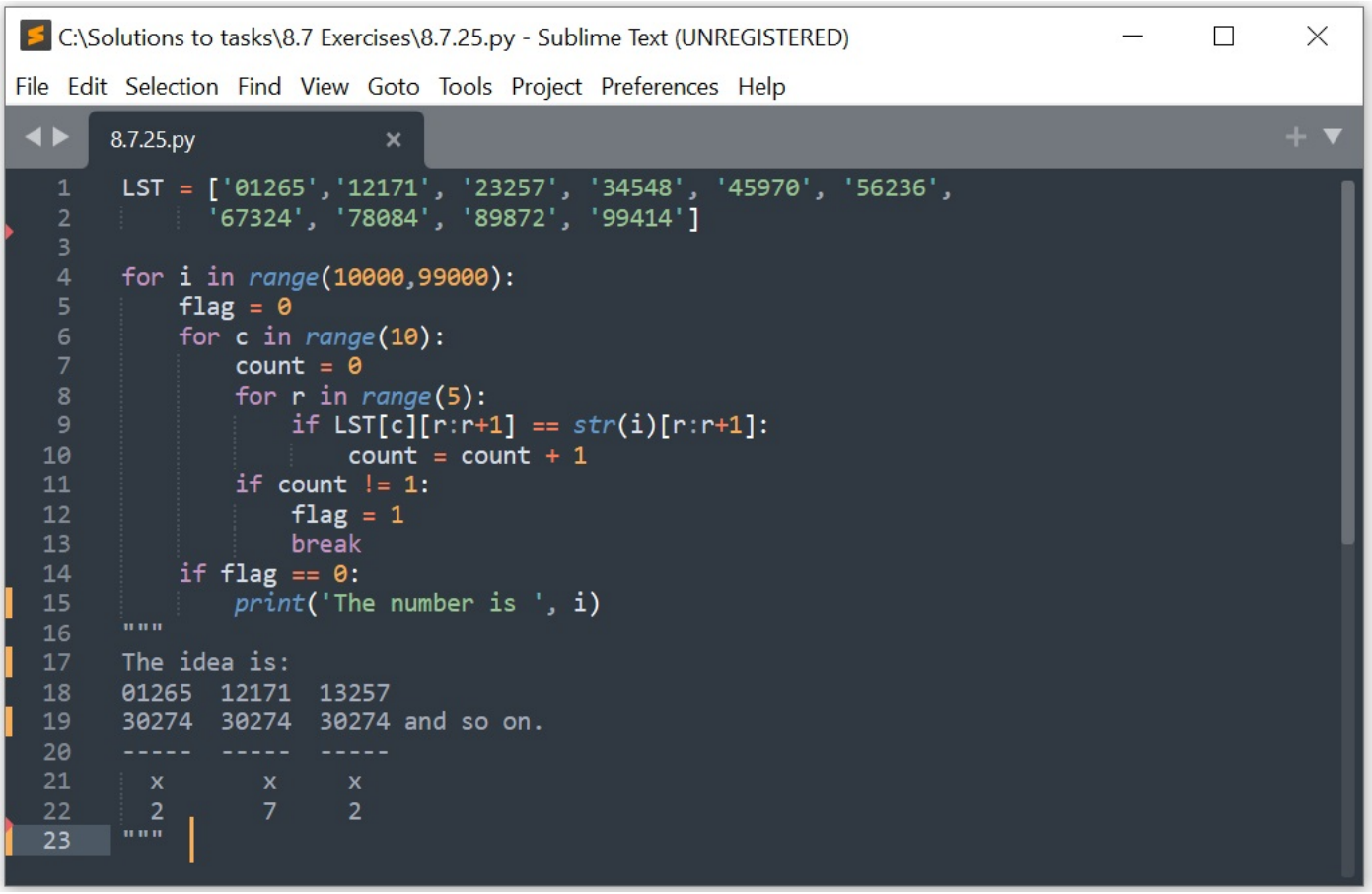


24. The following is useful in implementing computer players in a number of different games. Write a program that creates a 5×5 list consisting of zeroes and ones. Your program should then pick a random location in the list that contains a zero and change it to a one. If all the entries are one, the program should say so. [Hint: one way to do this is to create a new list whose items are the coordinates of all the ones in the list and use the *choice* method to randomly select one. Use a two-element list to represent a set of coordinates.]



25. Here is an old puzzle question you can solve with a computer program. There is only one five-digit number n that is such that every one of the following ten numbers shares exactly one digit in common in the same position as n. Find n.

01265, 12171, 23257, 34548, 45970, 56236, 67324, 78084, 89872, 99414

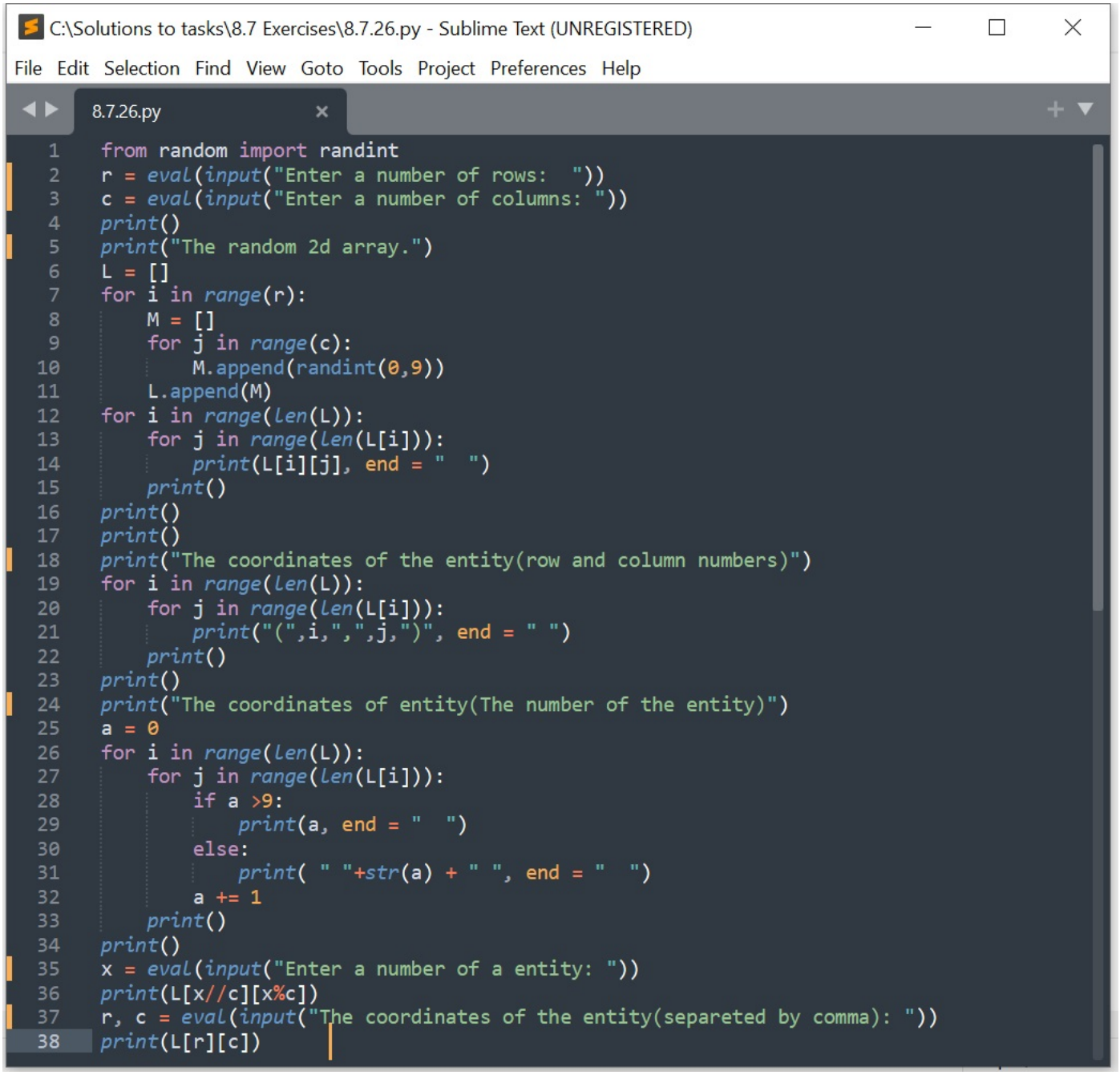


26. We usually refer to the entries of a two-dimensional list by their row and column, like below on the left. Another way is shown below on the right.

(0, 0)	(0, 1)	(0, 2)	0	1	2
(1, 0)	(1, 1)	(1, 2)	3	4	5
(2, 0)	(2, 1)	(2, 2)	6	7	8

(a). Write some code that translates from the left representation to the right one. The // and % operators will be useful. Be sure your code works for arrays of any size.

(b). Write some code that translates from the right representation to the left one.


C:\Solutions to tasks\8.7 Exercises\8.7.26.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
8.7.26.py
from random import randint
r = eval(input("Enter a number of rows: "))
c = eval(input("Enter a number of columns: "))
print()
print("The random 2d array.")
L = []
for i in range(r):
M = []
for j in range(c):
M.append(randint(0,9))
L.append(M)
for i in range(len(L)):
for j in range(len(L[i])):
print(L[i][j], end = " ")
print()
print()
print()
print("The coordinates of the entity(row and column numbers)")
for i in range(len(L)):
for j in range(len(L[i])):
print("(",i,",",j,")", end = " ")
print()
print()
print("The coordinates of entity(The number of the entity)")
a = 0
for i in range(len(L)):
for j in range(len(L[i])):
if a >9:
print(a, end = " ")
else:
print(" "+str(a) + " ", end = " ")
a += 1
print()
print()
x = eval(input("Enter a number of a entity: "))
print(L[x//c][x%c])
r, c = eval(input("The coordinates of the entity(separeted by comma): "))
print(L[r][c])

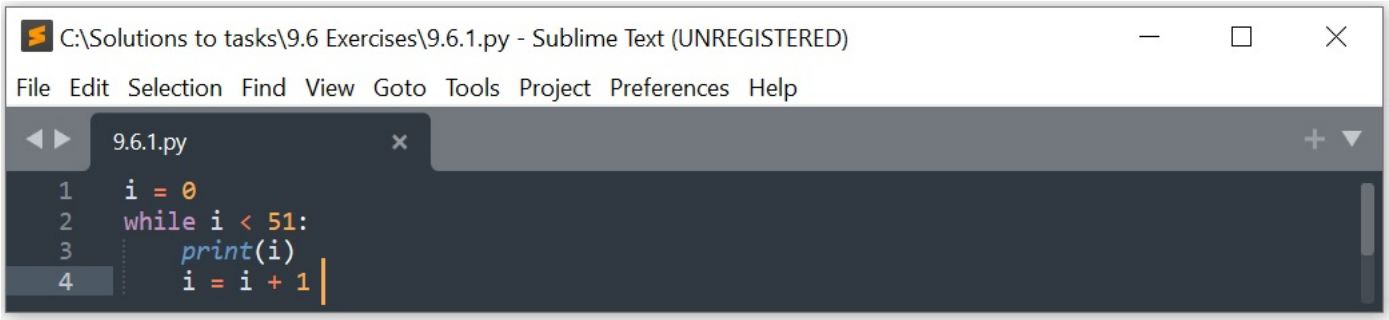
Chapter 9

While loops

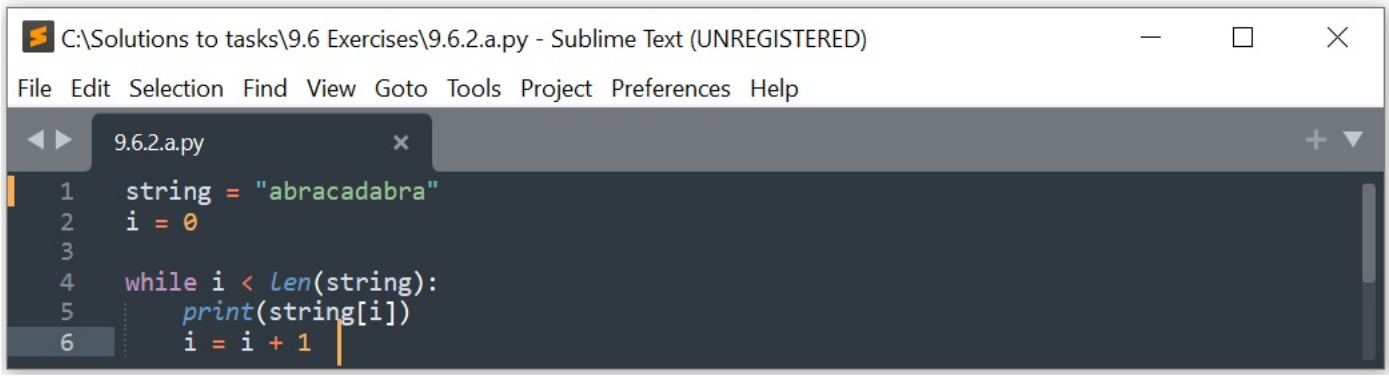
9.1 Exercise 9.6

1. The code below prints the numbers from 1 to 50. Rewrite the code using a while loop to accomplish the same thing.

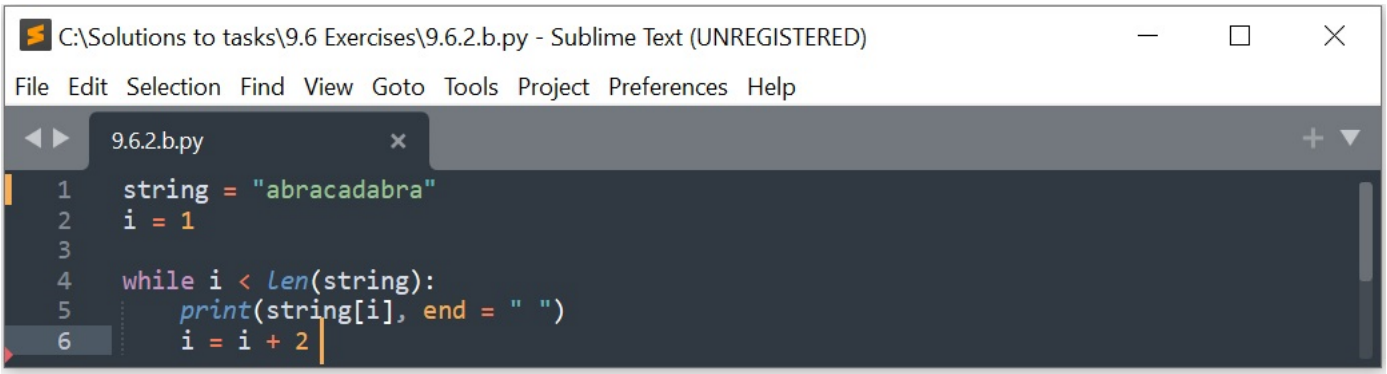
```
for i in range(1, 51):
    print( i )
```



2.(a). Write a program that uses a while (not a for loop) to read through a string and print the characters of the string one-by-one on separate lines.

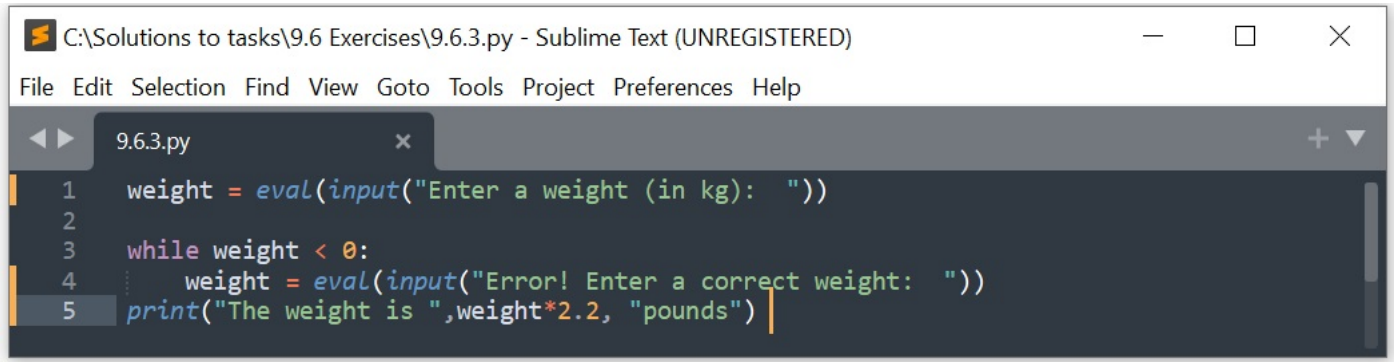


(b). Modify the program above to print out every second character of the string.

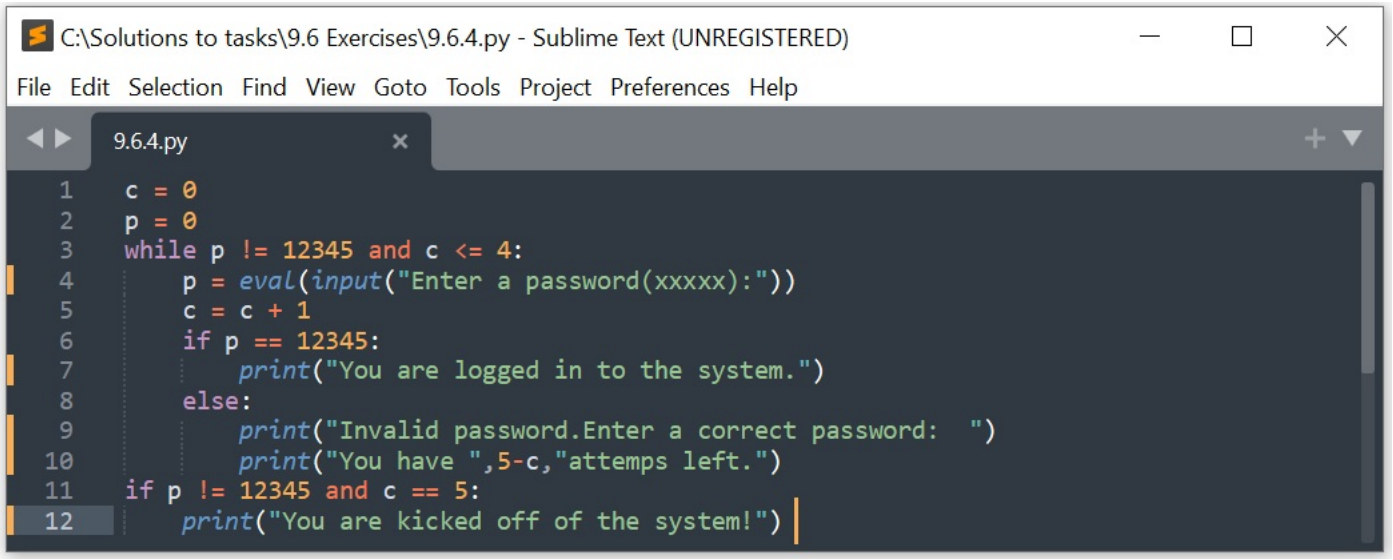


3. A good program will make sure that the data its users enter is valid. Write a program that asks the

user for a weight and converts it from kilograms to pounds. Whenever the user enters a weight below 0, the program should tell them that their entry is invalid and then ask them again to enter a weight. [Hint: Use a while loop, not an if statement].



4. Write a program that asks the user to enter a password. If the user enters the right password, the program should tell them they are logged in to the system. Otherwise, the program should ask them to reenter the password. The user should only get five tries to enter the password, after which point the program should tell them that they are kicked off of the system.



5. Write a program that allows the user to enter any number of test scores. The user indicates they are done by entering in a negative number. Print how many of the scores are A's (90 or above). Also print out the average.

C:\Solutions to tasks\9.6 Exercises\9.6.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.5.py

```
print("To complete, enter a negative number.")
score = 0
scores = []
grades = []

while score >= 0:
    score = eval(input("Enter a number of test scores:  "))
    scores.append(score)
for score in scores:
    if score > 90:
        grades.append("A")
    elif score > 60:
        grades.append("B")
    elif score > 60:
        grades.append("C")
    elif score > 50:
        grades.append("D")
    elif score > 40:
        grades.append("E")
    elif score > 0:
        grades.append("F")
    elif score < 0:
        scores.pop()
print(scores)
print(grades)
print("The number of A's is ",grades.count("A"))
print("The average of test scores -",sum(scores)/len(scores))
```

6. Modify the higher \lower program so that when there is only one guess left, it says 1 *guess*, not 1 *guesses*.

C:\Solutions to tasks\9.6 Exercises\9.6.6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.6.py

```
from random import randint

secret_num = randint(1,100)
num_guesses = 0
guess = 0
while guess != secret_num and num_guesses <= 4:
    guess = eval(input('Enter your guess (1-100): '))
    num_guesses = num_guesses + 1
    if guess < secret_num:
        if num_guesses != 4:
            print('HIGHER.', 5-num_guesses, 'guesses left.\n')
        else:
            print("HIGHER. 1 guess left.\n")
    elif guess > secret_num:
        if num_guesses != 4:
            print('LOWER.', 5-num_guesses, 'guesses left.\n')
        else:
            print("LOWER. 1 guess left.\n")
    else:
        print('You guessed!')
if num_guesses==5 and guess != secret_num:
    print('Wrong!. The correct answer - ', secret_num)
```

7. Recall that, given a string s, $s.index('x')$ returns the index of the first x in s and an error if there is no x.

(a). Write a program that asks the user for a string and a letter. Using a while loop, the program should print the index of the first occurrence of that letter and a message if the string does not contain the letter.

C:\Solutions to tasks\9.6 Exercises\9.6.7.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.7.a.py

```
1   string = input("Enter a string:  ")
2   letter = input("Enter a letter:  ")
3   i = 2
4   string = string.lower()
5   letter = letter.lower()
6
7   while i!= 0:
8       if letter in string:
9           print("The index is ",string.index(letter))
10          i = 1
11          break
12      i = 0
13  if i == 0:
14      print("The string does not contain the letter",letter)
```

(b). Write the above program using a for\break loop instead of a while loop.

C:\Solutions to tasks\9.6 Exercises\9.6.7.b.py - Sublime Text (UNREGISTERED)

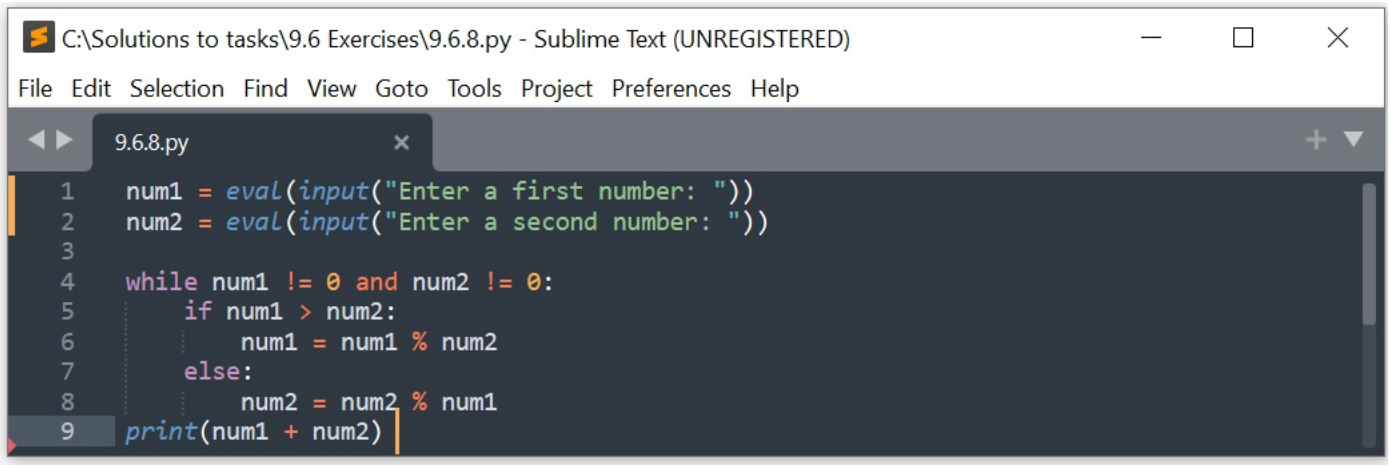
File Edit Selection Find View Goto Tools Project Preferences Help

9.6.7.b.py

```
1   string = input("Enter a string:  ")
2   letter = input("Enter a letter:  ")
3   string = string.lower()
4   letter = letter.lower()
5   for i in string:
6       if i == letter:
7           print("The index is",string.index(letter))
8           break
9   else:
10      print("The string does not contain the letter",letter)
```

8. The GCD (greatest common divisor) of two numbers is the largest number that both are divisible by. For instance, gcd(18, 42) is 6 because the largest number that both 18 and 42 are divisible by is 6. Write a program that asks the user for two numbers and computes their gcd. Shown below is a way to compute the GCD, called Euclid's Algorithm.

- First compute the remainder of dividing the larger number by the smaller number
- Next, replace the larger number with the smaller number and the smaller number with the remainder.
- Repeat this process until the smaller number is 0. The GCD is the last value of the larger number.



9. A 4000-year old method to compute the square root of 5 is as follows: Start with an initial guess, say 1. Then compute

$$\frac{1+\frac{5}{1}}{2} = 3$$

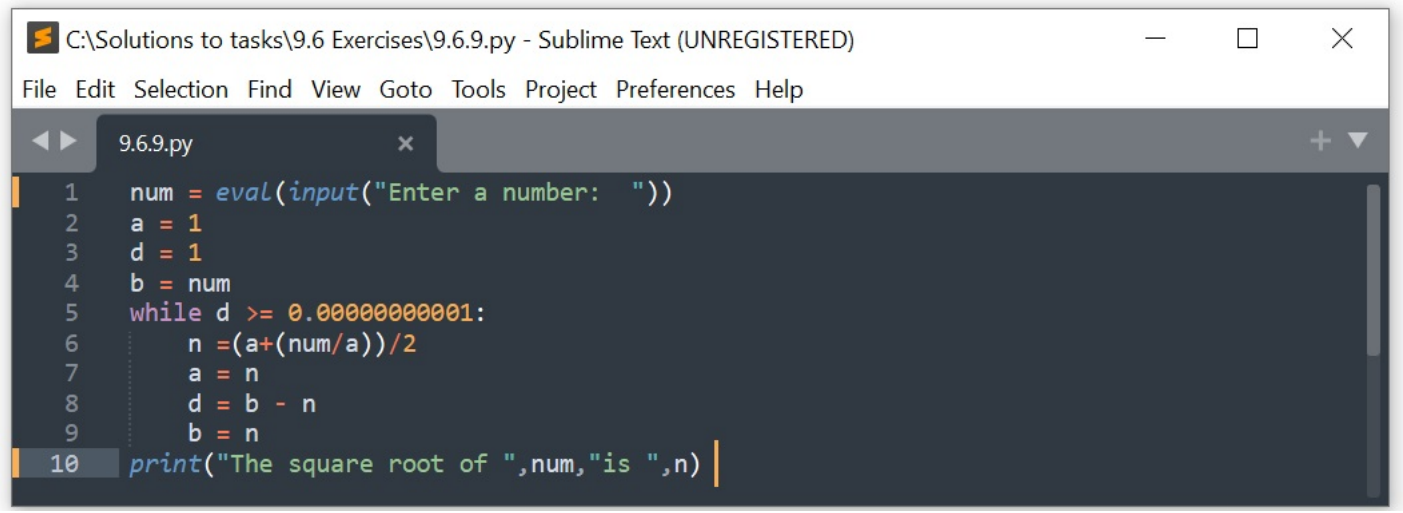
Next, take that 3 and replace the 1's in the previous formula with 3's . This gives

$$\frac{3+\frac{5}{3}}{2} \approx 2.33.$$

Next replace the 3 in the previous formula with 7\3. This gives

$$\frac{\frac{7}{3}+\frac{5}{\frac{7}{3}}}{2} = \frac{47}{21} \approx 2.24$$

If you keep doing this process of computing the formula, getting a result, and plugging it back in, the values will eventually get closer and closer to $\sqrt{5}$. This method works for numbers other than 5. Write a program that asks the user for a number and uses this method to estimate the square root of the number correct to within 10^{-10}. The estimate will be correct to within 10^{-10} when the absolute value of the difference between consecutive values is less than 10^{-10}.



10. Write a program that has a list of ten words, some of which have repeated letters and some which don't. Write a program that picks a random word from the list that does not have any repeated letters.

C:\Solutions to tasks\9.6 Exercises\9.6.10.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.10.py

```
1  from random import choice
2  L = ["word","letter","program","different","seven","value",
3      "right","root","list","guess"]
4  c = 0
5
6  while c != 1:
7      word = choice(L)
8      for i in range(len(word)):
9          c = word.count(word[i])
10         if c == 2:
11             break
12 print(word)
```

11. Write a program that starts with an 5×5 list of zeroes and randomly changes exactly ten of those zeroes to ones.

C:\Solutions to tasks\9.6 Exercises\9.6.11.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.11.py

```
1  from random import randint
2  L = [[0 for i in range(5)] for i in range(5)]
3  for i in range(len(L)):
4      for j in range(len(L[i])):
5          print(L[i][j],end="  ")
6      print()
7  print()
8
9  n = 0
10 while n < 10:
11     i = randint(0,4)
12     j = randint(0,4)
13     if L[i][j] == 0:
14         L[i][j] = 1
15         n = n + 1
16 for i in range(len(L)):
17     for j in range(len(L[i])):
18         print(L[i][j],end="  ")
19     print()
```

12. Write a program in which you have a list that contains seven integers that can be 0 or 1. Find the first nonzero entry in the list and change it to a 1. If there are no nonzero entries, print a message saying so.

C:\Solutions to tasks\9.6 Exercises\9.6.12.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.12.py

```
from random import randint

L = [randint(0,10) for _ in range(7)]
print(L)

n = 0
while True:
    for i in range(len(L)):
        if L[i] > 0:
            L[i] = 1
            n = 1
            print(L)
            break
    break
if n == 0:
    print("There are not nonzero entries.")
```

13. In Chapter 4 there was a problem that asked you to write a program that lets the user play Rock-Paper-Scissors against the computer. In that program there were exactly five rounds. Rewrite the program so that it is a best 3 out of 5. That is, the first player to win three times is the winner.

C:\Solutions to tasks\9.6 Exercises\9.6.13.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.13.py

```
from random import randint
win = 0
lost = 0
tie = 0
i = 1
print("Paper - 1,Rock - 2,Scissors - 3")
while True:
    r = randint(1,3)
    print("Round -",i)
    g = eval(input("Your turn- "))
    if r == 1 and g == 1 or r == 2 and g == 2 or r ==3 and g == 3:
        tie = tie + 1
    if r == 2 and g == 3 or r == 3 and g == 1 or r == 3 and g == 1 \
       or r == 1 and g == 2:
        lost = lost + 1
    if r == 3 and g == 2 or r == 1 and g == 3 or r == 1 and g == 3 \
       or r == 2 and g == 1:
        win = win + 1
    if lost == 3:
        print("The computer wins.")
        break
    if win == 3:
        print("The player wins.")
        break
    i = i + 1
    if i == 6:
        break
print("Game over.")
print("Ties - ",tie)
print("Loss - ",lost)
print("Wins - ",win)
```

14. Write a program to play the following simple game. The player starts with $100. On each turn a

coin is flipped and the player has to guess heads or tails. The player wins $9 for each correct guess and loses $10 for each incorrect guess. The game ends either when the player runs out of money or gets to $200.

C:\Solutions to tasks\9.6 Exercises\9.6.14.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.14.py

```
from random import randint
money = 100
while True:
    p = randint(0,1)
    guess = eval(input("Heads(1) or Tails(0) - "))
    if p == guess:
        money = money + 9
        print("Right!")
    else:
        money = money - 10
        print("Wrong!")
    print("The amount of money",money)
    if money >= 200:
        print("You have won! Your money - ",money)
        break
    if money <= 0:
        print("You have lost.Your money - ",money)
        break
```

15. Write a program to play the following game. There is a list of several country names and the program randomly picks one. The player then has to guess letters in the word one at a time. Before each guess the country name is displayed with correctly guessed letters filled in and the rest of the letters represented with dashes. For instance, if the country is *Canada* and the player has correctly guessed it a, *d*, and *n*, the program would display *-ana-da*. The program should continue until the player either guesses all of the letters of the word or gets five letters wrong.

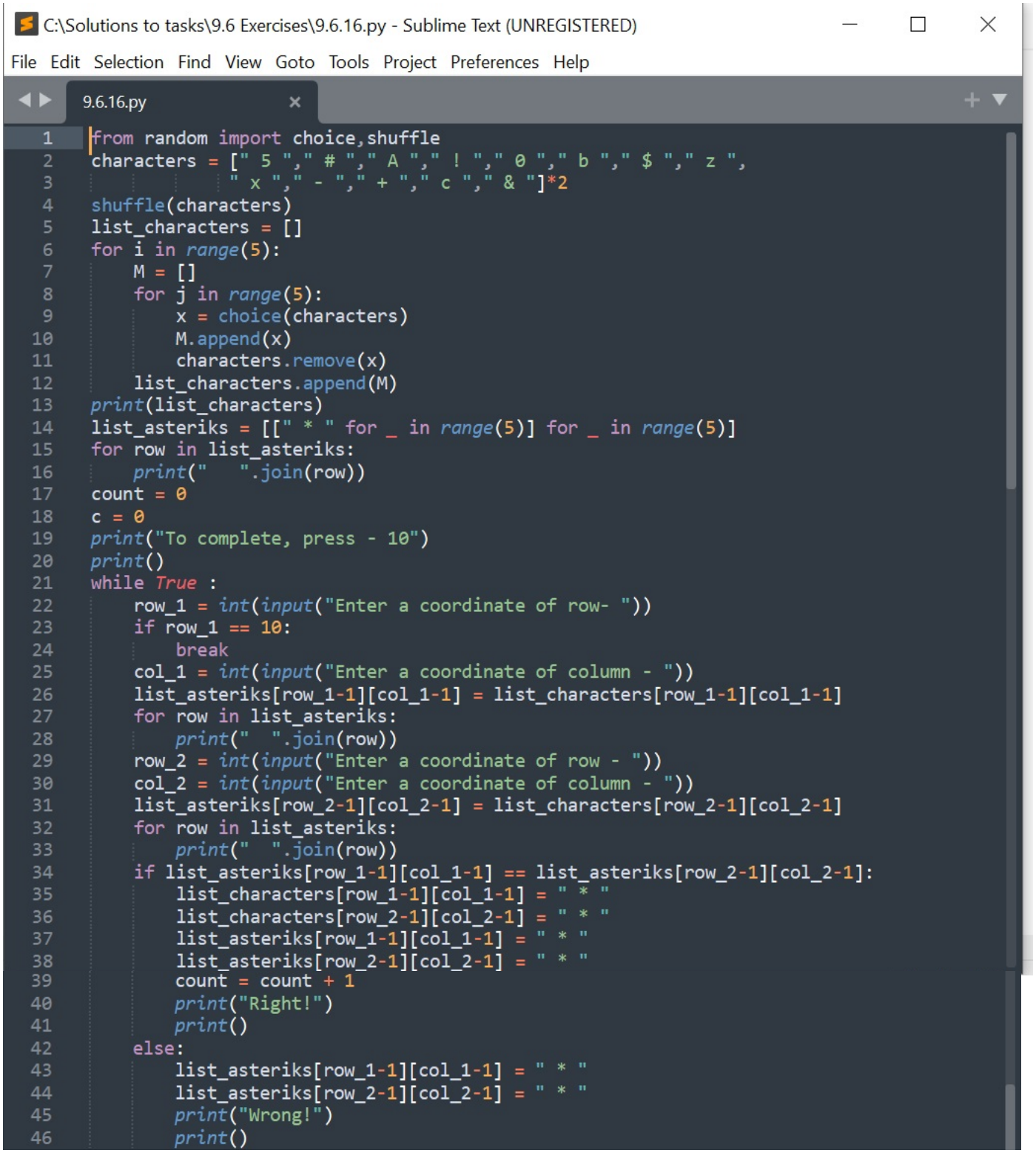
C:\Solutions to tasks\9.6 Exercises\9.6.15.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.15.py

```
from random import choice
C = ["Canada","Russian","China","Germany","Spain","Japan",
     "Indonesia","Polen","Chile"]
c = choice(C)
k = list(c)
A = []
for j in range(len(c)):
    A.append("-")
print(A)
count = 0
print("The first letter is uppercase.")
while "-" in A and count < 5:
    letter = input("Enter a letter - ")
    if letter in k:
        for i in range(len(k)):
            if k[i]==letter:
                A[i]=letter
    else:
        count = count + 1
        print("The letter was not found.")
        print("Guesses left - ",5 - count)
    print(" ".join(A))
if count >= 5:
    print("You have lost!")
    print("The word is - ",c)
```

16. Write a text-based version of the game *Memory*. The game should generate a 5×5 board (see the exercise from Chapter 8). Initially the program should display the board as a 5×5 grid of asterisks. The user then enters the coordinates of a cell. The program should display the grid with the character at those coordinates now displayed. The user then enters coordinates of another cell. The program should now display the grid with the previous character and the new character displayed. If the two characters match, then they should permanently replace the asterisks in those locations. Otherwise, when the user enters the next set of coordinates, those characters should be replaced by asterisks. The game continues this way until the player matches everything or runs out of turns. You can decide how many turns they player gets.



```
47      for row in list_characters:
48          c = c + row.count(" * ")
49      if c == 24:
50          break
51  print("Game over.")
52  print("The number of correct guesses- ", count)
```

17. Ask the user to enter the numerator and denominator of a fraction, and the digit they want to know. For instance, if the user enters a numerator of 1 and a denominator of 7 and wants to know the 4th digit, your program should print out 8, because $\frac{1}{7} = 0.142856\ldots$ and 8 is the 4th digit. One way to do this is to mimic the long division process you may have learned in grade school. It can be done in about five lines using the // operator at one point in the program.

C:\Solutions to tasks\9.6 Exercises\9.6.17.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.17.py

```
1  numerator = int(input("Enter the numerator of a fraction: "))
2  denominator = int(input("Enter the denominator of a fraction: "))
3  digit_position = int(input("Enter the position of the desired digit: "))
4
5  decimal = numerator / denominator
6
7  digit = int(decimal * 10**digit_position) % 10
8  print("The digit with a position ", digit_position, "is:", digit)
```

18. Randomly generate a 6×6 list that has exactly 12 ones placed in random locations in the list. The rest of the entries should be zeroes.

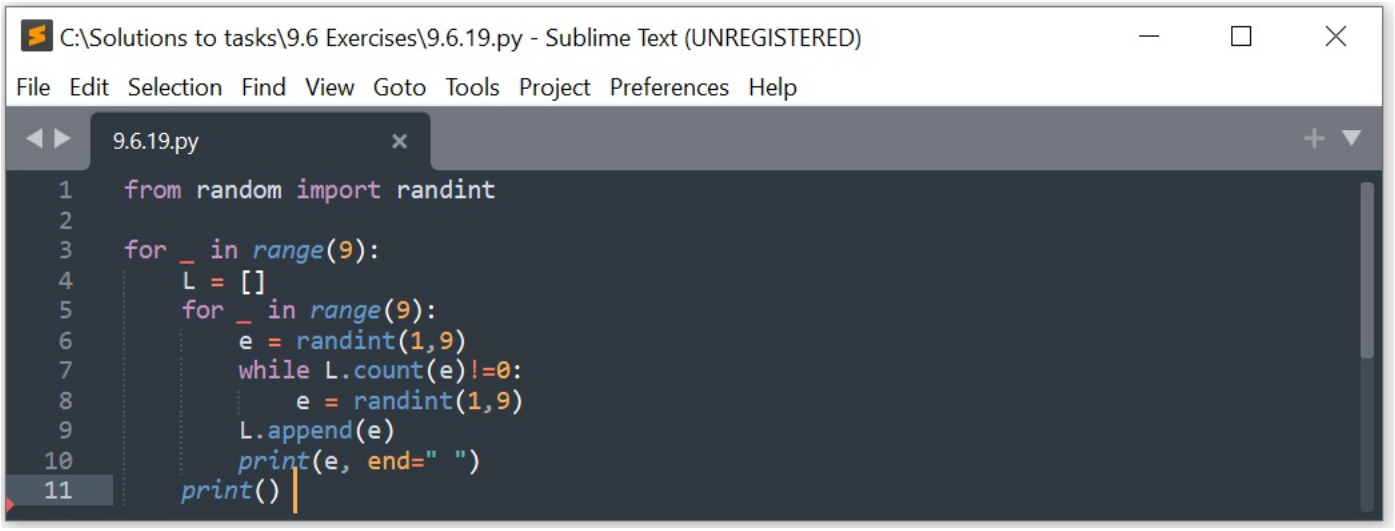
C:\Solutions to tasks\9.6 Exercises\9.6.18.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

9.6.18.py

```
1  from random import randint
2  L = [[0 for i in range(6)] for i in range(6)]
3  count = 0
4  while count < 12:
5      x = randint(0,5)
6      y = randint(0,5)
7      if L[x][y] != 1:
8          L[x][y] = 1
9          count = count + 1
10 for i in range(len(L)):
11     for j in range(len(L[i])):
12         print(L[i][j],end="  ")
13     print()
```

19. . Randomly generate a 9×9 list where the entries are integers between 1 and 9 with no repeat entries in any row or in any column.

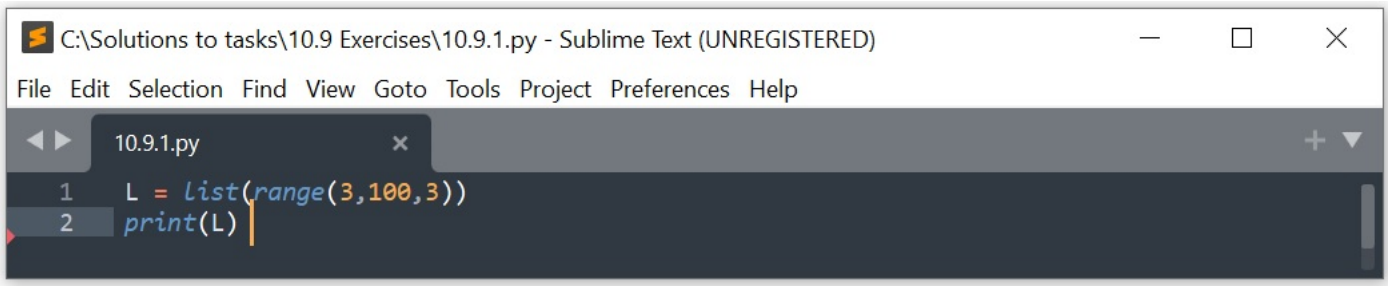

C:\Solutions to tasks\9.6 Exercises\9.6.19.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
9.6.19.py
from random import randint
for _ in range(9):
L = []
for _ in range(9):
e = randint(1,9)
while L.count(e)!=0:
e = randint(1,9)
L.append(e)
print(e, end=" ")
print()

Chapter 10

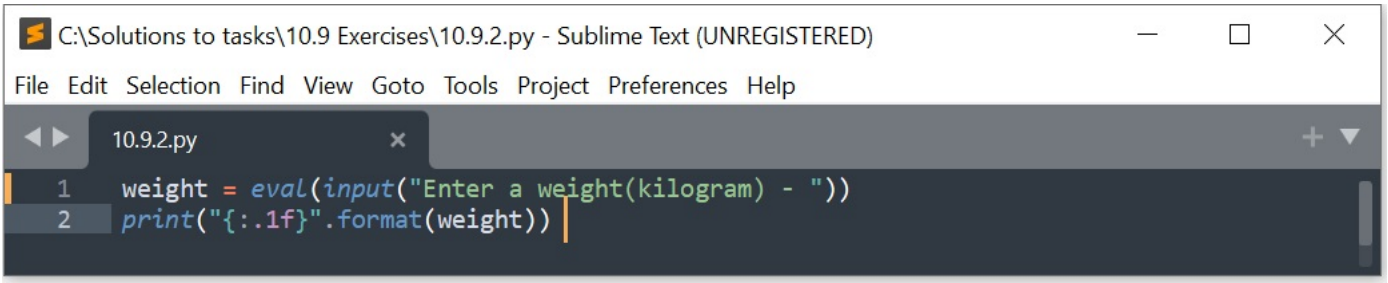
Miscellaneous Topics II

10.1 Exercises 10.9

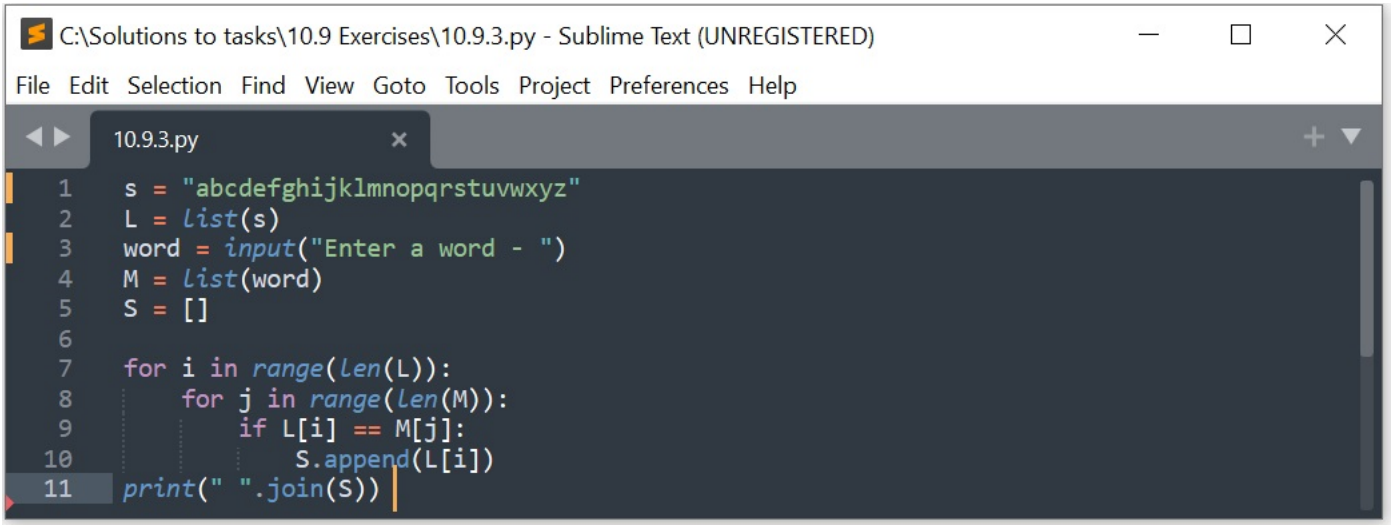
1. Write a program that uses **list** and **range** to create the list [3, 6, 9, . . . , 99].



2. Write a program that asks the user for a weight in kilograms. The program should convert the weight to kilograms, formatting the result to one decimal place.



3. Write a program that asks the user to enter a word. Rearrange all the letters of the word in alphabetical order and print out the resulting word. For example, *abracadabra* should become *aaaaabbcdrr*.



4. Write a program that takes a list of ten prices and ten products, applies an 11% discount to each of

the prices displays the output like below, right-justified and nicely formatted.

Apples $ 2.45
Oranges $ 18.02
. . .
Pears $ 120.03

C:\Solutions to tasks\10.9 Exercises\10.9.4.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.4.py

```
""" B IDLE Shell 3.11.4 the format() operator does not work correctly.
Run the program in py.exe"""

Prod = ["Apple","Pear","Orange","Strawberry","Plum","Apricot",
"Pineapple","Cherry","Grape","Gooseberry"]
Price = [12.36,45.78,112.12,125.32,69.14,47.65,82.36,72.18,54.95,173.54]
for i in range(10):
    print('{:12s} {:s} {:6.2f}'.format(Prod[i],'$', (Price[i]/1.11)))

input()
```

5. Use the following two lists and the format, method to create a list of card names in the format *card value* of *suit name* (for example, 'Two of Clubs').

suits = ['Hearts'♡, 'Diamonds'♢, 'Clubs'♣, 'Spades'♠]
values = ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven',
'Eight', 'Nine', 'Ten', 'Jack', 'Queen', 'King', 'Ace',]

C:\Solutions to tasks\10.9 Exercises\10.9.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.5.py

```
suits = ['Hearts', 'Diamonds', 'Clubs', 'Spades']
values = ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven',
          'Eight', 'Nine', 'Ten', 'Jack', 'Queen', 'King', 'Ace']
L = []
for i in range(len(values)):
    for j in range(len(suits)):
        x = "{}  {}".format(values[i],suits[j])
        L.append((x + " "))
print(L)
```

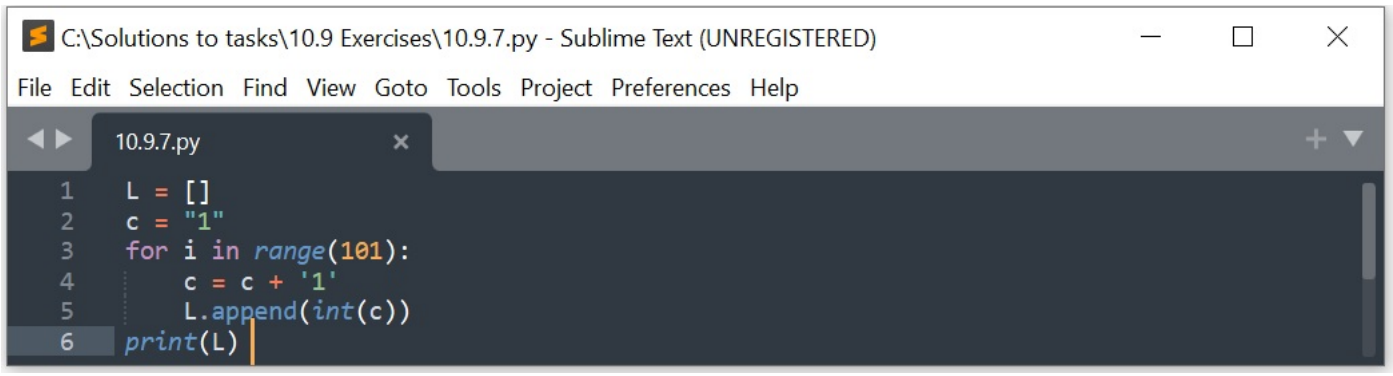
6. Write a program that uses a boolean flag variable in determining whether two lists have any items in common.

C:\Solutions to tasks\10.9 Exercises\10.9.6.py - Sublime Text (UNREGISTERED)

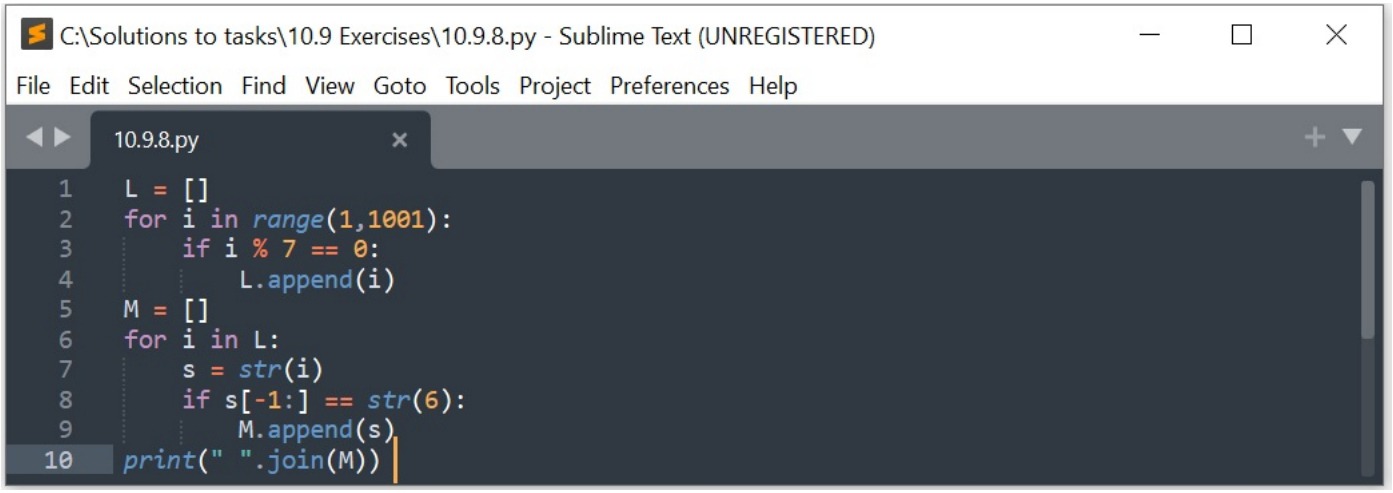
10.9.6.py

```
L1 = eval(input("Enter a first list: "))
L2 = eval(input("Enter a second list: "))
flag = 0
for ent in L2:
    if ent in L1:
        flag = 1
if flag == 1:
    print("There are items in common. ")
else:
    print("There are not items in common.")
```

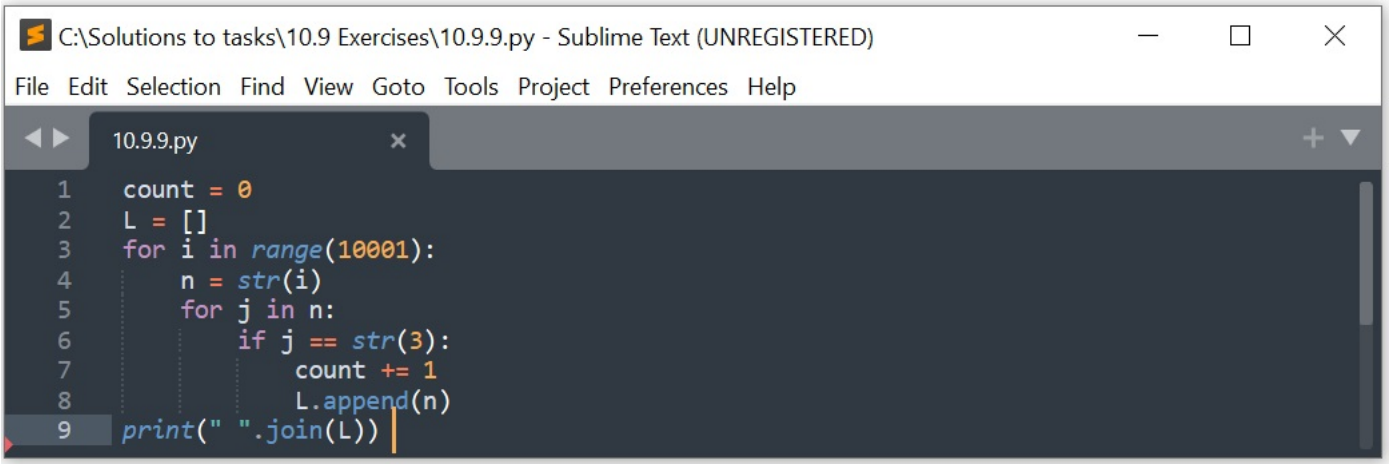
7. Write a program that creates the list [1, 11, 111, 1111, ..., 111 ... 1], where the entries have an ever increasing number of ones, with the last entry having 100 ones.



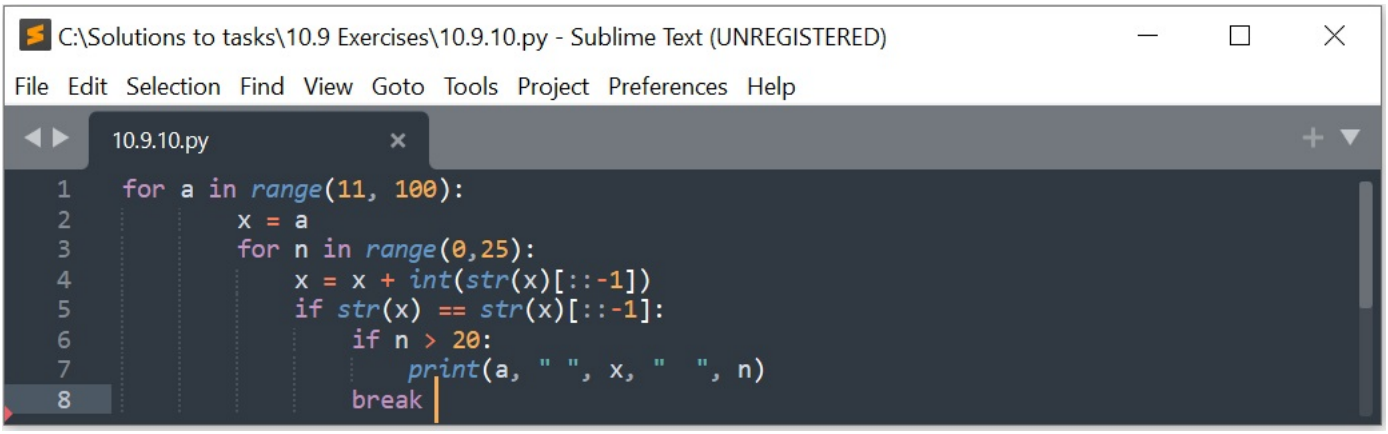
8.Write a program to find all numbers between 1 and 1000 that are divisible by 7 and end in a 6.



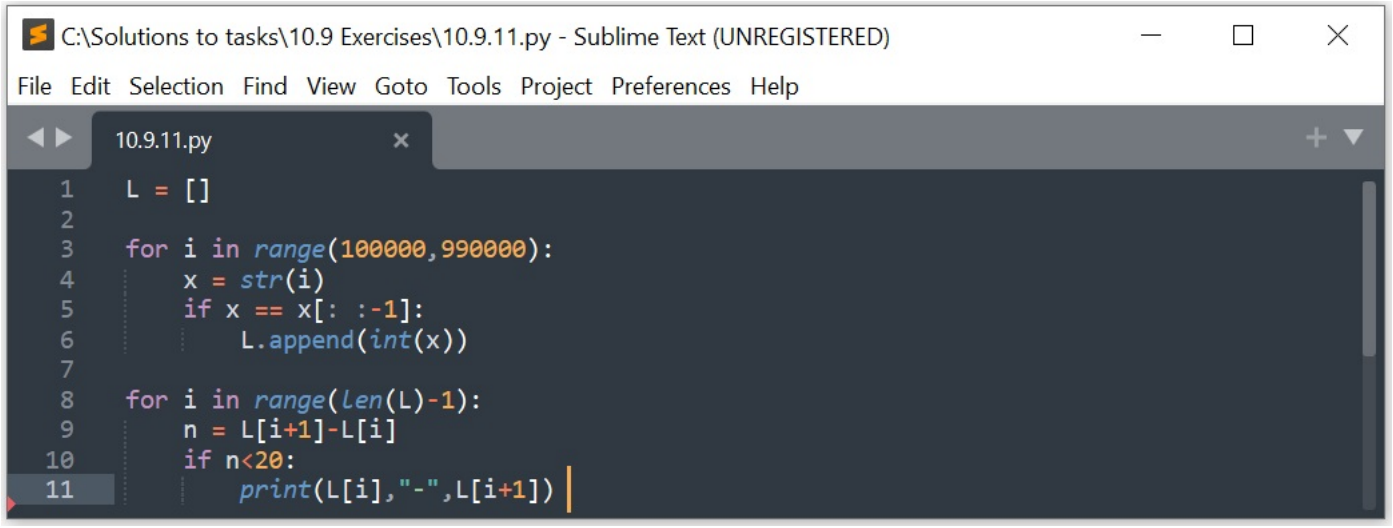
9.Write a program to determine how many of the numbers between 1 and 10000 contain the digit 3.



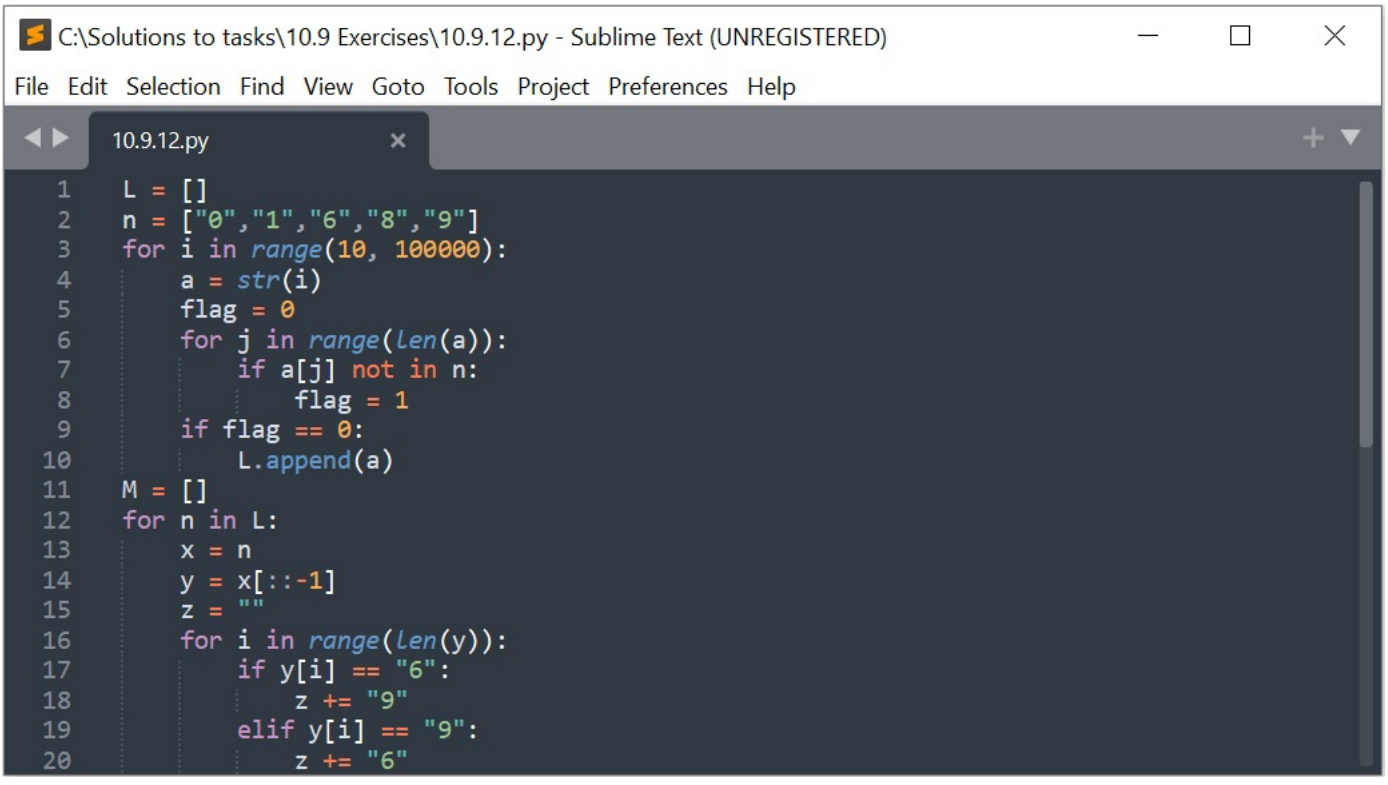
10. Adding certain numbers to their reversals sometimes produces a palindromic number. For instance, $241 + 142 = 383$. Sometimes, we have to repeat the process. For instance, $84 + 48 = 132$ and $132 + 231 = 363$. Write a program that finds both two-digit numbers for which this process must be repeated more than 20 times to obtain a palindromic number.



11. Write a program that finds all pairs of six-digit palindromic numbers that are less than 20 apart. One such pair is 199991 and 200002.



12. The number 1961 reads the same upside-down as right-side up. Print out all the numbers between 1 and 100000 that read the same upside-down as right-side up.



```
21          else:
22              z += y[i]
23      if x == z:
24          M.append(int(z))
25  print(M)
```

13. The number 99 has the property that if we multiply its digits together and then add the sum of its digits to that, we get back to 99. That is, $(9 \times 9) + (9 + 9) = 99$. Write a program to find all of the numbers less than 10000 with this property. (There are only nine of them.)

C:\Solutions to tasks\10.9 Exercises\10.9.13.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.13.py

```
1   L = []
2   for i in range(11,10001):
3       n = str(i)
4       ad = 0
5       mul = 1
6       for j in range(len(n)):
7           ad = ad + int(n[j])
8           mul = mul * int(n[j])
9           add = ad + mul
10      if i == add:
11          L.append(add)
12  print(L)
```

14. Write a program to find the smallest positive integer that satisfies the following property: If you take the leftmost digit and move it all the way to the right, the number thus obtained is exactly 3.5 times larger than the original number. For instance, if we start with 2958 and move the 2 all the way to the right, we get 9582, which is roughly 3.2 times the original number.

C:\Solutions to tasks\10.9 Exercises\10.9.14.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.14.py

```
1   for i in range(11,10000001):
2       n = str(i)
3       r = n[1 : :] + n[0]
4
5       div = int(r)/i
6       if div == 3.5:
7           print("i",i,"r",r)
```

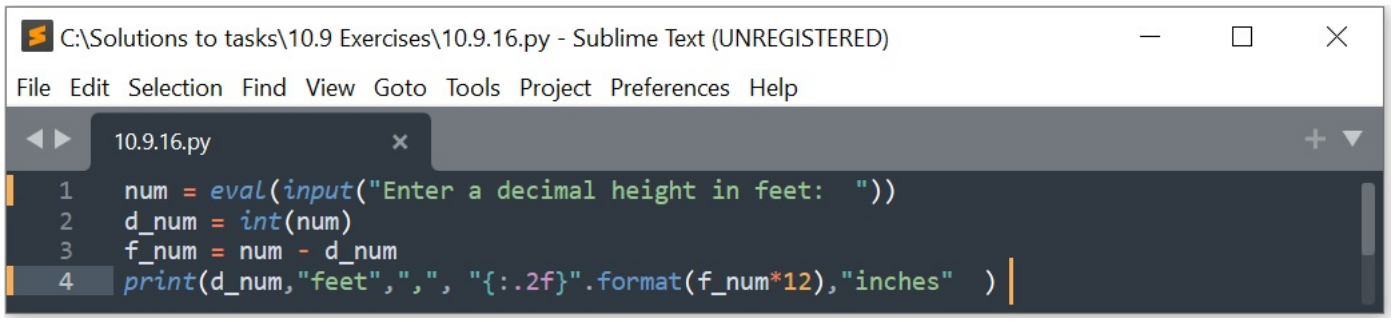
15. Write a program to determine how many zeroes 1000! ends with.

C:\Solutions to tasks\10.9 Exercises\10.9.15.py - Sublime Text (UNREGISTERED)

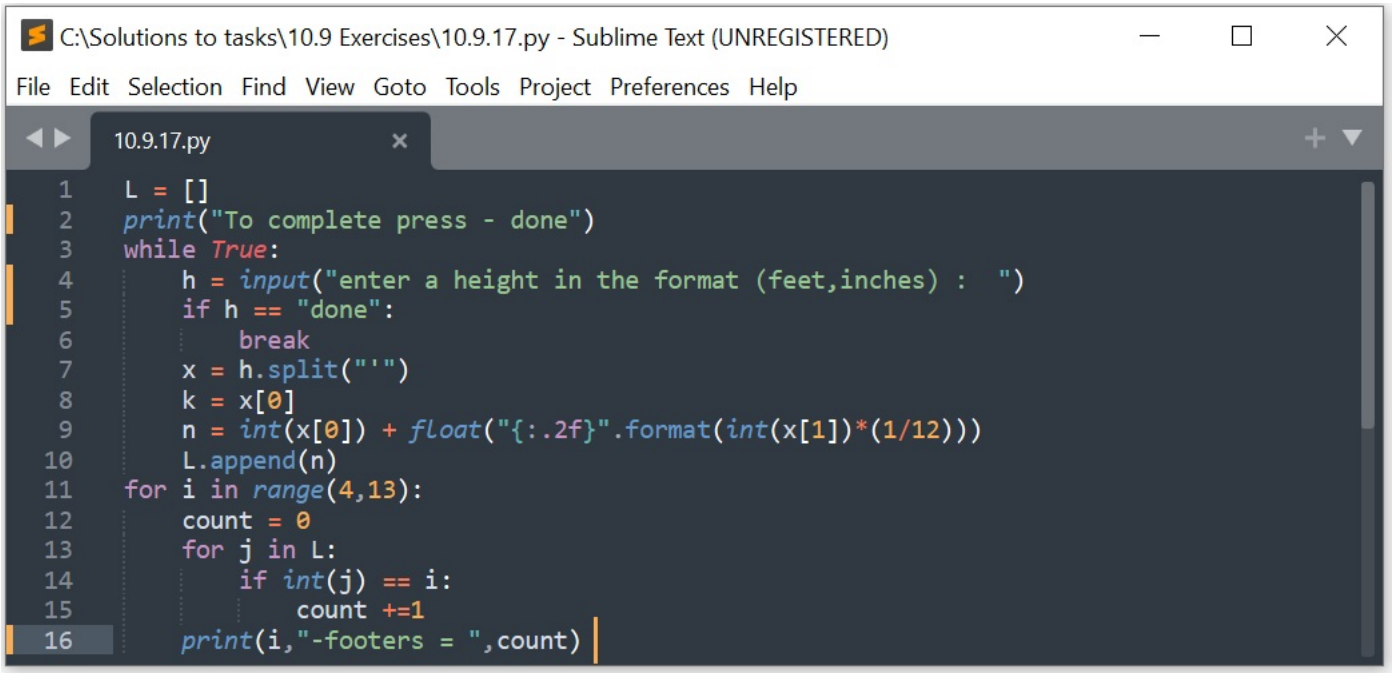
10.9.15.py

```
1   from math import factorial
2   f = factorial(1000)
3   n = str(f)
4   count = 0
5   for i in n[::-1]:
6           if i == "0":
7                   count += 1
8           else:
9                   break
10  print("The number of zeroes ", count)
```

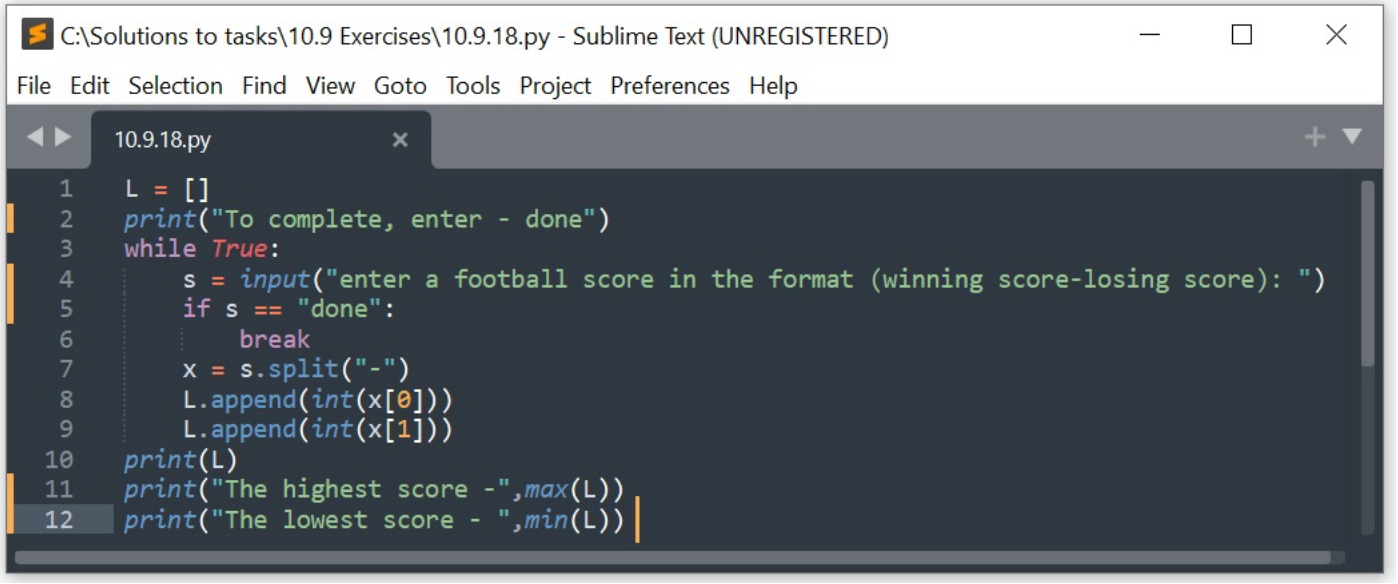
16. Write a program that converts a decimal height in feet into feet and inches. For instance, an input of *4.75* feet should become *4 feet, 9 inches*.



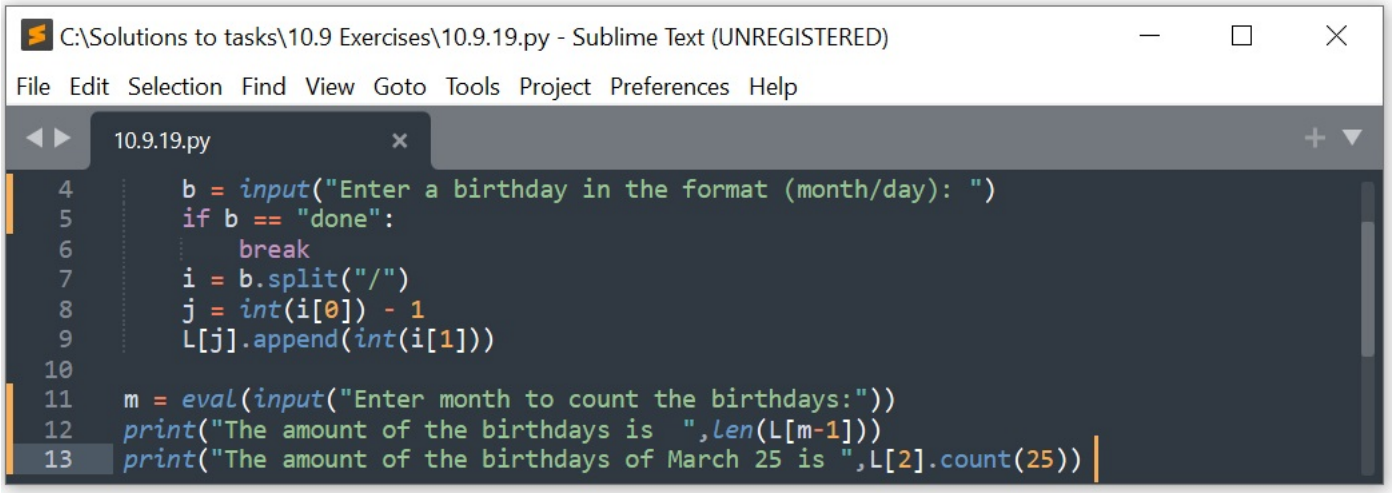
17. Write a program that repeatedly asks the user to enter a height in the format *feet'inches*"(like 5'11" or 6'3"). The user indicates they are done entering heights by entering *done*. The program should return a count of how many 4-footers, 5-footers, 6-footers, and 7-footers were entered.



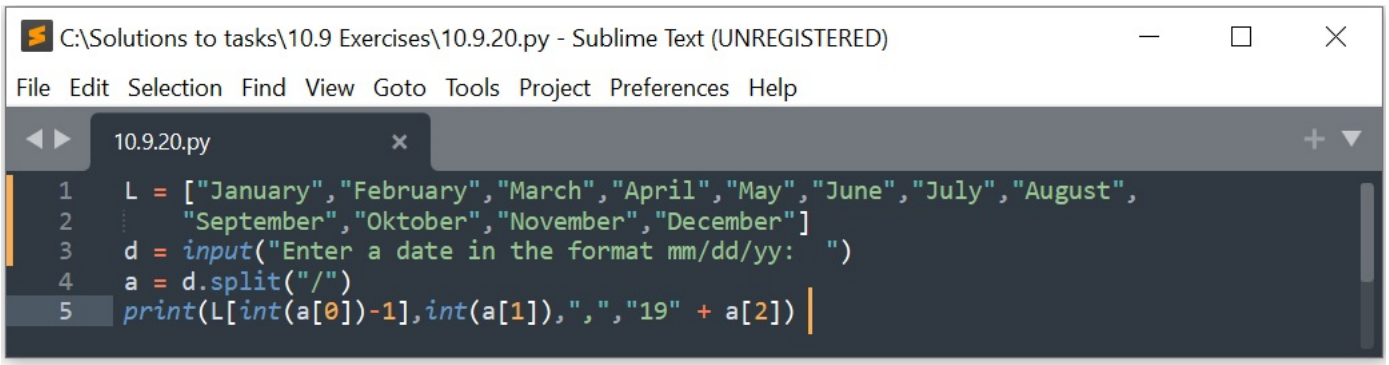
18. Write a program that repeatedly asks the user to enter a football score in the format winning *score-losing score* (like 27-13 or 21-3). The user indicates they are done entering scores by entering *done*. The program should then output the highest score and the lowest score out of all the scores entered.



19. Write a program that repeatedly asks the user to enter a birthday in the format *month/day* (like 12/25 or 2/14). The user indicates they are done entering birthdays by entering *done*. The program should return a count of how many of those birthdays are in February and how many are on the 25th of some month (any month).



20. Write a program that asks the user to enter a date in the format *mm/dd/yy* and converts it to a more verbose format. For example, *02/04/77* should get converted into *February 4, 1977*.



21. Write a program that asks the user to enter a fraction in the form of a string like '1/2' or '8/24'. The program should reduce the fraction to lowest terms and print out the result.

C:\Solutions to tasks\10.9 Exercises\10.9.21.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.21.py

```
f = input("Enter a fraction:  ")
x = f.split("/")
a = int(x[0])
b = int(x[1])
while a != 0 and b != 0:
    if a > b:
        a = a % b
    else:
        b = b % a
c = a + b
a = int(x[0])/c
b = int(x[1])/c
print("The fraction in the lowest terms is  ",int(a),"/",int(b))
```

22. Write a program to find all four solutions to the following problem: If a starfruit is worth $5, a mango is worth $3, and three oranges together cost $1, how many starfruits, mangoes, and oranges, totaling 100 , can be bought for $100?

C:\Solutions to tasks\10.9 Exercises\10.9.22.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.22.py

```
for x in range(0,101):
    for y in range(0,101):
        z = 100 - x - y
        if 5*x + 3*y + (1/3)*z == 100:
            print("Starfruits",x," Mangoes", y," Oranges", z)
```

23.The currency of a strange country has coins worth 7 cents and 11 cents. Write a program to determine the largest purchase price that cannot be paid using these two coins.

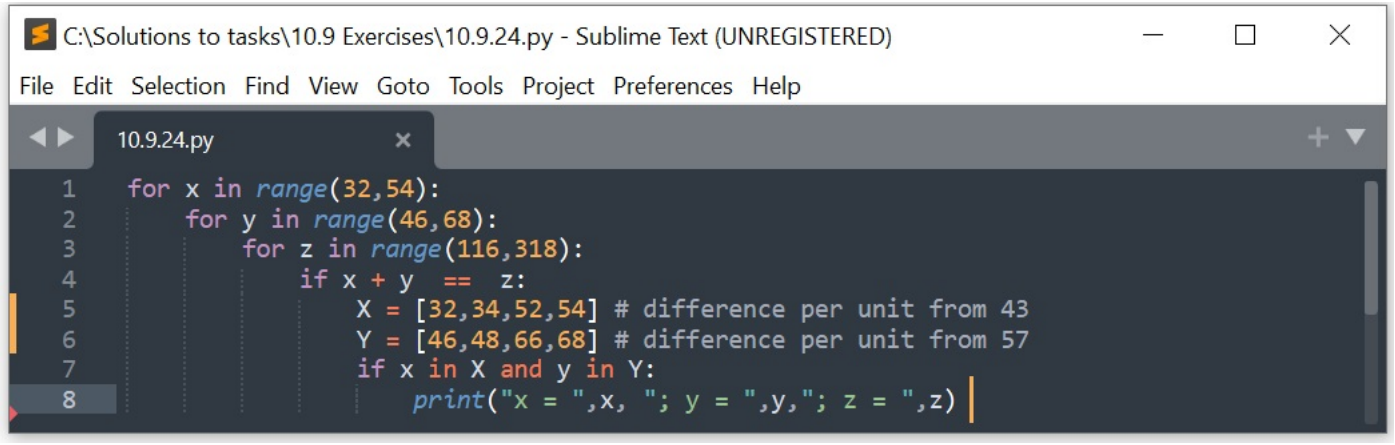
C:\Solutions to tasks\10.9 Exercises\10.9.23.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

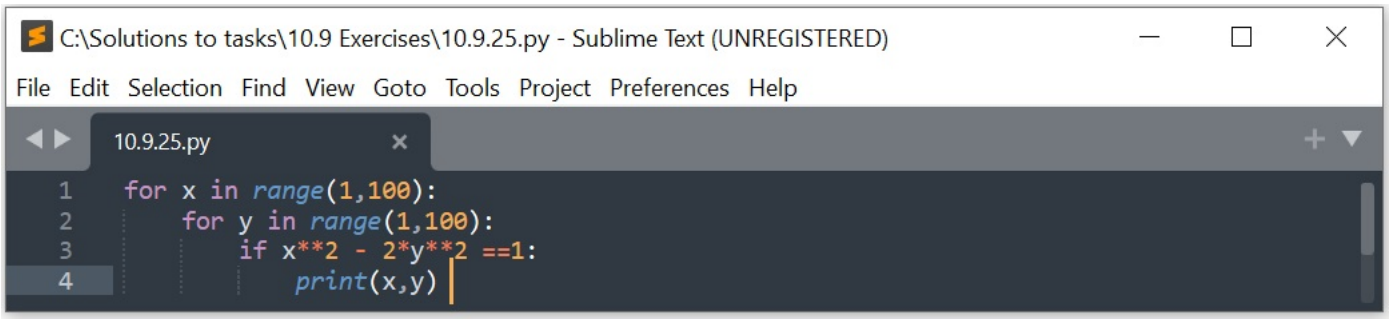
10.9.23.py

```
L = []
for v in range(1,100):
    for x in range(0,100):
        for y in range(0,100):
            if 7*x + 11*y == v:
                L.append(v)
M = []
for i in range(100):
    if i not in L:
        M.append(i)
print("The largest purchase price that cannot be paid - ",max(M), "cents.")
```

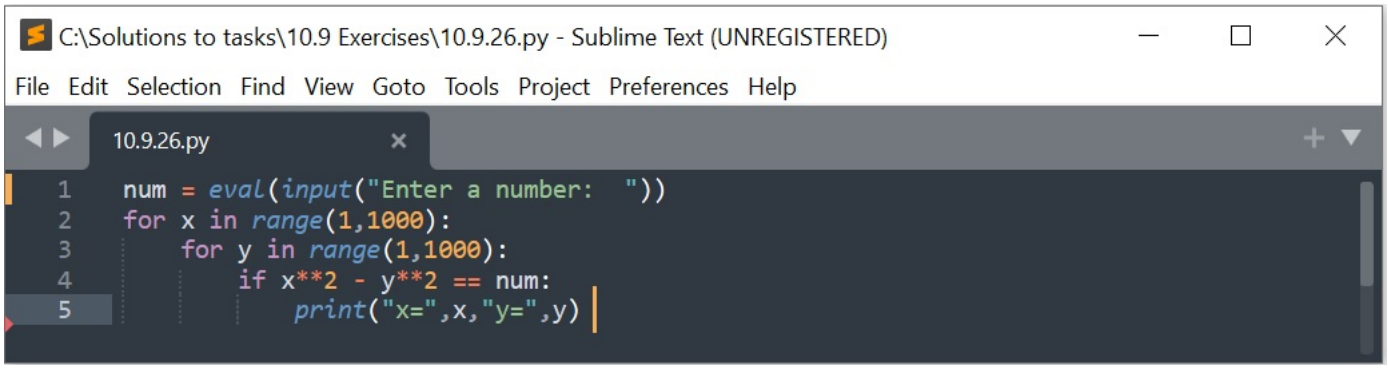
24. Here is an old puzzle you can solve using brute force by using a computer program to check all the possibilities: In the calculation 43 + 57 = 207, every digit is precisely one away from its true value. What is the correct calculation?



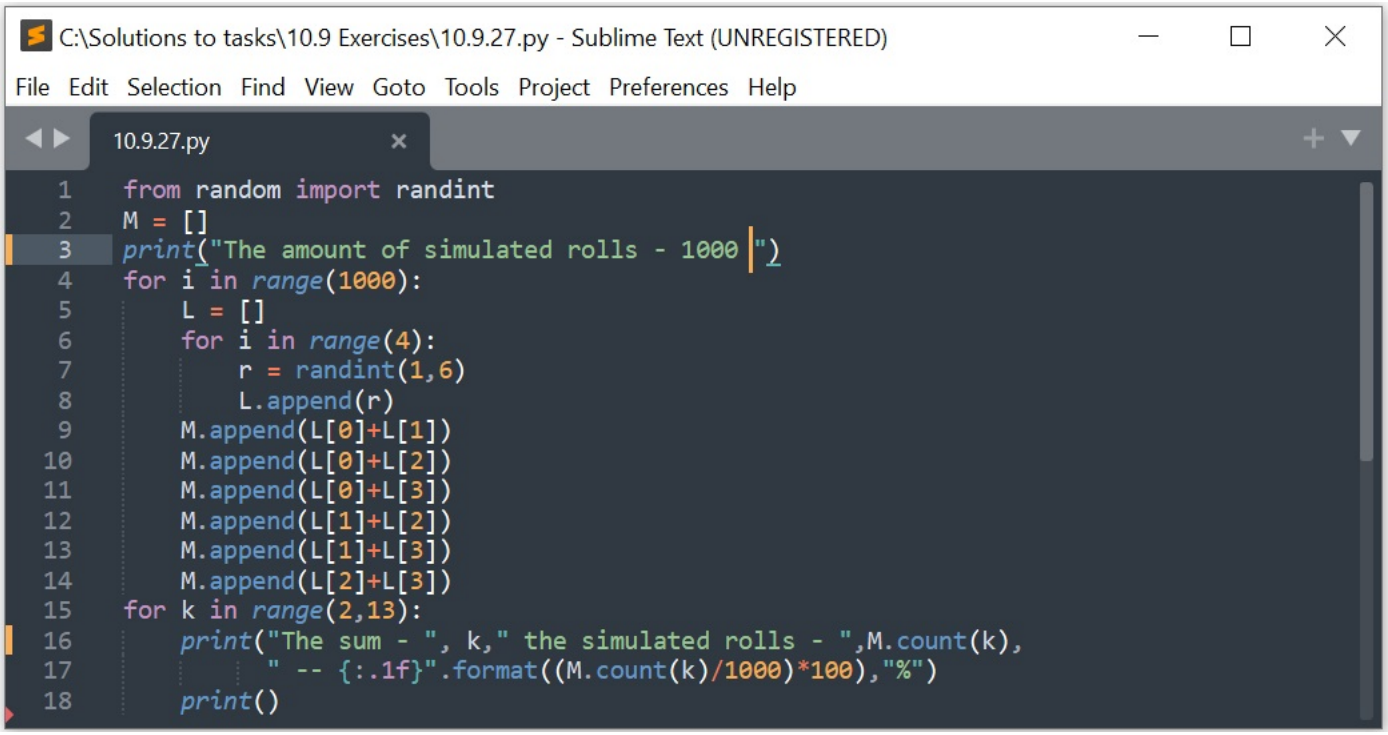
25. Write a program that finds all integer solutions to Pell's equation $x^2 - 2y^2 = 1$, where x and y are between 1 and 100.



26. Write a program that asks the user for a number and prints out all the ways to write the number as difference of two perfect squares, $x^2 - y^2$, where x and y are both between 1 and 1000. Writing a number as a difference of two squares leads to clever techniques for factoring large numbers.

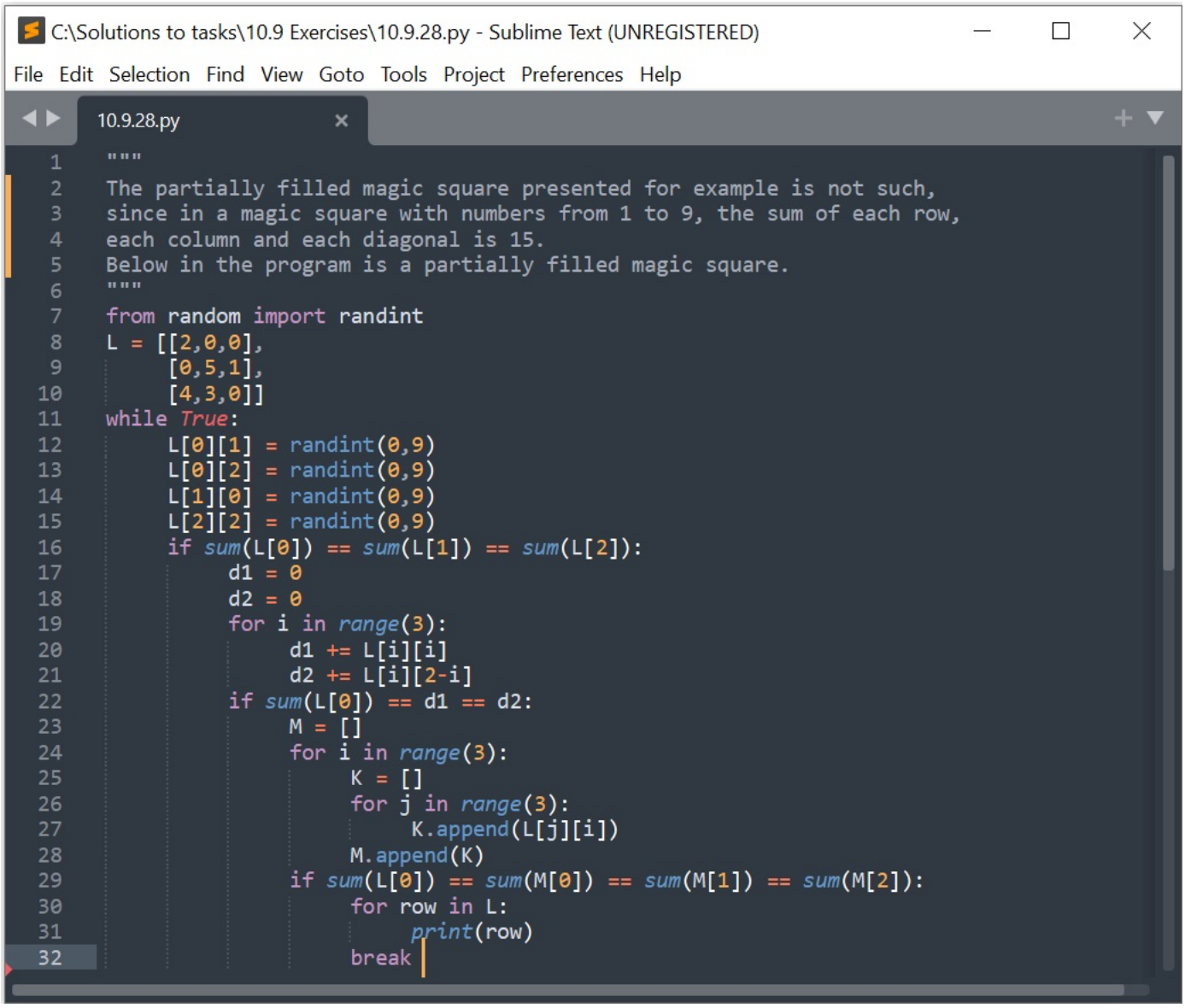


27. Write a program that simulates all possible rolls of four dice and for each simulated roll, finds the sums of pairs of dice. For instance, if the roll is 5 1 2 4, the sums are 6, 8, 9, 3 ,5, and 6. For each of the possible sums from 2 to 12, find the total number of simulated rolls in which the sum appears and what percentage of the simulated rolls had those sums appear. Output the totals and percentages, nicely formatted, with the percentages formatted to one decimal place. To check your work, you should find that the sum 2 comes up in 171 rolls, which is 13.2% of the rolls.



28. In a magic square, each row, each column, and both diagonals add up to the same number. A partially filled magic square is shown below. Write a program to check through all the possibilities to fill in the magic square.

5	_	_
_	6	2
3	8	_



29.The following is useful as part of a program to play *Minesweeper*. Suppose you have a 5 × 5 list that consists of zeros and M's. Write a program that creates a new 5×5 list that has M's in the same place, but the zeroes are replaced by counts of how many M's are in adjacent cells (adjacent either horizontally, vertically, or diagonally). An example is shown below. [Hint: short-circuiting may be helpful for avoiding index-out-of-range errors.]

0	M	0	M	0		1	M	3	M	1
0	0	M	0	0		1	2	M	2	1
0	0	0	0	0		2	3	2	1	0
M	M	0	0	0		M	M	2	1	1
0	0	0	M	0		2	2	2	M	1

C:\Solutions to tasks\10.9 Exercises\10.9.29.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.29.py

```
board = [[0, "M", 0, "M", 0],
         [0,  0, "M", 0,  0],
         [0,  0,  0,  0,  0],
         ["M","M",0,  0,  0],
         [0,  0,  0, "M", 0]]

new_board = [["0" for _ in range(5)] for _ in range(5)]
for row in range(5):
    for col in range(5):
        if board[row][col] == "M":
            new_board[row][col] = "M"
        else:
            c = 0
            for i in range(max(0,row - 1),min(5,row + 2)):
                for j in range(max(0,col - 1), min(5,col + 2)):
                    if board[i][j] == "M":
                        c += 1
            new_board[row][col] = " " +str(c)

for r in new_board:
    print("    ".join(r))
```

30. Pascal's triangle is shown below. On the outside are 1's and each other number is the sum of the two numbers directly above it. Write a program to generate Pascal's triangle. Allow the user to specify the number of rows. Be sure that it is nicely formatted, like below.

```
                1
             1     1
          1     2     1
       1     3     3     1
    1     4     6     4     1
 1     5    10    10     5     1
```

C:\Solutions to tasks\10.9 Exercises\10.9.30.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.30.py

```
"""
IDLE Shell 3.11.4- the output incorrect.
Use py.exe
"""
n = eval(input("Enter a height of Pascal's triangle: "))
L = []

for i in range(0, n):
    r = ["1"] * (i + 1)
    for j in range(i + 1):
        if j != 0 and j != i:
            r[j] = str(int(L[i-1][j-1]) + int(L[i-1][j]))
    L.append(r)
for i in range(len(L)):
    s = " ".join(L[i])
    print("{:^100s}".format(s))
input()
```

31. Given two dates entered as strings in the form mm/dd/yyyy where the years are between 1901 and 2099, determine how many days apart they are. Here is a bit of information that may be useful: Leap years between 1901 and 2099 occur exactly every four years, starting at 1904. February has 28 days, 29 during a leap year. November, April, June, and September each have 30 days. The other months have 31 days.

C:\Solutions to tasks\10.9 Exercises\10.9.31.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

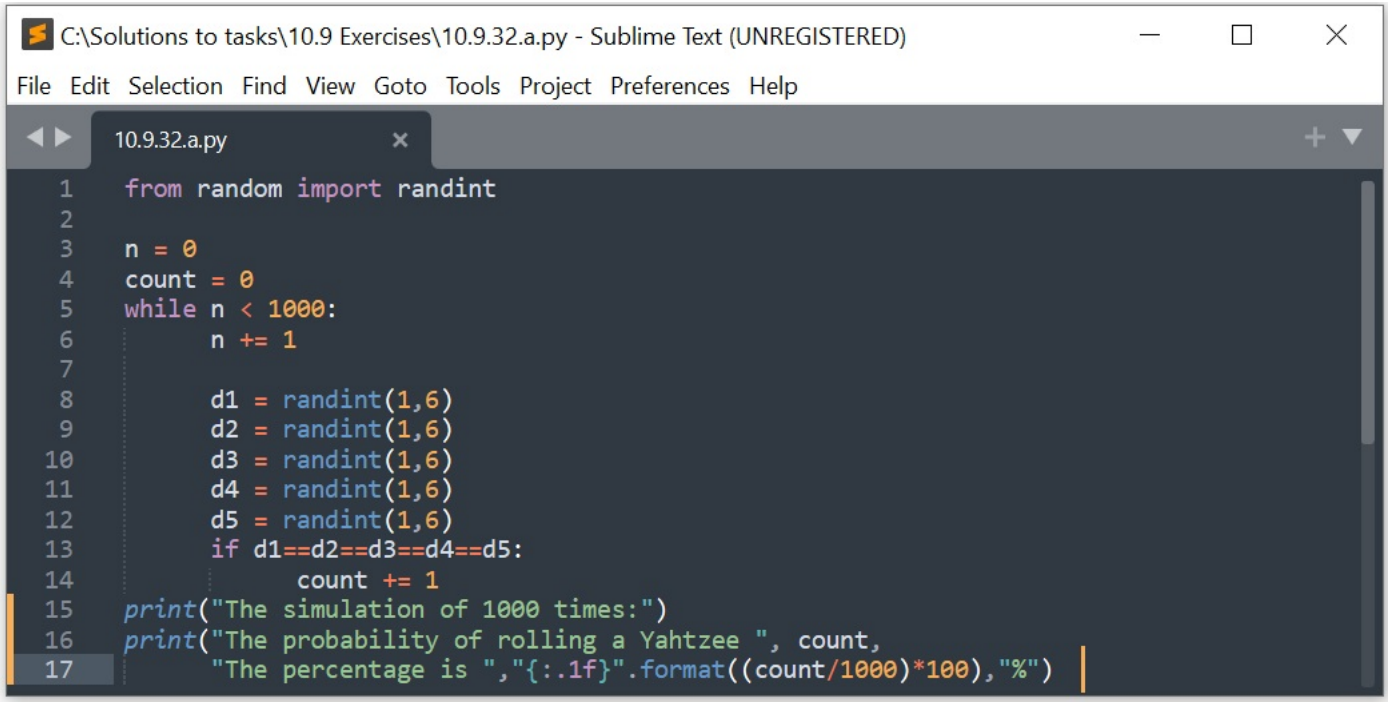
10.9.31.py

```
#The program written by Igor Antonenko

data1 = input("Enter a first data( mm/dd/yyyy) between 1901 - 2099 - ")
data2 = input("Enter a second data( mm/dd/yyyy) between 1901 - 2099 - ")
print()
year = [31,28,31,30,31,30,31,31,30,31,30,31]
data1_split = data1.split("/")
data2_split = data2.split("/")
month_1 = int(data1_split[0])
day_1 = int(data1_split[1])
year_1 = int(data1_split[2])
month_2 = int(data2_split[0])
day_2 = int(data2_split[1])
year_2 = int(data2_split[2])
# the same year
if year_2 - year_1 == 0:
    # the same month
    if year_1 % 4 != 0 or year_1 % 4 == 0 and month_1 == month_2 :
        days = day_2 - day_1
    # if not leap year
    elif year_1 % 4 != 0:
        days = (year[month_1 - 1] - day_1) + year[(month_1 ):(month_2 - 1)] + day_2
    # if leap year
    elif year_1 % 4 == 0:
        if month_1 > 2 and month_2 > 2:
            days = (year[month_1 - 1] - day_1) + sum(year[(month_1 ):(month_2 - 1)])\
            + day_2
        elif month_1 == 1 and month_2 > 2:
            days = (year[month_1 - 1] - day_1) + sum(year[(month_1 ):(month_2 - 1)])\
             + day_2 + 1
        elif month_1 == 1 and month_2 == 2 :
            days = (year[month_1 - 1] - day_1) + day_2
elif year_2 - year_1 >= 1:
    days_years = ((year_2 - year_1) - 1) * 365

    leap_years = 0
    for leap in range(year_1 + 1, year_2 ):
        if leap % 4 == 0:
            leap_years += 1

    if year_1 % 4 != 0 and year_2 % 4 != 0:
        days_year_1 = 365 - year[month_1 - 1] - day_1
        days_year_2 = sum(year[:(month_2 - 1)]) + day_2
        days = days_year_1 + days_year_2 + days_years + leap_years

    elif year_1 % 4 == 0 and year_2 % 4 != 0:
        if month_1 > 2:
            days_year_1 = 365 - sum(year[:month_1 - 1]) - day_1
        elif month_1 <= 2:
            days_year_1 = 366 - sum(year[:month_1 - 1]) - day_1
        days_year_1 = 366 - sum(year[:month_1 - 1]) - day_1
        days_year_2 = sum(year[:(month_2 - 1)]) + day_2
        days = days_year_1 + days_year_2 + days_years + leap_years
```

```
54
55      elif year_1 % 4 != 0 and year_2 % 4 == 0:
56          days_year_1 = 365 - sum(year[:month_1 - 1]) - day_1
57          if month_2 == 1:
58              days_year_2 = sum(year[:(month_2 - 1)]) + day_2
59          elif month_2 == 2:
60              if day_2 <= 28:
61                  days_year_2 = sum(year[:(month_2 - 1)]) + day_2
62              elif day_2 == 29:
63                  days_year_2 = sum(year[:(month_2 - 1)]) + day_2
64          elif month_2 > 2:
65              days_year_2 = sum(year[:(month_2 - 1)]) + day_2 + 1
66
67          days = days_year_1 + days_year_2 + days_years + leap_years
68
69      elif  year_1 % 4 == 0 and year_2 % 4 == 0:
70          if month_1 > 2:
71              days_year_1 = 365 - sum(year[:month_1 - 1]) - day_1
72          elif month_1 <= 2:
73              days_year_1 = 366 - sum(year[:month_1 - 1]) - day_1
74          if month_2 == 1:
75              days_year_2 = sum(year[:(month_2 - 1)]) + day_2
76
77          elif month_2 == 2:
78              if day_2 <= 28:
79                  days_year_2 = sum(year[:(month_2 - 1)]) + day_2
80              elif day_2 == 29:
81                  days_year_2 = sum(year[:(month_2 - 1)]) + day_2
82          elif month_2 > 2:
```

32. Monte Carlo simulations can be used to estimate all sorts of things, including probabilities of coin flip and dice events. As an example, to estimate the probability of rolling a pair of sixes with two dice, we could use random integers to simulate the dice and run the simulation thousands of times, counting what percentage of the time a pair of sixes comes up.

(a). Estimate the probability of rolling a *Yahtzee* in a single roll of five dice. That is estimate the probability that when rolling five dice they all come out to be the same number.



(b). Estimate the probability of rolling a large straight in a single roll of five dice. A large straight is a roll where the dice come out 1-2-3-4-5 or 2-3-4-5-6 in any order.

C:\Solutions to tasks\10.9 Exercises\10.9.32.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.32.b.py

```
from random import randint
L1 = [1,2,3,4,5]
L2 = [2,3,4,5,6]
count = 0

for i in range(10000):
    L = []
    for j in range(5):
        d = randint(1,6)
        L.append(d)
        L.sort()
        if L == L1 or L == L2:
            count += 1

print("The probability of a large straight is ",round((count/10000)*100,2), "%")
```

(c).Estimate the average longest run of heads or tails when flipping a coin 200 times.

C:\Solutions to tasks\10.9 Exercises\10.9.32.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.32.c.py

```
from random import choice

N = []
for _ in range(100):
    L = []
    M = ["H","T"]
    for _ in range(200):
        L.append(choice(M))

    longest = 0
    s = 0
    for i in L:
        if i == "H":
            s += 1
        else:
            s = 0

        if s > longest:
            longest = s
    N.append(longest)
print(" the average longest run of heads is",sum(N)/100)
```

(d). Estimate the average number of coin flips it takes before five heads in a row come up.

C:\Solutions to tasks\10.9 Exercises\10.9.32.d.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.32.d.py

```
from random import choice
M = ["H","T"]
L = []

for _ in range(100):
    s = 0
    n = 0
    while s != 5:
        f = choice(M)
        n += 1
        if f == "H":
            s += 1
            if s == 5:
                L.append((n-5))
        else:
            s = 0
print(sum(L)/100)
```

(e). Estimate the average number of coin flips it takes before the string s comes up, where s is a string of heads and tails, like $HHTTH$.

C:\Solutions to tasks\10.9 Exercises\10.9.32.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

10.9.32.e.py

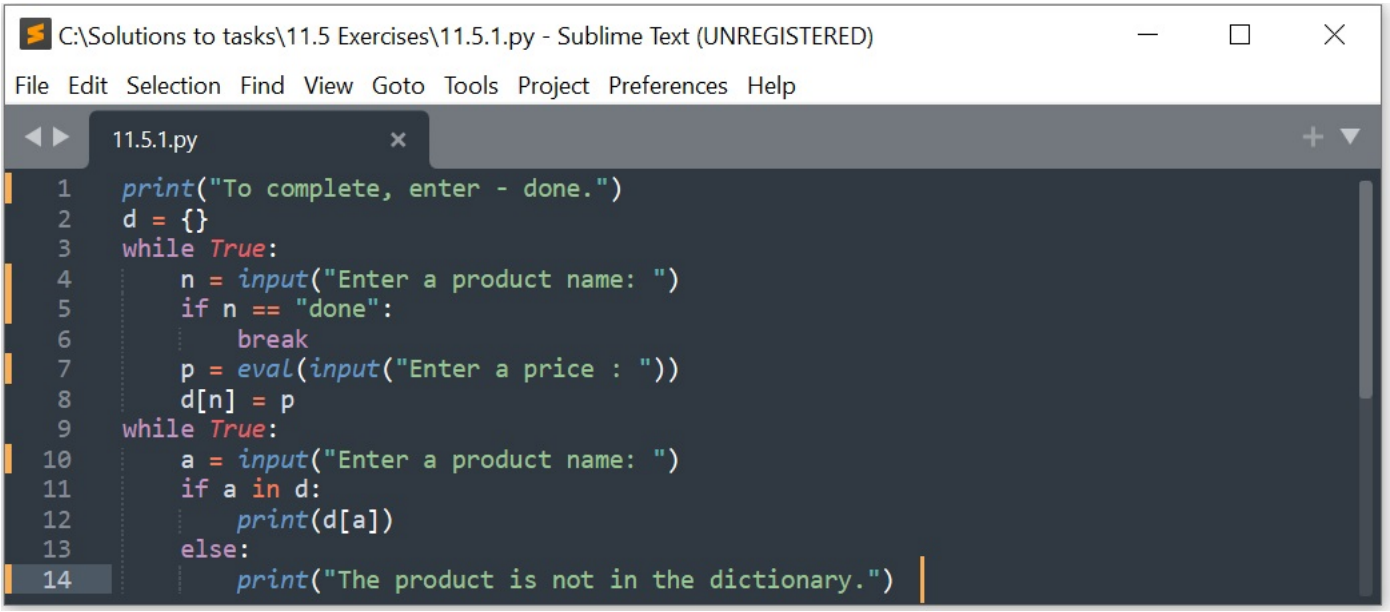
```
from random import choice
M = ["H","T"]
s = "HHTTH"
L = []
for _ in range(100):
    count = 0
    sequence = ""
    while True:
        count += 1
        coin_flip = choice(M)
        sequence += coin_flip
        if len(sequence) >= 5:
            if sequence[-5:] == s:
                L.append(count)
                break
print("the average number of coin flips of the run,like HHTTH, is ", sum(L)/100)
```

Chapter 11

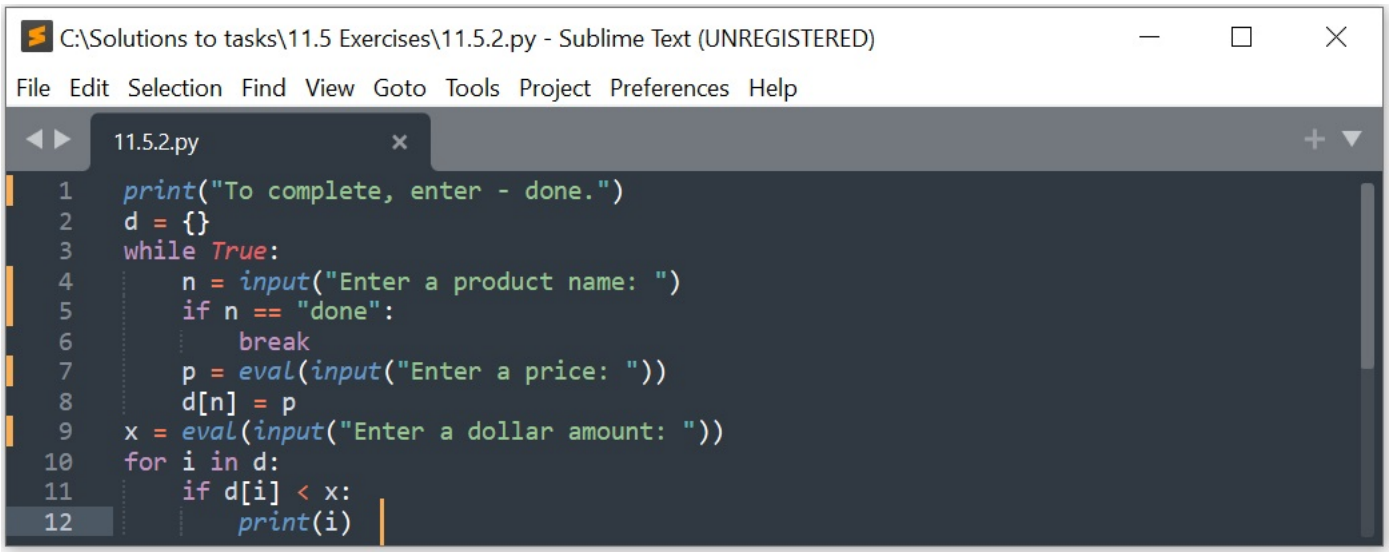
Dictionaries

11.1 Exercises 11.5

1. Write a program that repeatedly asks the user to enter product names and prices. Store all of these in a dictionary whose keys are the product names and whose values are the prices. When the user is done entering products and prices, allow them to repeatedly enter a product name and print the corresponding price or a message if the product is not in the dictionary.

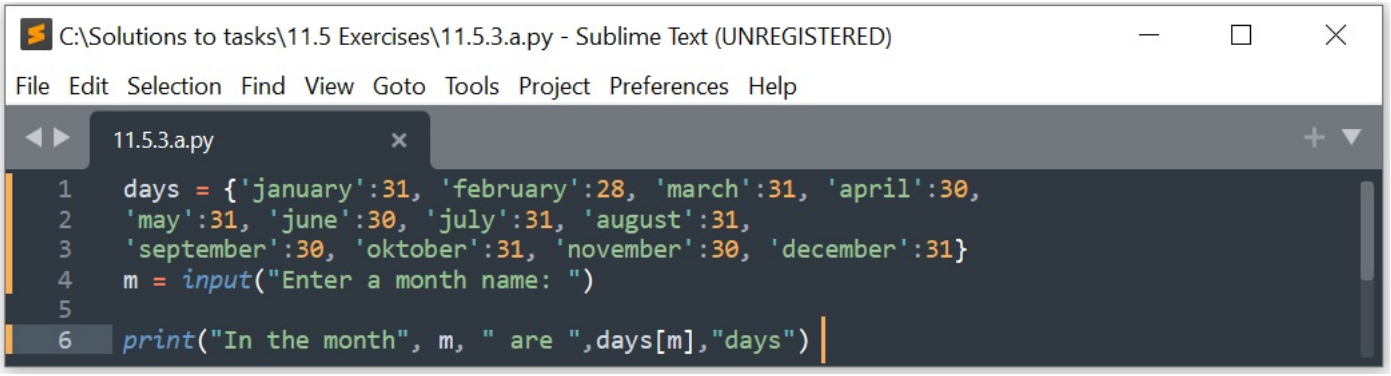


2. Using the dictionary created in the previous problem, allow the user to enter a dollar amount and print out all the products whose price is less than that amount.

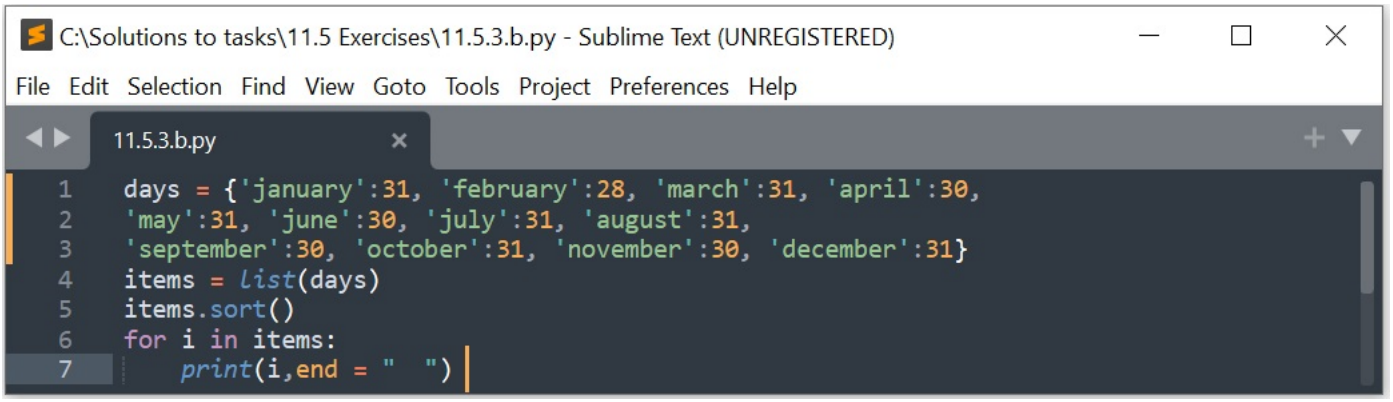


3. For this problem, use the dictionary from the beginning of this chapter whose keys are month names and whose values are the number of days in the corresponding months.

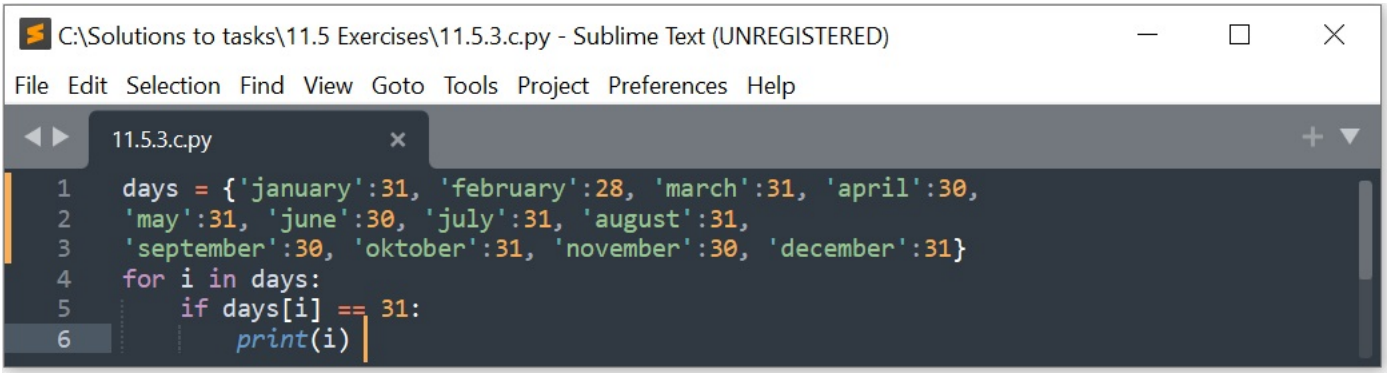
(a) Ask the user to enter a month name and use the dictionary to tell them how many days are in the month.



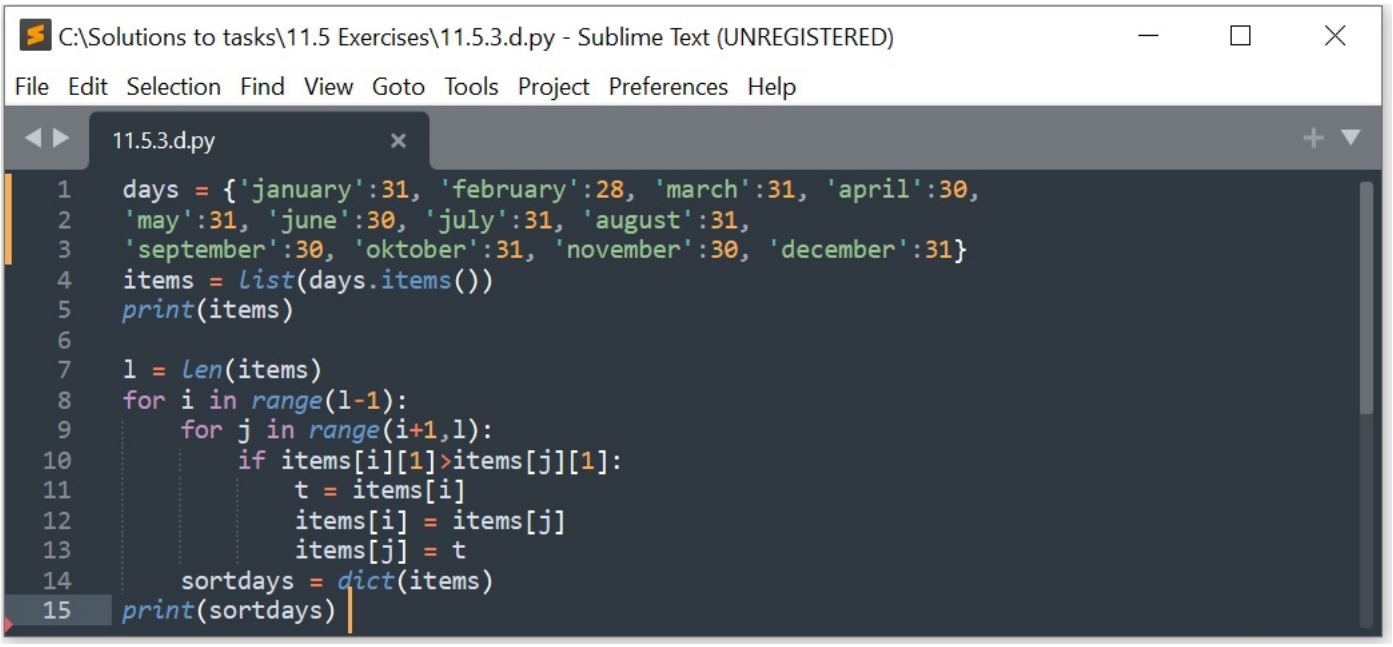
(b) Print out all of the keys in alphabetical order.



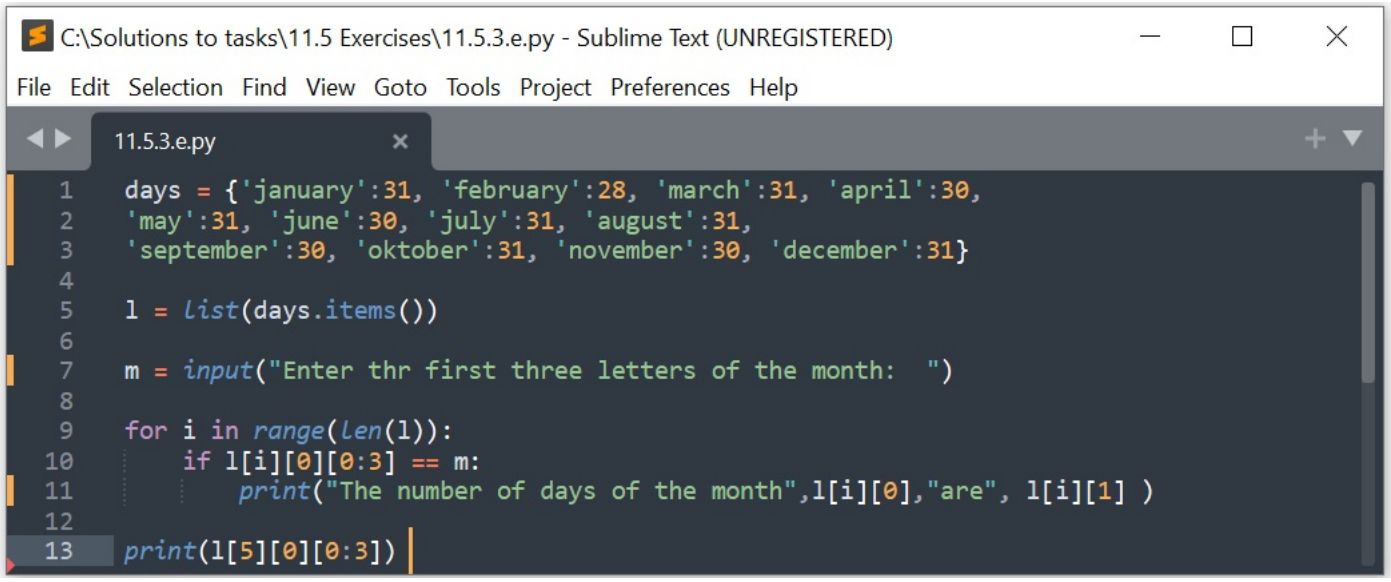
(c) Print out all of the months with 31 days.



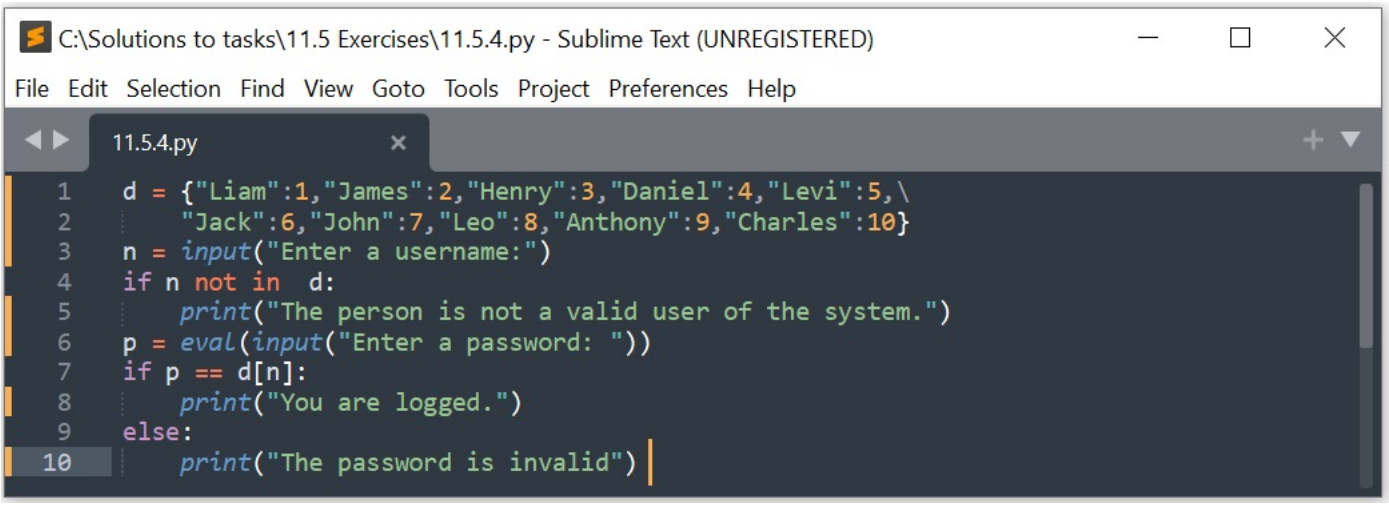
(d) Print out the (key-value) pairs sorted by the number of days in each month.



(e) Modify the program from part (a) and the dictionary so that the user does not have to know how to spell the month name exactly. That is, all they have to do is spell the first three letters of the month name correctly.

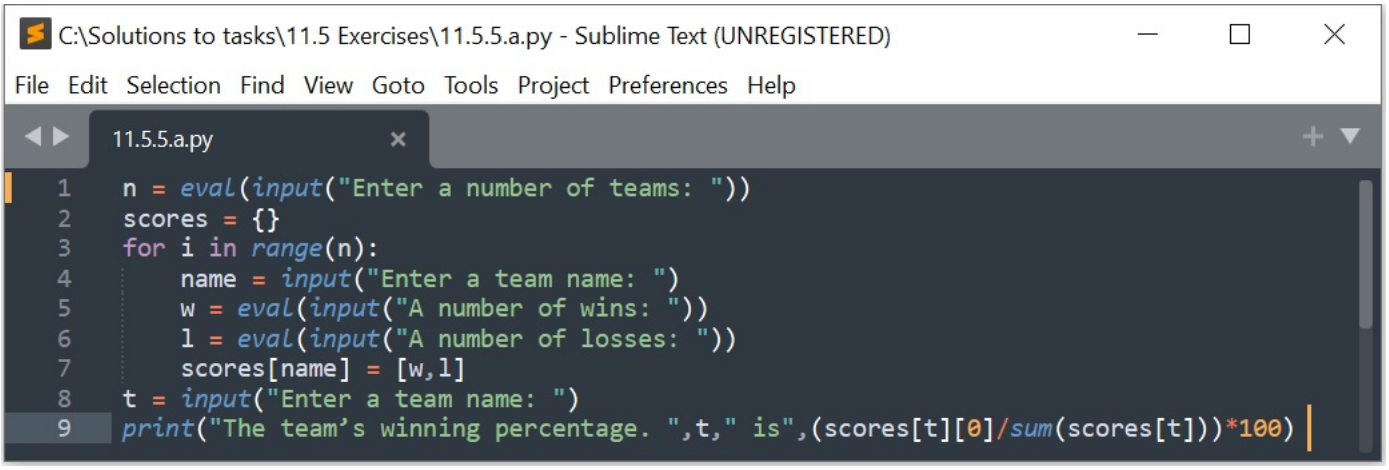


4. Write a program that uses a dictionary that contains ten user names and passwords. The program should ask the user to enter their username and password. If the username is not in the dictionary, the program should indicate that the person is not a valid user of the system. If the username is in the dictionary, but the user does not enter the right password, the program should say that the password is invalid. If the password is correct, then the program should tell the user that they are now logged in to the system.

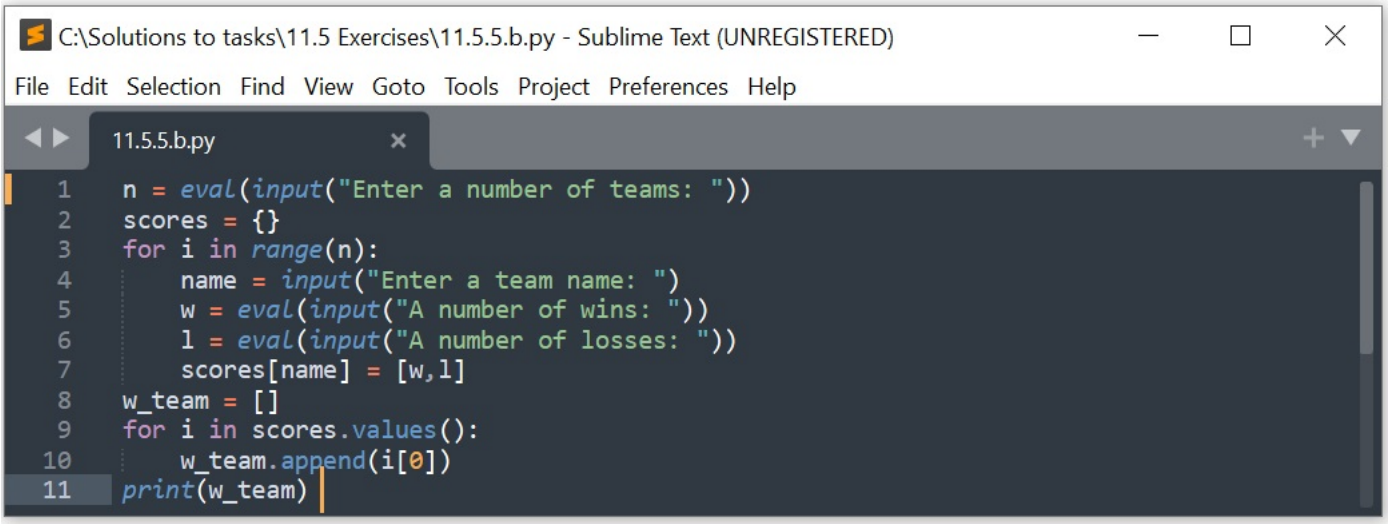


5.Repeatedly ask the user to enter a team name and the how many games the team won and how many they lost. Store this information in a dictionary where the keys are the team names and the values are lists of the form [*wins*, *losses*].

(a). Using the dictionary created above, allow the user to enter a team name and print out the team's winning percentage.



(b). Using the dictionary, create a list whose entries are the number of wins of each team.



(c). Using the dictionary, create a list of all those teams that have winning records.

C:\Solutions to tasks\11.5 Exercises\11.5.5.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.5.c.py

```
n = eval(input("Enter a number of teams: "))
scores = {}
for i in range(n):
    name = input("Enter a name team: ")
    w = eval(input("A number of wins: "))
    l = eval(input("A number of losses: "))
    scores[name] = [w,l]
w_rec = []
for i in scores:
    if scores[i][0] > 0:
        w_rec.append(i)
print(w_rec)
```

6. Repeatedly ask the user to enter game scores in a format like *team1 score1 - team2 score2*. Store this information in a dictionary where the keys are the team names and the values are lists of the form [*wins*, *losses*].

C:\Solutions to tasks\11.5 Exercises\11.5.6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.6.py

```
scores_dict = {}
print("To complete, press - q")
print("Enter game scores in a format like team1 score1 - team2 score2. ")
while True:
    game_score = input(" > ")
    if game_score == "q":
        break
    L = game_score.split(" ")
    team1 = L[0]
    score1 = L[1]
    team2 = L[3]
    score2 = L[4]
    if team1 not in scores_dict:
        scores_dict[team1] = [0,0]
    if team2 not in scores_dict:
        scores_dict[team2] = [0,0]

    if score1 > score2:
        scores_dict[team1][0] += 1
        scores_dict[team2][1] += 1
    elif score1 < score2:
        scores_dict[team2][0] += 1
        scores_dict[team1][1] += 1
print(scores_dict)
```

7.Create a 5 × 5 list of numbers. Then write a program that creates a dictionary whose keys are the numbers and whose values are the how many times the number occurs. Then print the three most common numbers.

C:\Solutions to tasks\11.5 Exercises\11.5.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.7.py

```
from random import randint
L = []
for i in range(5):
    M = []
    for j in range(5):
        M.append(randint(1,10))
    L.append(M)
M = []
for row in L:
    for j in row:
        M.append(j)
M.sort()

d = {}
for i in range(1,11):
    d[str(i)] = M.count(i)
print(d)
```

8.Using the card dictionary from earlier in this chapter, create a simple card game that deals two players three cards each. The player with the highest card wins. If there is a tie, then compare the second highest card and, if necessary, the third highest. If all three cards have the same value, then the game is a draw.

C:\Solutions to tasks\11.5 Exercises\11.5.8.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.8.py

```
# "11 - Jack, 12 - Queen, 13 - King, 14 - Ace"
from random import shuffle, randint
suits = ['spades', 'clubs', 'hearts', 'diamonds']
deck = []
deck = [{'value':i, 'suit':c}
for c in suits
for i in range(2,15)]
shuffle(deck)
player1 = []
player2 = []

while True:
    for i in range(3):
        player1.append(deck[randint(2,15)]["value"])
    for i in range(3):
        player2.append(deck[randint(2,15)]["value"])
    player1.sort()
    player2.sort()

    if player1 == player2:
        print("Tie")
        break
    if player1[2] > player2[2]:
        print("Player1 won. Score is",player1[2]," > ",player2[2])
        break
    if player1[2] < player2[2]:
        print("Player2 won.Score is",player1[2]," < ",player2[2])
        break
```

```
    if player1[2] == player2[2]:
        if player1[1] > player2[1]:
            print("Player1 won. Score is",player1[1]," > ",player2[1])
            break
        if player1[1] < player2[1]:
            print("Player2 won. Score is",player1[1]," < ",player2[1])
            break

    if player1[1] == player2[1]:
        if player1[0] > player2[0]:
            print("Player1 won. Score is",player1[0]," > ",player2[0])
            break
        else:
            print("Player2 won. Score is",player1[0]," < ",player2[0])
            break
```

9. Using the card dictionary from earlier in the chapter, deal out three cards. Determine the following:

(a). If the three cards form a flush (all of the same suit)

C:\Solutions to tasks\11.5 Exercises\11.5.9.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.9.a.py

```
from random import randint,shuffle
deck = [{'value':i, 'suit':c}
for c in [['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
shuffle(deck)
L = []
for i in range(3):
    L.append(deck[randint(0,51)]["suit"])
if L[0] == L[1] == L[2]:
    print("The flush - ",L[0])
else:
    print("No flush")
```

(b). If there is a three-of-a-kind (all of the same value)

C:\Solutions to tasks\11.5 Exercises\11.5.9.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.9.b.py

```
from random import randint,shuffle
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
shuffle(deck)
L = []
for i in range(3):
    L.append(deck[randint(0,51)]["value"])
#print(L)
if L[0] == L[1] == L[2]:
    print("The three-of-a-kind",L[0])
else:
    print("No")
```

(c).If there is a pair, but not three-of-a-kind

C:\Solutions to tasks\11.5 Exercises\11.5.9.c.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.9.c.py

```
from random import randint,shuffle
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
shuffle(deck)
L = []
for i in range(3):
    L.append(deck[randint(0,51)]["value"])
if L.count(L[0]) == 2 or L.count(L[2]) == 2:
    print("The pair ")
else:
    print("No")
```

(d). If the three cards form a straight (all in a row, like (2, 3, 4) or (10, Jack, Queen))

C:\Solutions to tasks\11.5 Exercises\11.5.9.d.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.9.d.py

```
from random import randint,shuffle
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
shuffle(deck)
L = []
for i in range(3):
    L.append(deck[randint(0,51)]["value"])
L.sort()
if L[2] - L[1] == 1 and L[1] - L[0] == 1:
    print("The straight", L[0],L[1],L[2])
else:
    print("No")
```

10. Using the card dictionary from earlier in the chapter run a Monte Carlo simulation to estimate the probability of being dealt a flush in a five card hand. See Exercise 32 of Chapter 10 for more about Monte Carlo simulations.

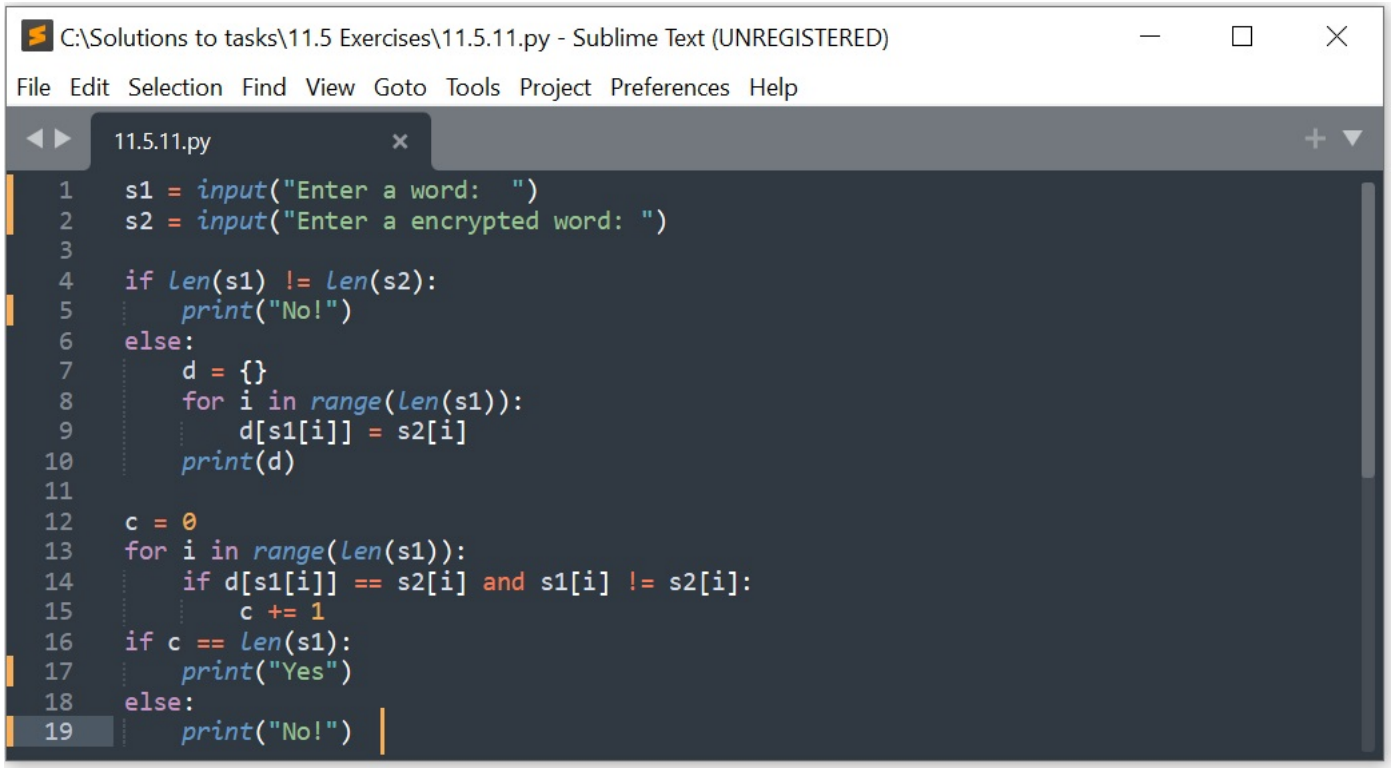
C:\Solutions to tasks\11.5 Exercises\11.5.10.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.10.py

```
from random import randint,shuffle
deck = [{'value':i, 'suit':c}
for c in ['spades', 'clubs', 'hearts', 'diamonds']
for i in range(2,15)]
shuffle(deck)
count = 0
for _ in range(100000):
    L = []
    for i in range(5):
        L.append(deck[randint(0,51)]["suit"])
        if L.count(L[0]) == 5:
            count += 1
print("The probability of being dealt a flush in a five card",(count/100000)*100, "%")
```

11. In Section 6.10 we met the substitution cipher. This cipher replaces every letter with a different letter. For instance every *a* might be replaced with an *e*, every *b* might be replaced with an *a*, etc. Write a program that asks the user to enter two strings. Then determine if the second string could be an encoded version of the first one with a substitution cipher. For instance, CXYZ is not an encoded version of BOOK because O got mapped to two separate letters. Also, CXXK is not an encoded version of BOOK, because K got mapped to itself. On the other hand, CXXZ would be an encoding of BOOK. This problem can be done with or without a dictionary.



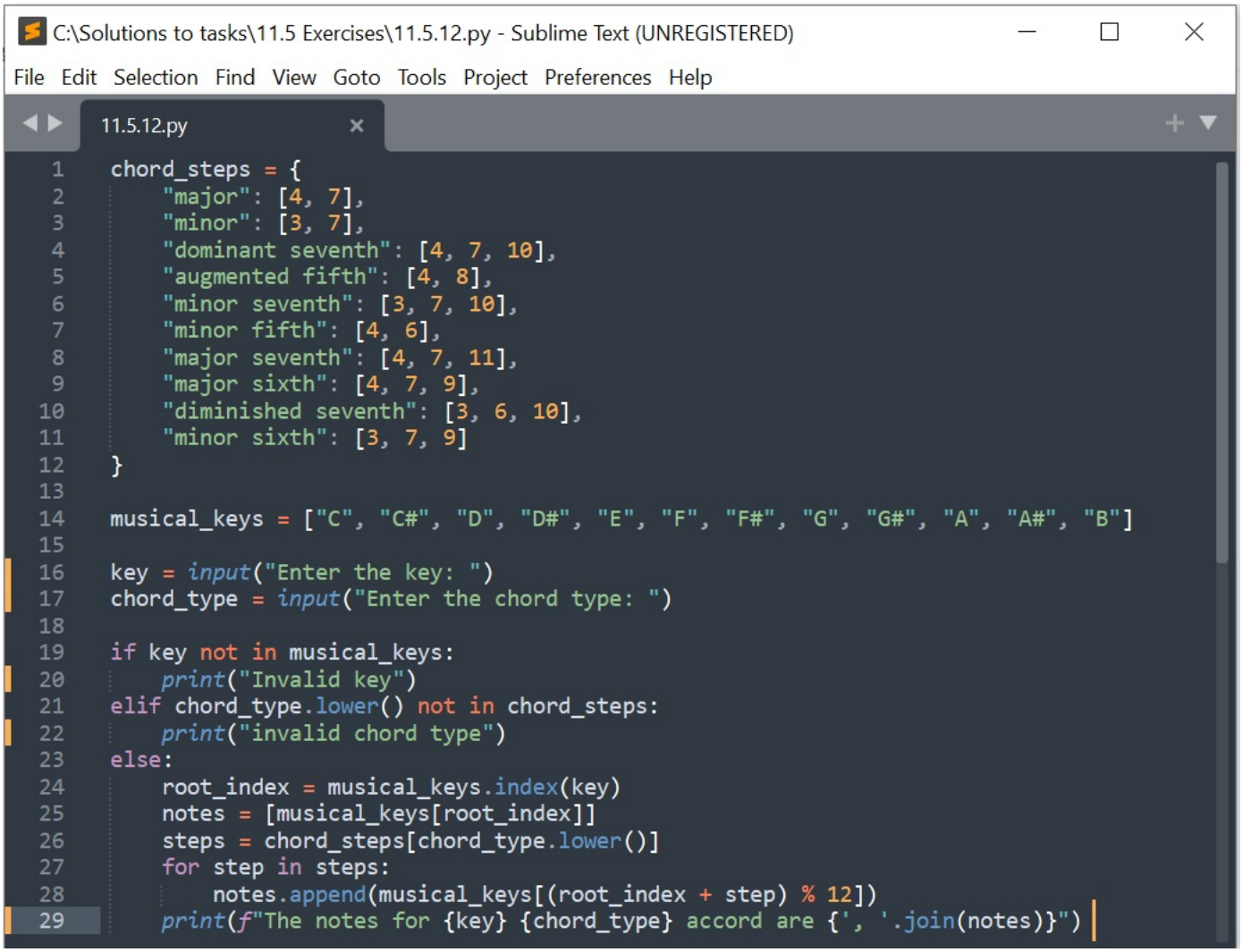
12. Below are the notes used in music:

C C# D D# E F F# G G# A A# B

The notes for the C major chord are C, E, G. A mathematical way to get this is that E is 4 steps past C and G is 7 steps past C. This works for any base. For example, the notes for D major are D, F#, A. We can represent the major chord steps as a list with two elements: [4,7]. The corresponding lists for some other chord types are shown below:

Minor	[3, 7]	Dominant seventh	[4, 7, 10]
Augmented fifth	[4, 8]	Minor seventh	[3, 7, 10]
Minor fifth	[4, 6]	Major seventh	[4, 7, 11]
Major sixth	[4, 7, 9]	Diminished seventh	[3, 6, 10]
Minor sixth	[3, 7, 9]		

Write a program that asks the user for the key and the chord type and prints out the notes of the chord. Use a dictionary whose keys are the (musical) keys and whose values are the lists of steps.



13. Suppose you are given the following list of strings:

L = ['aabaabac ', 'cabaabca ', 'aaabbcba ', 'aabacbab ', 'acababba ']

Patterns like this show up in many places, including DNA sequencing. The user has a string of their own with only some letters filled in and the rest as asterisks. An example is *a**a****. The user would like to know which of the strings in the list fit with their pattern. In the example just given, the matching strings are the first and fourth. One way to solve this problem is to create a dictionary whose keys are the indices in the user's string of the non-asterisk characters and whose values are those characters. Write a program implementing this approach (or some other approach) to find the strings that match a user-entered string.

C:\Solutions to tasks\11.5 Exercises\11.5.13.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.13.py

```
L = ['aabaabac', 'cabaabca', 'aaabbcba', 'aabacbab', 'acababba']
s = "a**a****"

d = {}
for i in range(len(s)):
    if s[i] != "*":
        d[i] = s[i]
for k in L:
    while True:
        for i in d:
            if d[i] != k[i]:
                break
        else:
            print(k)
            break
        break
```

14. Dictionaries provide a convenient way to store structured data. Here is an example dictionary:

```
d = [ { 'name' : 'Todd',  'phone' : '555-1414',  'email' : 'todd@mail.net'} ,
      { 'name' : 'Helga',  'phone' : '555-1618',  'email' : 'helga@mail.net'} ,
      { 'name' : 'Princess',  'phone' : '555-3141',  'email' : ''} ,
      { 'name' : 'LJ',  'phone' : '555-2718',  'email' : 'lj@mail.net'} ]
```

Write a program that reads through any dictionary like this and prints the following:

(a). All the users whose phone number ends in an 8

C:\Solutions to tasks\11.5 Exercises\11.5.14.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.14.a.py

```
d = [{'name':'Todd', 'phone':'555-1414', 'email':'todd@mail.net'},
     {'name':'Helga', 'phone':'555-1618', 'email':'helga@mail.net'},
     {'name':'Princess', 'phone':'555-3141', 'email':''},
     {'name':'LJ', 'phone':'555-2718', 'email':'lj@mail.net'}]

for k in d:
    if k["phone"][-1::] == str(8):
        print(k["name"],"- ", k["phone"])
```

(b).All the users that don't have an email address listed

C:\Solutions to tasks\11.5 Exercises\11.5.14.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.14.b.py

```
d = [{'name':'Todd', 'phone':'555-1414', 'email':'todd@mail.net'},
     {'name':'Helga', 'phone':'555-1618', 'email':'helga@mail.net'},
     {'name':'Princess', 'phone':'555-3141', 'email':''},
     {'name':'LJ', 'phone':'555-2718', 'email':'lj@mail.net'}]

for k in d:
    if k["email"] == "":
        print(k["name"],"- ", "no email")
```

15. The following problem is from Chapter 6. Try it again, this time using a dictionary whose keys are the names of the time zones and whose values are offsets from the Eastern time zone.

Write a program that converts a time from one time zone to another. The user enters the time in the usual American way, such as *3:48pm* or *11:26am*. The first time zone the user enters is that of the original time and the second is the desired time zone. The possible time zones are *Eastern*, *Central*, *Mountain*, or *Pacific*.

Time: 11:48pm
Starting zone: Pacific
Ending zone: Eastern
2:48am

C:\Solutions to tasks\11.5 Exercises\11.5.15.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.15.py

```
print("Eastern - 1,", "Central - 2,", "Mountain - 3,", "Pacific - 4,")
time = input("Time:  ")
start = input("Starting zone:  ")
end = input("Ending zone:  ")

d = {"1": 0,"2": 1,"3": 2,"4": 3}
s = time.split(":")
if d[start] > d[end]:
    h = (int(s[0]) + d[start]) % 12
    if (int(s[0]) + d[start]) > 12:
        print(str(h),":",s[1][:2]+"am")
    else:
        print(str(h),":",s[1][:2],"pm")
else:
    h = (int(s[0]) - d[end]) % 12
    print(str(h),":",s[1][:2])
```

16.(a). Write a program that converts Roman numerals into ordinary numbers. Here are the conversions: M=1000, D=500, C=100, L=50, X=10, V=5 I=1. Don't forget about things like IV being 4 and XL being 40.

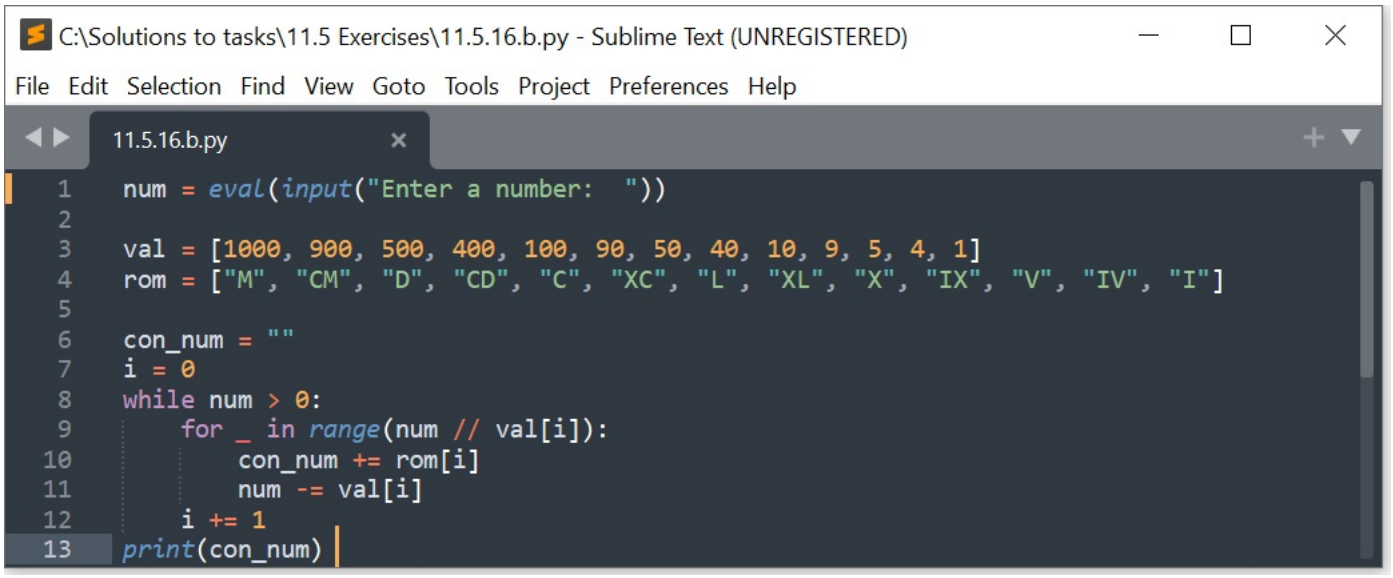
C:\Solutions to tasks\11.5 Exercises\11.5.16.a..py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

11.5.16.a..py

```
s  = input("Enter Roman numeral:  ")
n = s.upper()

d = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
val = 0

for i in range(len(n)):
    if i > 0 and d[n[i]] > d[n[i - 1]]:
        val += d[n[i]] - 2*d[n[i - 1]]
    else:
        val += d[n[i]]
print(val)
```

(b).Write a program that converts ordinary numbers into Roman numerals



Chapter 12

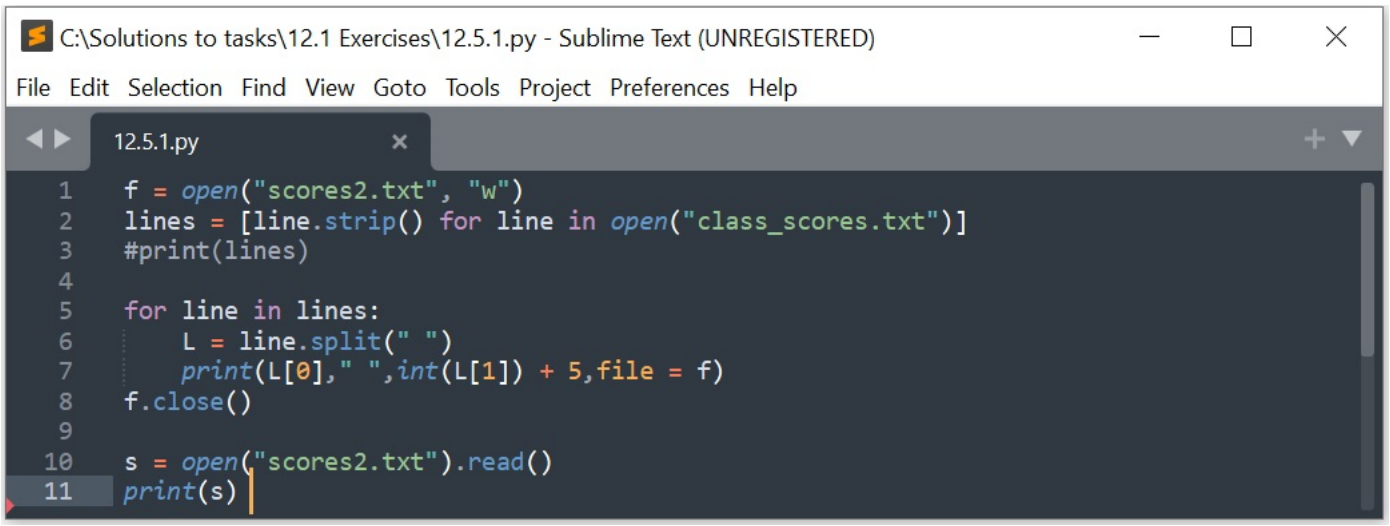
Text Files

12.1 Exercises 12.5

1. You are given a file called *class_ scores.txt*, where each line of the file contains a one word username and a test score separated by spaces, like below:

GWashington 83
JAdams 86

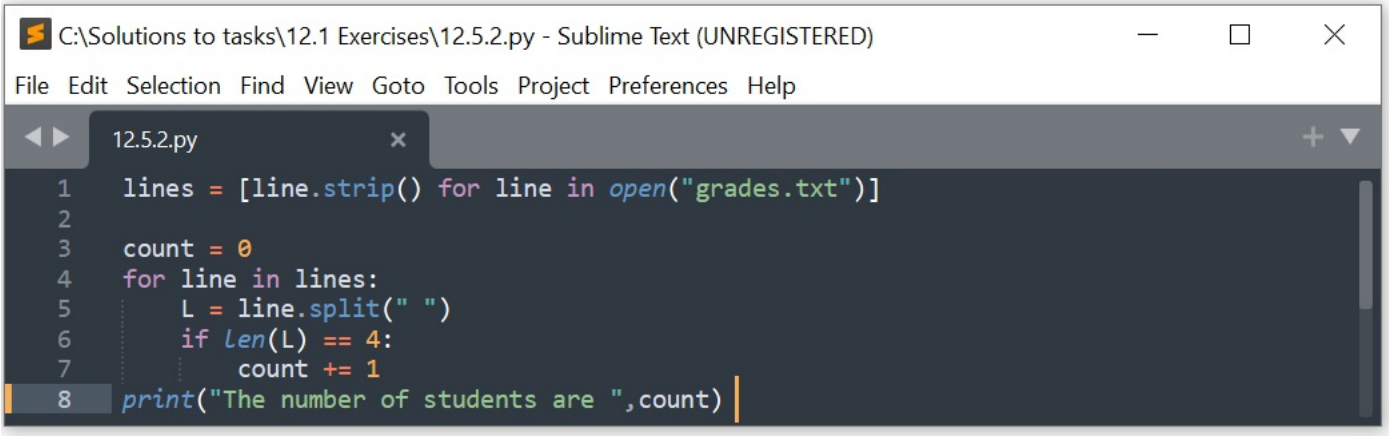
Write code that scans through the file, adds 5 points to each test score, and outputs the user names and new test scores to a new file, *scores2.txt*



2. You are given a file called *grades.txt*, where each line of the file contains a one-word student username and three test scores separated by spaces, like below

GWashington 83 77 54
JAdams 86 69 90

Write code that scans through the file and determines how many students passed all three tests.

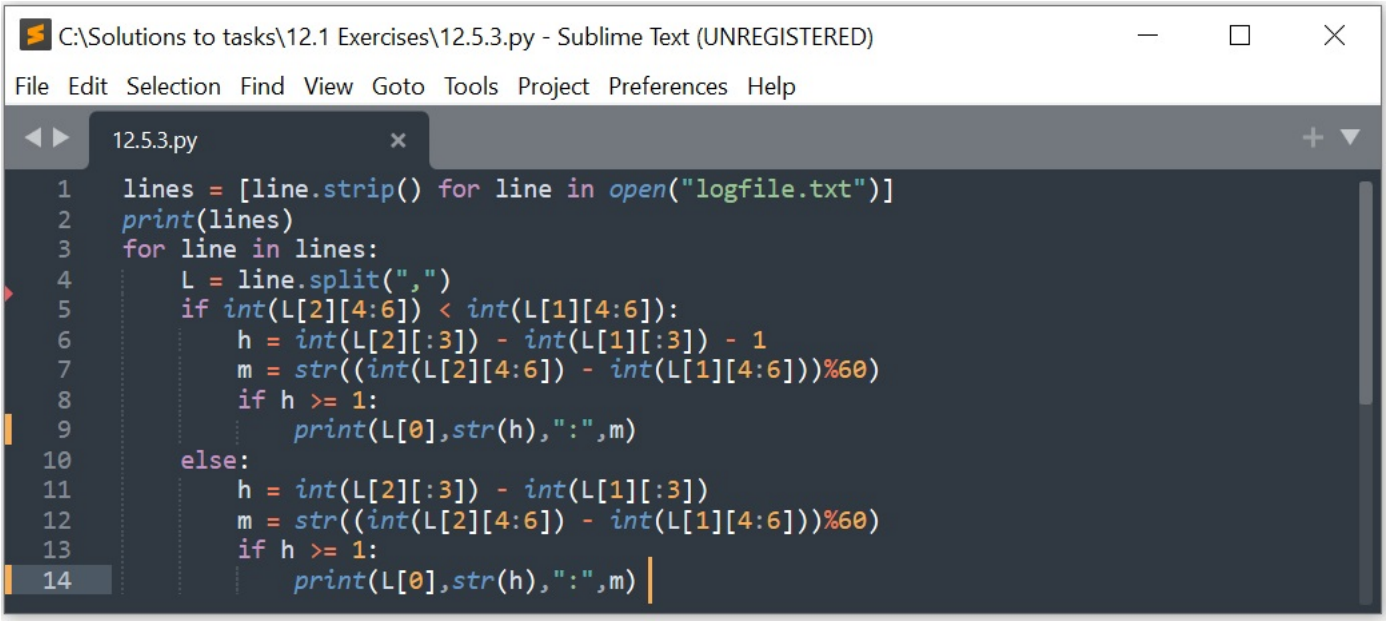


3. You are given a file called *logfile.txt* that lists log-on and log-off times for users of a system. A typical line of the file looks like this:

Van Rossum, 14:22, 14:37

Each line has three entries separated by commas: a username, a log-on time, and a log-off time. Times are given in 24-hour format. You may assume that all log-ons and log-offs occur within a single workday.

Write a program that scans through the file and prints out all users who were online for at least an hour.

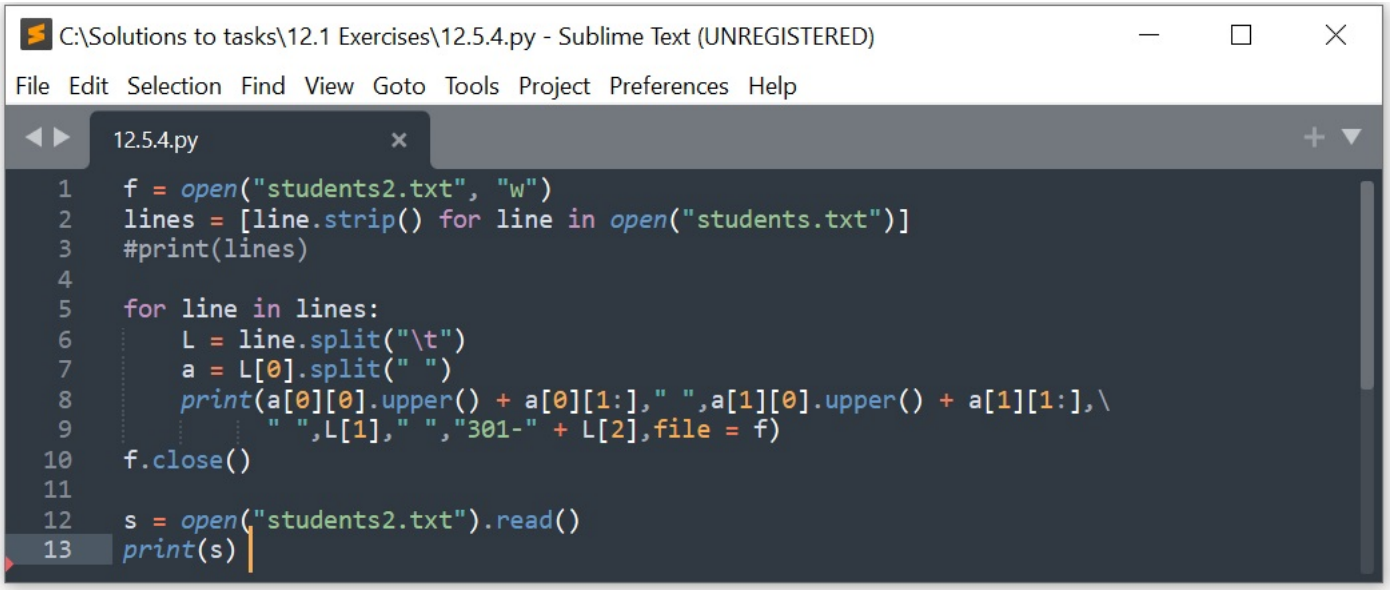


4.You are given a file called *students.txt*. A typical line in the file looks like:

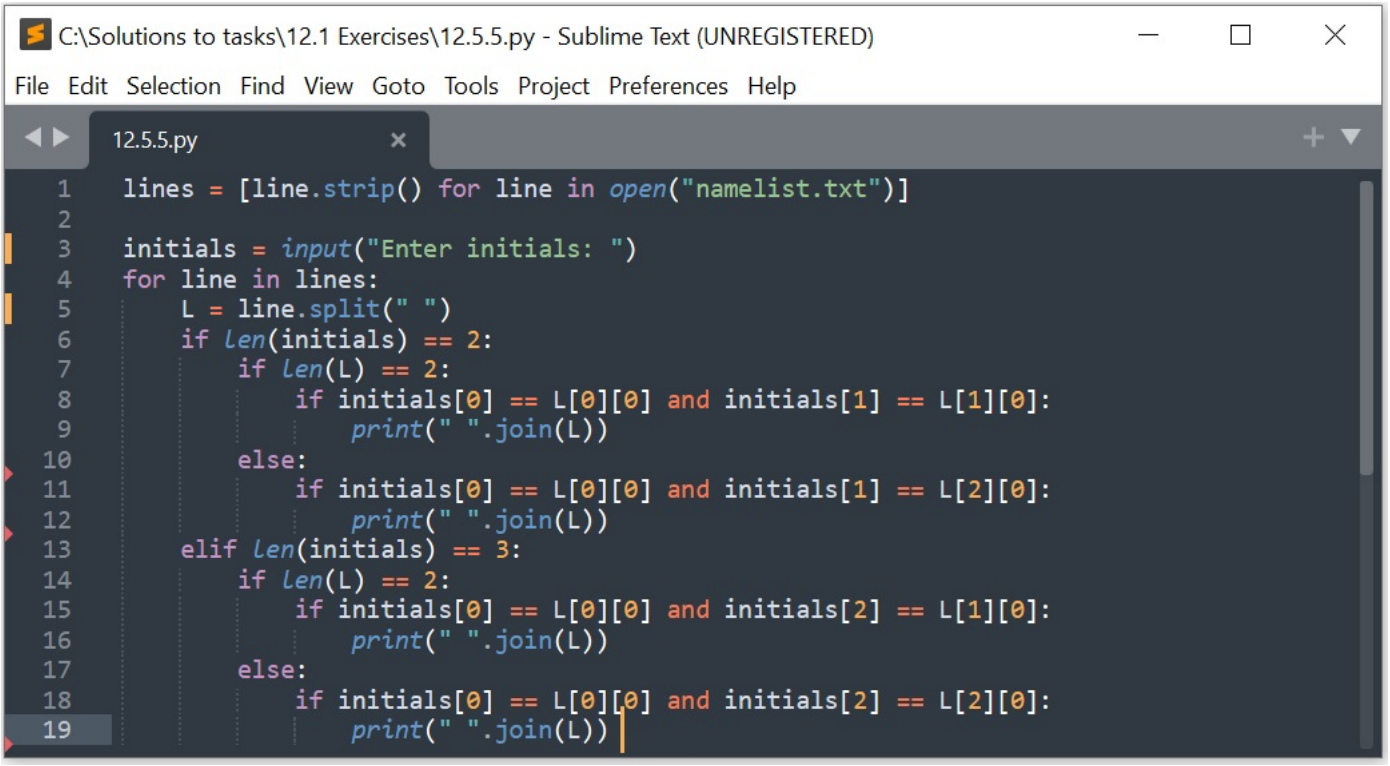
walter melon melon@email.msmary.edu 555-3141

There is a name, an email address, and a phone number, each separated by tabs. Write a program that reads through the file line-by-line, and for each line, capitalizes the first letter of the first and last name and adds the area code 301 to the phone number. Your program should write this to a new file called *students2.txt*. Here is what the first line of the new file should look like:

Walter Melon melon@email.msmary.edu 301-555-3141



5. You are given a file *namelist.txt* that contains a bunch of names. Some of the names are a first name and a last name separated by spaces, like *George Washington*, while others have a middle name, like *John Quincy Adams*. There are no names consisting of just one word or more than three words. Write a program that asks the user to enter initials, like *GW* or *JQA*, and prints all the names that match those initials. Note that initials like JA should match both *John Adams* and *John Quincy Adams*.



6. You are given a file *namelist.txt* that contains a bunch of names. Print out all the names in the list in which the vowels *a*, *e*, *i*, *o*, and *u* appear in order (with repeats possible). The first vowel in the name must be *a* and after the first *u*, it is okay for there to be other vowels. An example is *Ace Elvin Coulson*.

C:\Solutions to tasks\12.1 Exercises\12.5.6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.6.py

```
lines = [line.strip() for line in open("namelist.txt")]

L = ["a","e","i","o",'u']
for line in lines:
    M = []
    for i in range(len(line)):
        k = line[i].lower()
        if k in L:
            if k not in M:
                M.append(k)
    if M == L:
        print(line)
```

7. You are given a file called *baseball.txt*. A typical line of the file starts like below.

Ichiro Suzuki SEA 162 680 74 . . . [more stats]

Each entry is separated by a tab, \t. The first entry is the player's name and the second is their team. Following that are 16 statistics. Home runs are the seventh stat and stolen bases are the eleventh. Print out all the players who have at least 20 home runs and at least 20 stolen bases.

C:\Solutions to tasks\12.1 Exercises\12.5.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.7.py

```
l = [line.strip() for line in open('baseball.txt')]

for i in range(len(l) - 1):
    L = l[i].split("\t")
    if int(L[6]) > 20 and int(L[10]) > 20:
        print(L[0],L[6],L[10])
```

8. For this problem, use the file of NCAA basketball scores as described in Section 12.3.

(a). Find the average of the points scored over all the games in the file.

C:\Solutions to tasks\12.1 Exercises\12.5.8.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.8.a.py

```
lines = [line.strip() for line in open('scores.txt')]
M = []
for line in lines:
    L = line.split(' ')
    M.append(int(L[2]))
    M.append(int(L[4]))
print(sum(M)/len(M))
```

(b). Pick your favorite team and scan through the file to determine how many games they won and how many games they lost.

C:\Solutions to tasks\12.1 Exercises\12.5.8.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.8.b.py

```
lines = [line.strip() for line in open('scores.txt')]
team = input("Enter the team name from scores.txt -  ")

loss = 0
win = 0
for line in lines:
    L = line.split(' ')
    if L[1] == team:
        if int(L[2]) > int(L[4]):
            win += 1
        else:
            loss += 1

    if L[3] == team:
        if int(L[4]) > int(L[2]):
            win += 1
        else:
            loss += 1
print("Wins - ", win, "Losses - ", loss)
```

(c). Find the team(s) that lost by 30 or more points the most times

(d). Find all the teams that averaged at least 70 points a game.

C:\Solutions to tasks\12.1 Exercises\12.5.8.d.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.8.d.py

```
lines = [line.strip() for line in open('scores.txt')]

d = {}
M = []
for line in lines:
    L = line.split(' ')
    if L[1] not in M :
        M.append(L[1])
    if  L[3] not in M:
        M.append(L[3])
for i in M:
    d[i] = []
    for line in lines:
        L = line.split(' ')
        if L[1] == i:
            d[i].append(int(L[2]))
        if L[3] == i:
            d[i].append(int(L[4]))
C = []
for i in d:
    average = round(sum(d[i])/len(d[i]), 2)
    if average >= 70:
        n = i,average
        C.append(n)
print(C)
```

(e). Find all the teams that had winning records but were collectively outscored by their opponents. A team is collectively outscored by their opponents if the total number of points the team scored over all their games is less than the total number of points their opponents scored in their games against the

team.

C:\Solutions to tasks\12.1 Exercises\12.5.8.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.8.e.py

```
lines = [line.strip() for line in open('scores.txt')]
d = {}
M = []
for line in lines:
    L = line.split(' ')
    if L[1] not in M :
        M.append(L[1])
    if  L[3] not in M:
        M.append(L[3])
C = []
for i in M:
    w = 0
    op = 0
    for line in lines:
        L = line.split(' ')
        if L[1] == i:
            w += int(L[2])
            op += int(L[4])
        if L[3] == i:
            w += int(L[4])
            op += int(L[2])
    if w < op:
        c = i,w,op
        C.append(c)
print(C)
```

9. Benford's law states that in real data where the values are spread across several orders of magnitude, about 30% of the values will start with the number 1, whereas only about 4.6% of the values will start with the number 9. This is contrary to what we might expect, namely that values starting with 1 and 9 would be equally likely. Using the file *expenses.txt* which consists of a number of costs from an expense account, determine what percentage start with each of the digits 1 through 9. This technique is used by accountants to detect fraud.

C:\Solutions to tasks\12.1 Exercises\12.5.9.py - Sublime Text (UNREGISTERED)

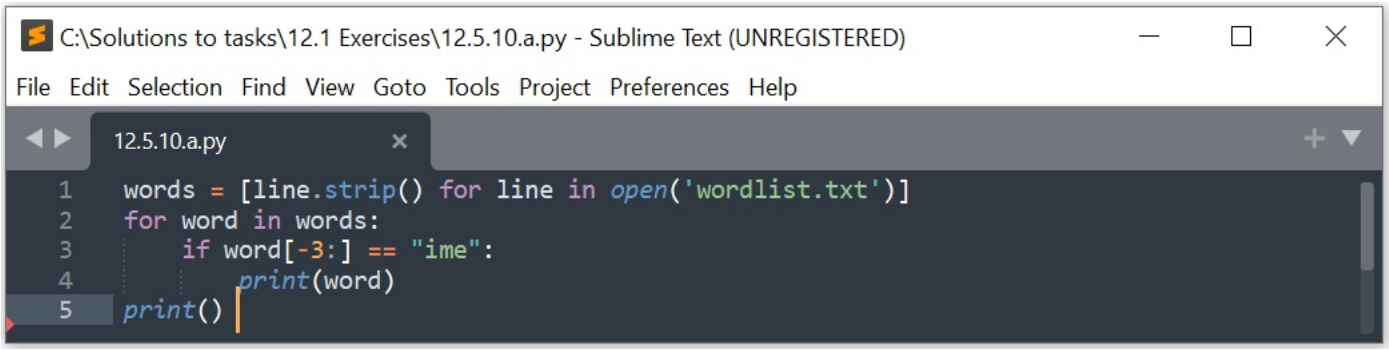
File Edit Selection Find View Goto Tools Project Preferences Help

12.5.9.py

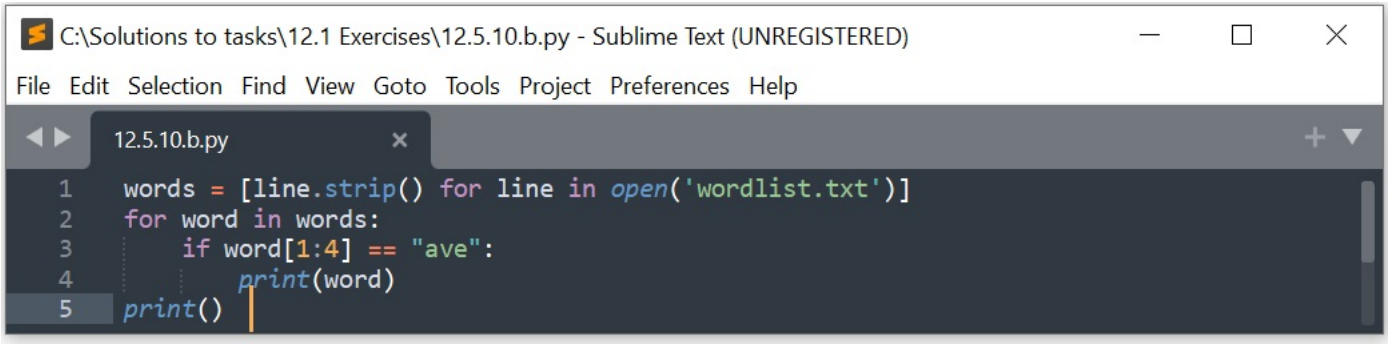
```
lines = [line.strip() for line in open('expenses.txt')]

for i in range(1,10):
    count = 0
    for line in lines:
        if int(line[0]) == i:
            count +=1
    print("For ",i," - ",(count/len(lines))*100,"%")
    print()
```

10. *Wordplay* – Use the file *wordlist.txt* for this problem. Find the following:

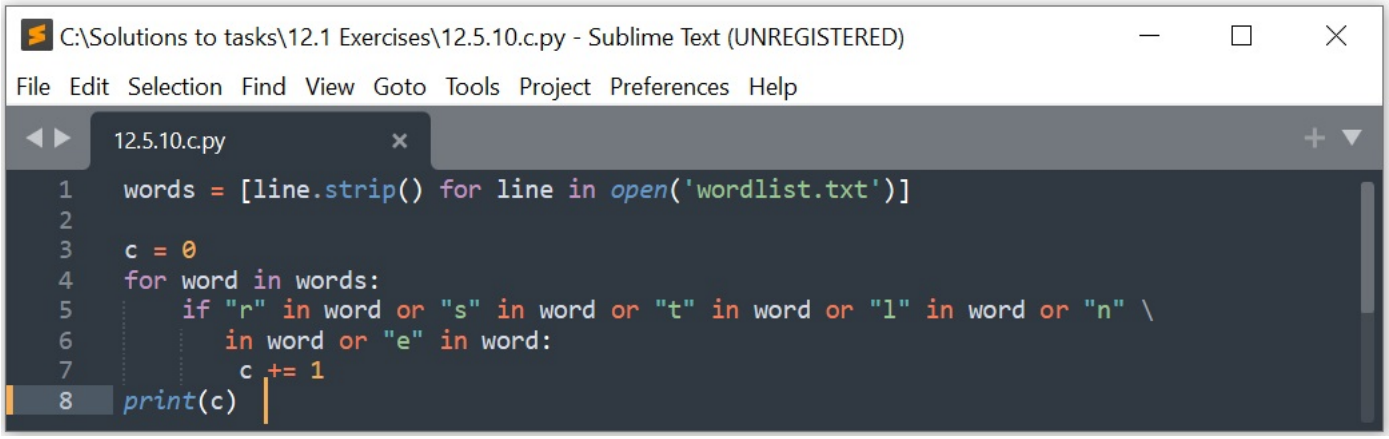
(a) All words ending in *ime*.



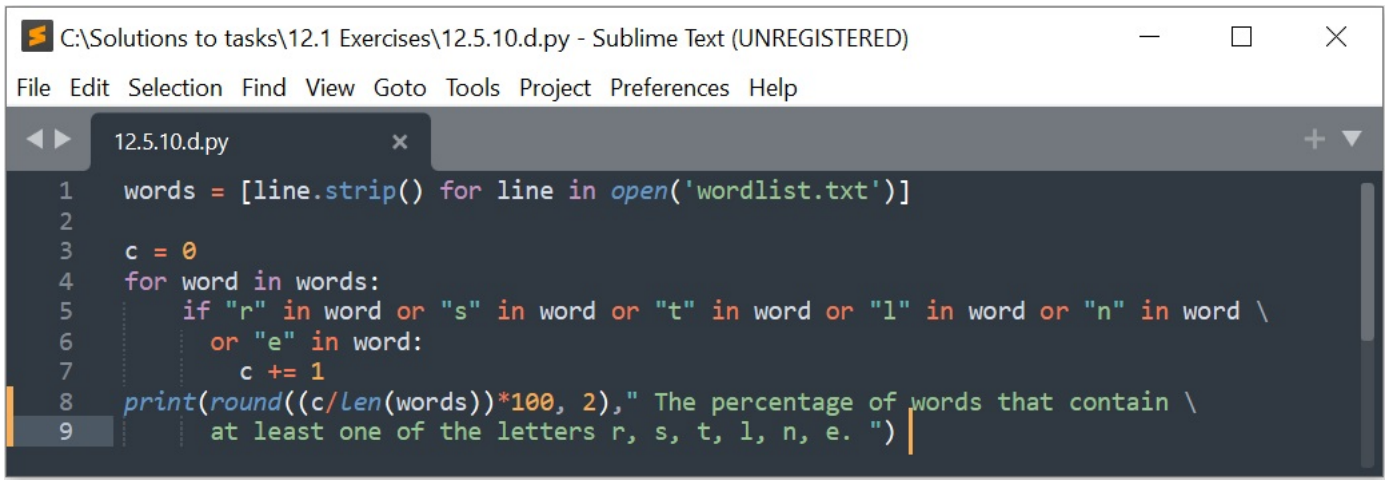
(b) All words whose second, third, and fourth letters are *ave*.



(c) How many words contain at least one of the letters *r, s, t, l, n, e*.



(d) The percentage of words that contain at least one of the letters *r, s, t, l, n, e*.



(e) All words with no vowels.

C:\Solutions to tasks\12.1 Exercises\12.5.10.e.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.e.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2
3  for word in words:
4      if "a" not in word and "e" not in word and "i" not in word and "o" not in \
5      word and "u" not in word:
6          print(word)
```

(f) All words that contain every vowel.

C:\Solutions to tasks\12.1 Exercises\12.5.10.f.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.f.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2
3  for word in words:
4      if "a" in word and "e" in word and "i" in word and "o" in word and "u" in word:
5          print(word)
```

(g) Whether there are more ten-letter words or seven-letter words.

C:\Solutions to tasks\12.1 Exercises\12.5.10.g.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.g.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  ten = 0
3  seven = 0
4  for word in words:
5      if len(word) == 10:
6          ten += 1
7      if len(word) == 7:
8          seven += 1
9  if ten > seven:
10     print("There are more ten-letter words - ", ten, ">", seven)
11 else:
12     print("There are more seven-letter words - ", ten, "<", seven)
```

(h) The longest word in the list.

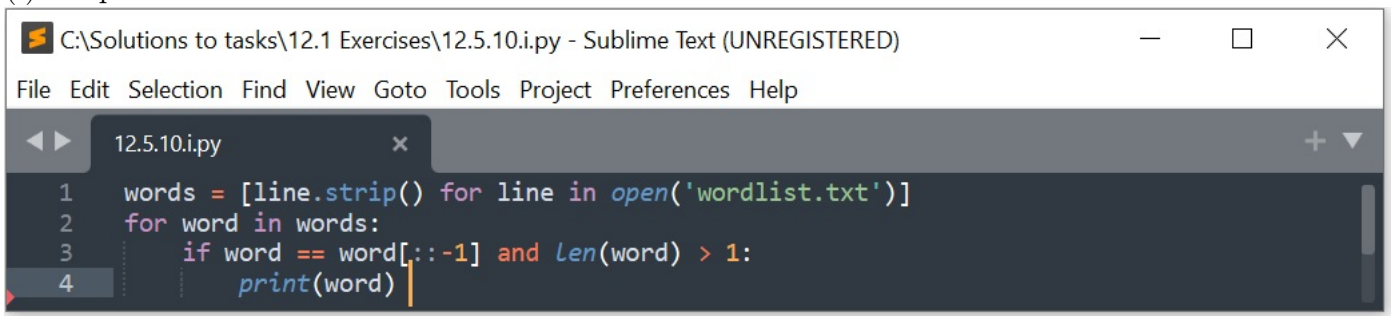
C:\Solutions to tasks\12.1 Exercises\12.5.10.h.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

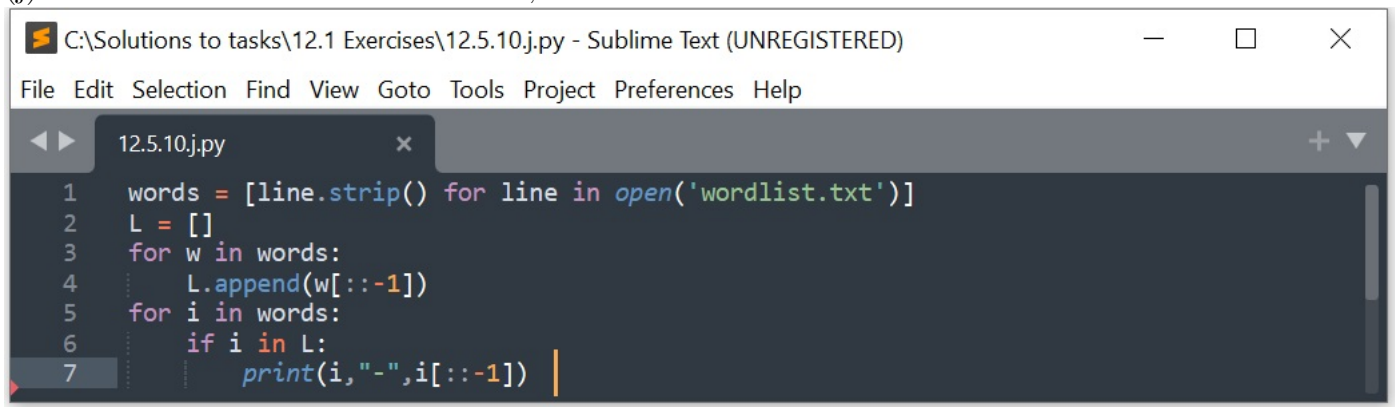
12.5.10.h.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  l = 0
3  for word in words:
4      if len(word) > l:
5          l =len(word)
6          l_word = word
7  print(l_word)
```

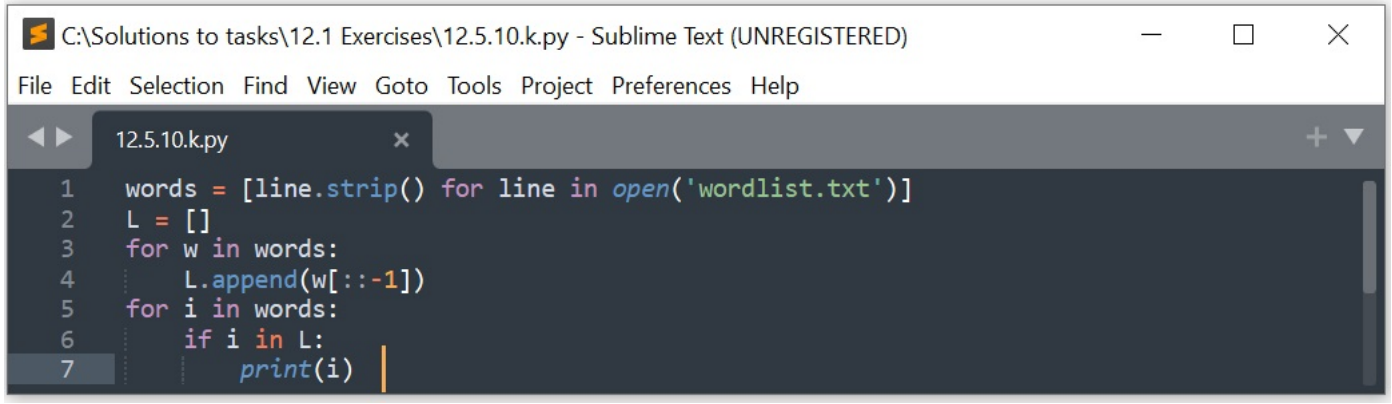
(i) All palindromes.



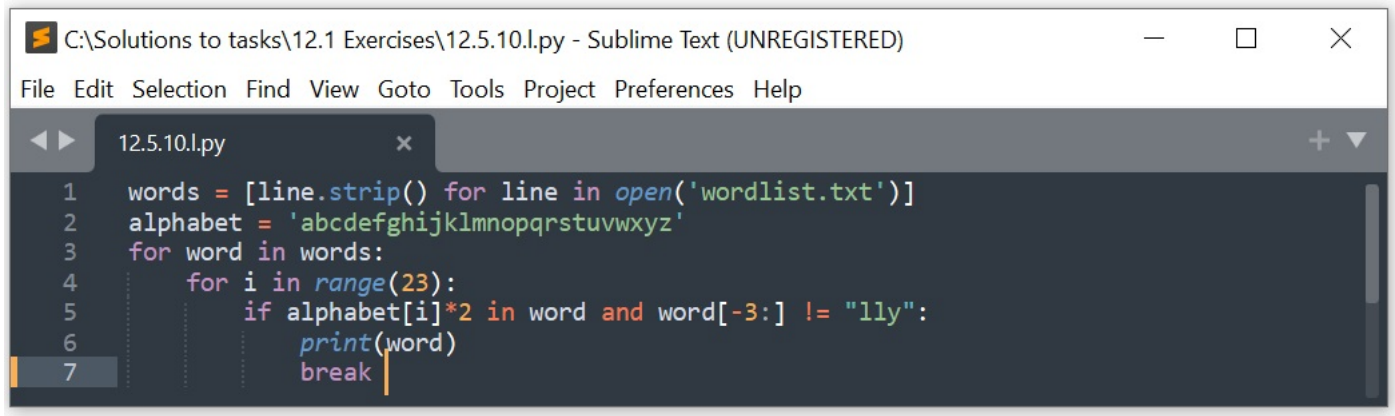
(j) All words that are words in reverse, like *rat* and *tar*.



(k) Same as above, but only print one word out of each pair.



(l) All words that contain double letters next each other like *aardvark* or *book*, excluding words that end in *lly*.



(m) All words that contain a *q* q that isn't followed by a *u*.

C:\Solutions to tasks\12.1 Exercises\12.5.10.m.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.m.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  for word in words:
3      if "q" in word and "qu" not in word:
4          print(word)
```

(n) All words that contain *zu* anywhere in the word.

C:\Solutions to tasks\12.1 Exercises\12.5.10.n.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.n.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  for word in words:
3      if "zu" in word:
4          print(word)
```

All words that contain *ab* in multiple places, like *habitable*.

C:\Solutions to tasks\12.1 Exercises\12.5.10.o.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.o.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  for word in words:
3      count = 0
4      for i in range(len(word)):
5          if word[i:i+2] == "ab":
6              count +=1
7      if count >= 2:
8          print(word)
9  print()
10 # Another solution
11 for word in words:
12     if word.count("ab") >= 2:
13         print(word)
```

(p) All words with four or more vowels in a row.

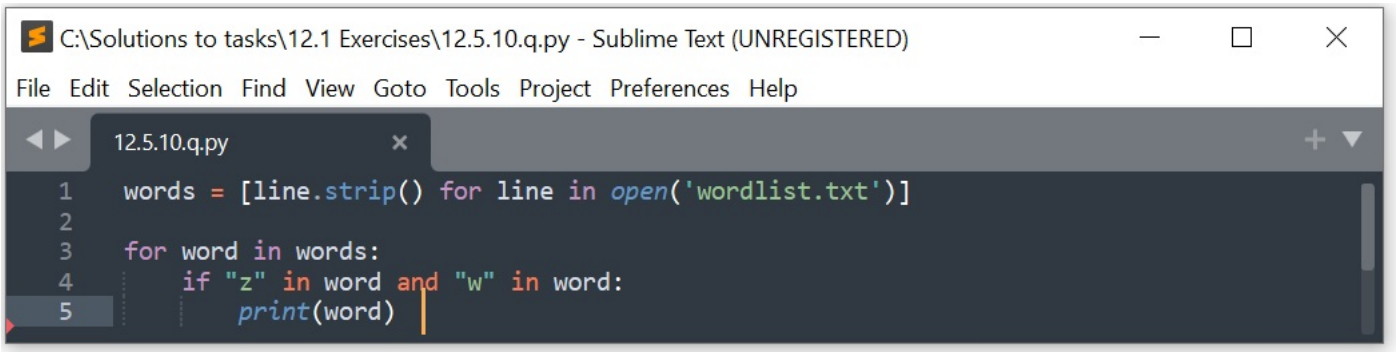
C:\Solutions to tasks\12.1 Exercises\12.5.10.p.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

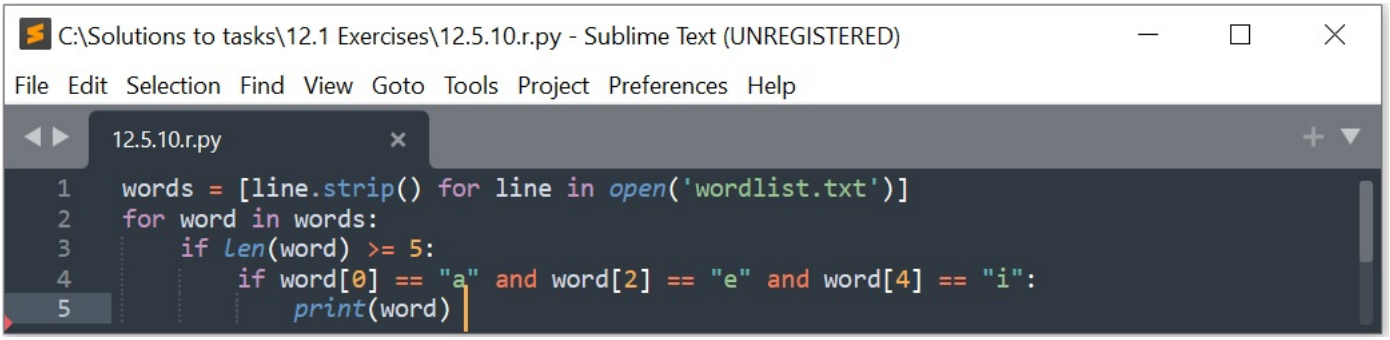
12.5.10.p.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  for word in words:
3      vowels = "aeiou"
4      count = 0
5      max_count = 0
6      for i in range(len(word)):
7          if word[i] in vowels:
8              count += 1
9              max_count = max(max_count, count)
10         else:
11             count = 0
12     if max_count >= 4:
13         print(word)
```

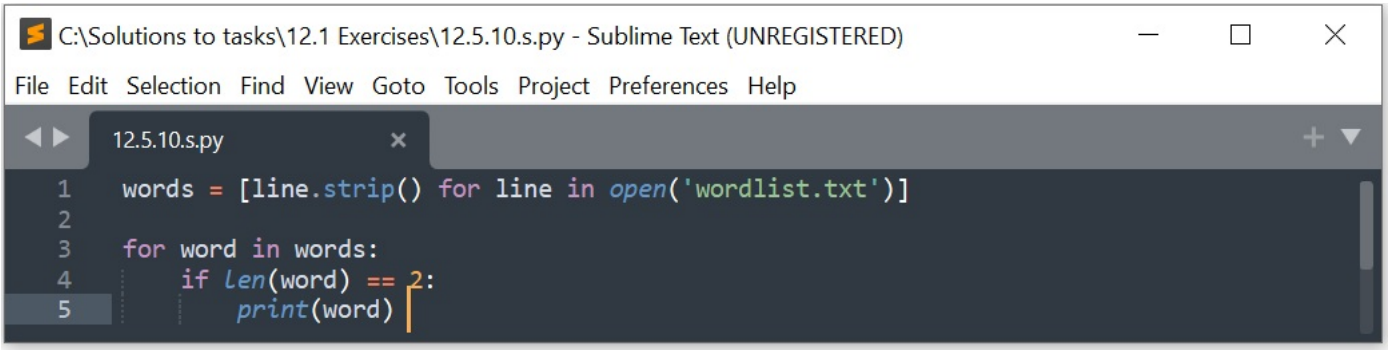
(q) All words that contain both a z or w.



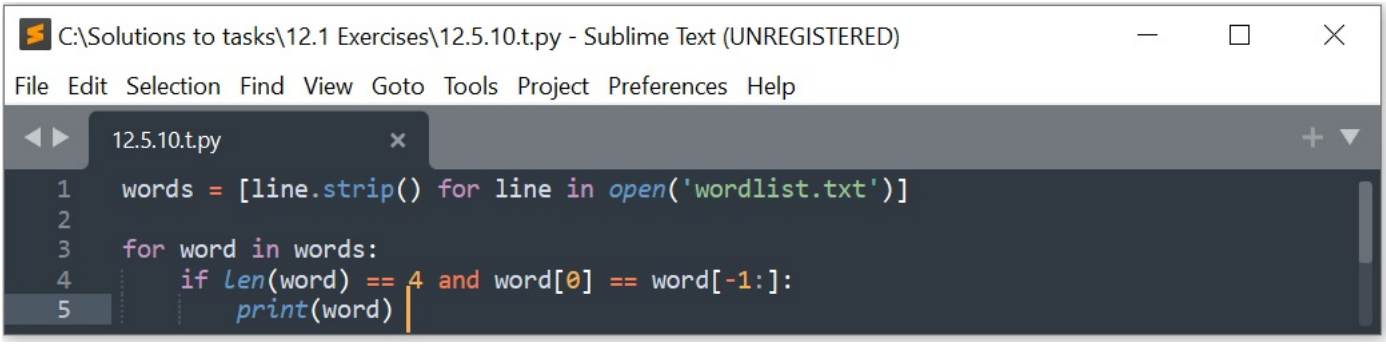
(r) All words whose first letter is a, third letter is e and fifth letter is i.



(s) All two-letter words.



(t) All four-letter words that start and end with the same letter.



(u) All words that contain at least nine vowels.

C:\Solutions to tasks\12.1 Exercises\12.5.10.u.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.u.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  v = "aeiou"
3
4  for word in words:
5      count = 0
6      for i in range(len(word)):
7          if word[i] in v:
8              count += 1
9      if count >= 9:
10         print(word)
```

(v) All words that contain each of the letters *a*, *b*, *c*, *d*, *e*, and *f* in any order. There may be other letters in the word. Two examples are *backfield* and *feedback*.

C:\Solutions to tasks\12.1 Exercises\12.5.10.v.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.v.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2
3  for word in words:
4      if "a" in word and "b" in word and "c" in word and "d" in word and "f" in word:
5          print(word)
```

(w) All words whose first four and last four letters are the same

C:\Solutions to tasks\12.1 Exercises\12.5.10.w.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.w.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2
3  for word in words:
4      if len(word) >= 8:
5          start = list(word[:4])
6          end = list(word[-4:])
7          start.sort()
8          end.sort()
9          if start == end:
10             print(word)
```

(x) All words of the form *abcd*dcba*, where * is arbitrarily long sequence of letters.

C:\Solutions to tasks\12.1 Exercises\12.5.10.x.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.x.py

```
1  words = [line.strip() for line in open('wordlist.txt')]
2  for word in words:
3      if len(word) >= 8:
4          w_rev = word[-4::]
5          if word[:4] == w_rev[::-1]:
6              print(word)
```

(y) All groups of 5 words, like *pat, pet, pit, pot, put*, where each word is 3 letters, all words share the same first and last letters, and the middle letter runs through all 5 vowels.

C:\Solutions to tasks\12.1 Exercises\12.5.10.y.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.y.py

```
1
2   words = [line.strip() for line in open('wordlist.txt')]
3
4   vowels = "aeiou"
5   for let_first in "bcdfghjklmnpqrstvwxyz":
6       for let_last in "bcdfghjklmnpqrstvwxyz":
7           L = []
8           for vovel in vowels:
9               if (let_first + vovel + let_last)   in words:
10                  L.append(let_first + vovel + let_last)
11              else:
12                  break
13          if len(L) == 5:
14              print(" ".join(L))
```

(z) The word that has the most i's.

C:\Solutions to tasks\12.1 Exercises\12.5.10.z.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.10.z.py

```
1   words = [line.strip() for line in open('wordlist.txt')]
2   largest = ""
3   x = 0
4   for word in words:
5       c = 0
6       for i in range(len(word)):
7           if word[i] == "i":
8               c += 1
9       if c>x:
10          x = c
11          largest = word
12  print(largest)
```

11.Write a program to help with word games. The user enters a word and the program uses the wordlist to determine if the user's word is a real word or not.

C:\Solutions to tasks\12.1 Exercises\12.5.11.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.11.py

```
1   word = input("Enter a word:    ")
2   words = [line.strip() for line in open('wordlist.txt')]
3   if word in words:
4       print("Yes")
5   else:
6       print("No")
```

12. Suppose we write all the words in the wordlist backwards and then arrange these backwards words

alphabetically. Write a program that prints the last word in this modified wordlist.

C:\Solutions to tasks\12.1 Exercises\12.5.12.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.12.py

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
L = []
words = [line.strip() for line in open('wordlist.txt')]
for word in words:
    word_back = word[::-1]
    L.append(word_back)
M = []
for i in range(26):
    for w in L:
        if w[0] == alphabet[i]:
            M.append(w)
print(M[-1::])
```

13. Print out all combinations of the string 'Python' plus a three letter English word. Capitalize the first letter of the three letter word. Example combinations are 'PythonCat', 'PythonDog' and 'PythonTag'. These are valid combinations because *cat, dog* and *tag* are English words. On the other hand, 'PythonQqz' would not be a valid combination because *qqz* is not an English word. Use a wordlist to determine which three letter combinations are words.

C:\Solutions to tasks\12.1 Exercises\12.5.13.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.13.py

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
L = []
for i in alphabet:
    for j in alphabet:
        for z in alphabet:
            L.append(i + j + z)

words = [line.strip() for line in open('wordlist.txt')]
M = []
for i in L:
    if i in words:
        M.append("Python" + i[0].upper() + i[1:3])
print(M)
```

14. Write a simple spell-checking program. The user should enter a string and the program should print out a list of all the words it thinks are misspelled. These would be all the words it cannot find in a wordlist.

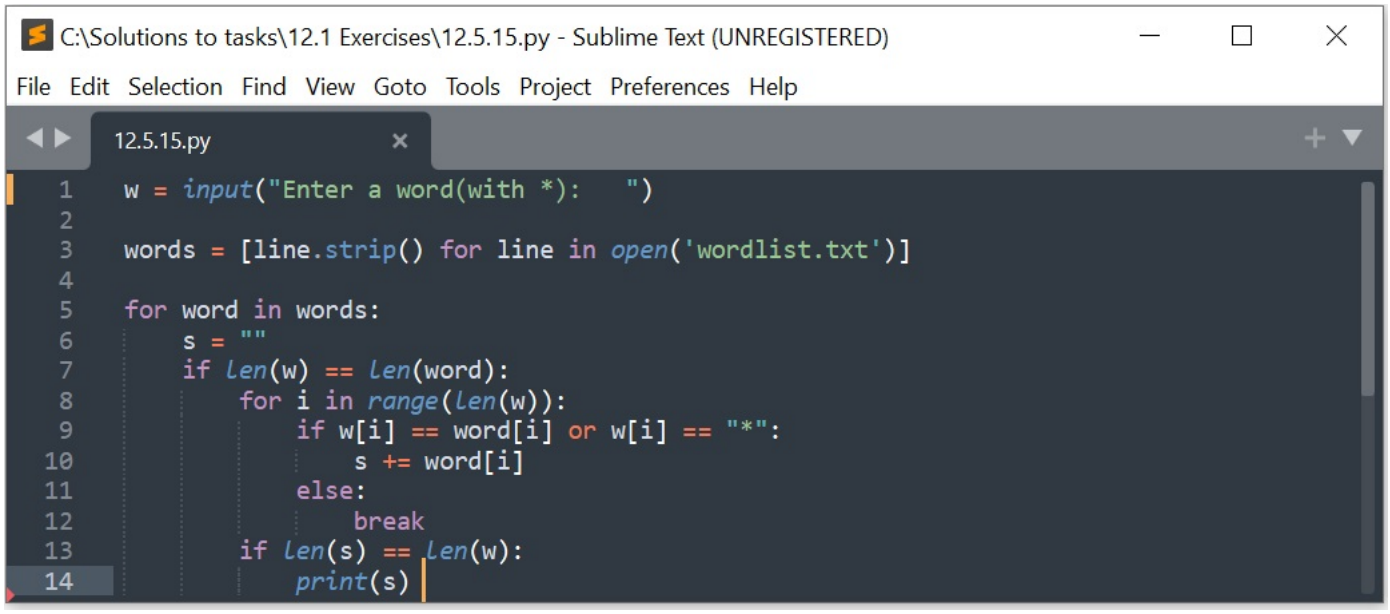
C:\Solutions to tasks\12.1 Exercises\12.5.14.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

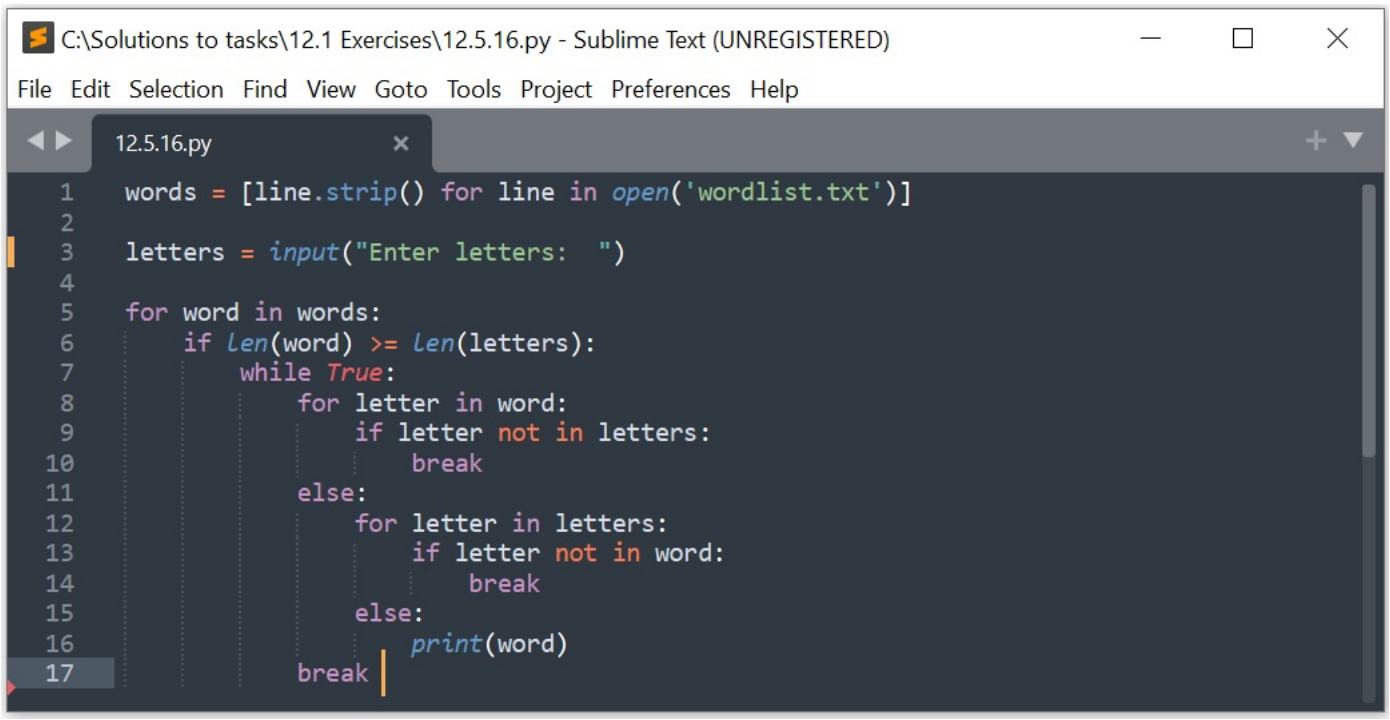
12.5.14.py

```
words = [line.strip() for line in open('wordlist.txt')]
s = input("Enter a string ")
s_l = s.lower()
L = s_l.split()
for i in L:
    if i not in words:
        print("The word ",i," is misspelled")
```

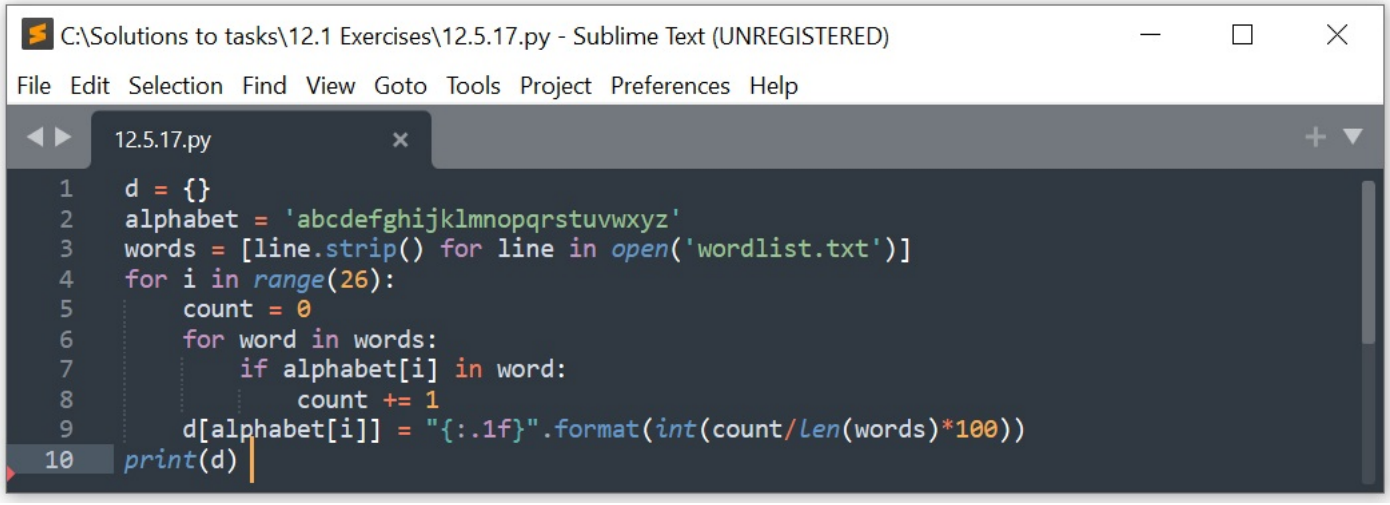
15. Crossword cheater: When working on a crossword puzzle, often you will have a word where you know several of the letters, but not all of them. You can write a computer program to help you. For the program, the user should be able to input a word with the letters they know filled in and asterisks for those they don't know. The program should print out a list of all words that fit that description. For example, the input *th***ly* y should return all the words that could work, namely *thickly* and *thirdly*



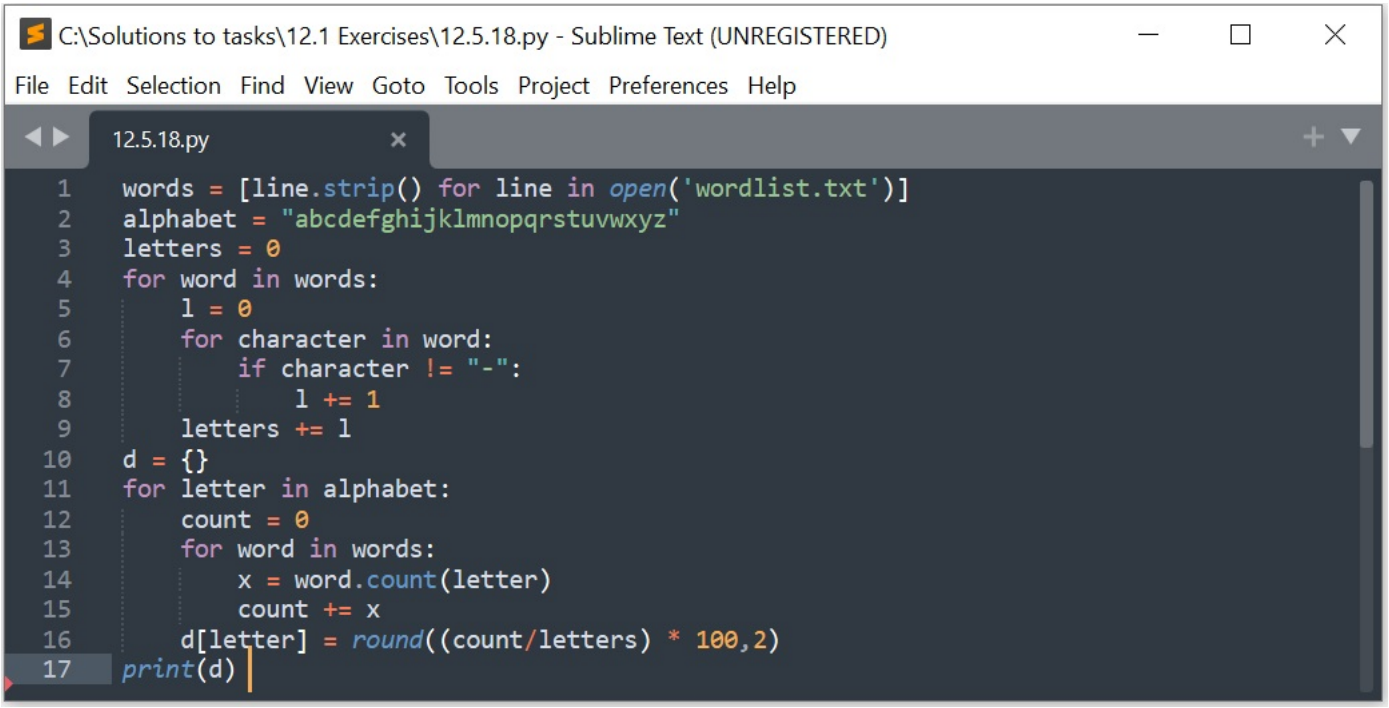
16. Ask the user to enter several letters. Then find all the words that can be made with those letters, repeats allowed.



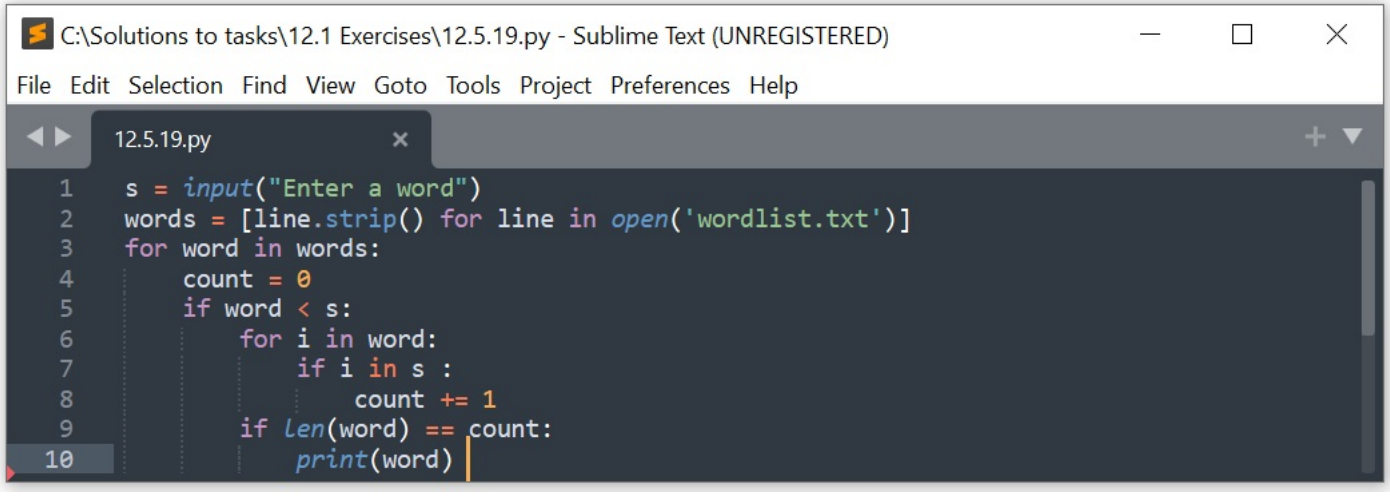
17. Using the wordlist, produce a dictionary whose keys are the letters a through z and whose values are the percentage of words that use that letter.



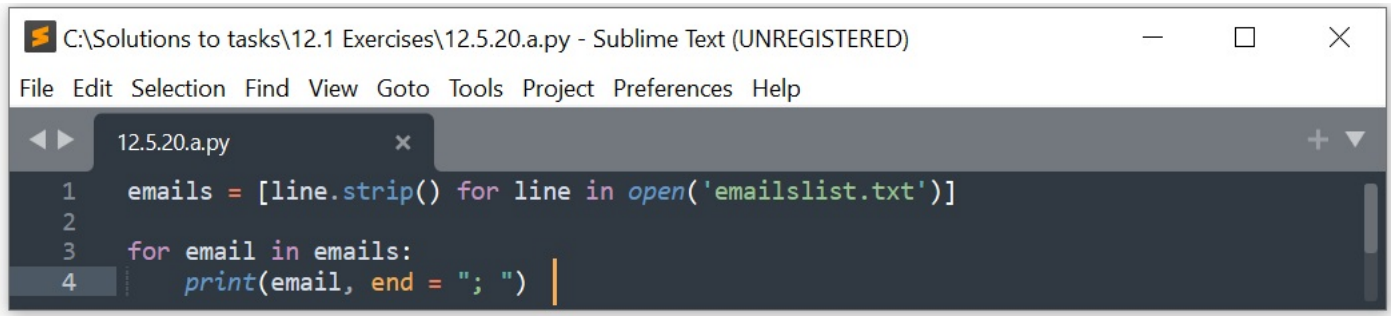
18. Using the wordlist, produce a dictionary whose keys are the letters a through z and whose values are the percentage of total letters in the wordlist that are that letter.



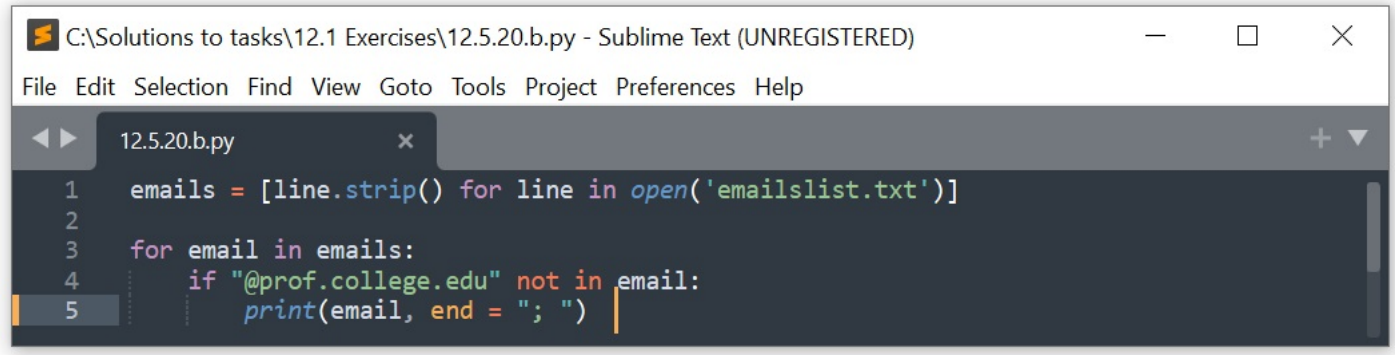
19. Write a program that asks the user for a word and finds all the smaller words that can be made from the letters of that word. The number of occurrences of a letter in a smaller word can't exceed the number of occurrences of the letter in the user's word.



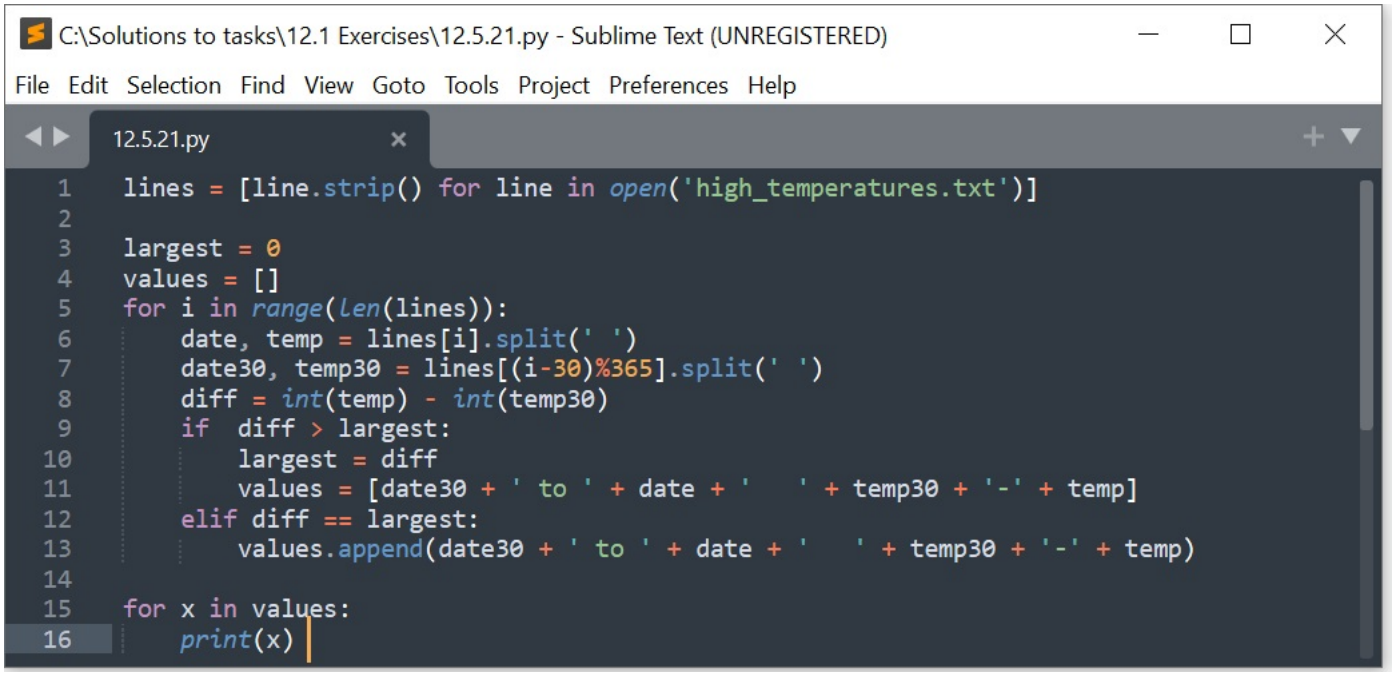
20.(a). Write a program that reads a file consisting of email addresses, each on its own line. Your program should print out a string consisting of those email addresses separated by semicolons.



(b).Write the same program as above, but the new string should contain only those email addresses that do not end in *@prof.college.edu*

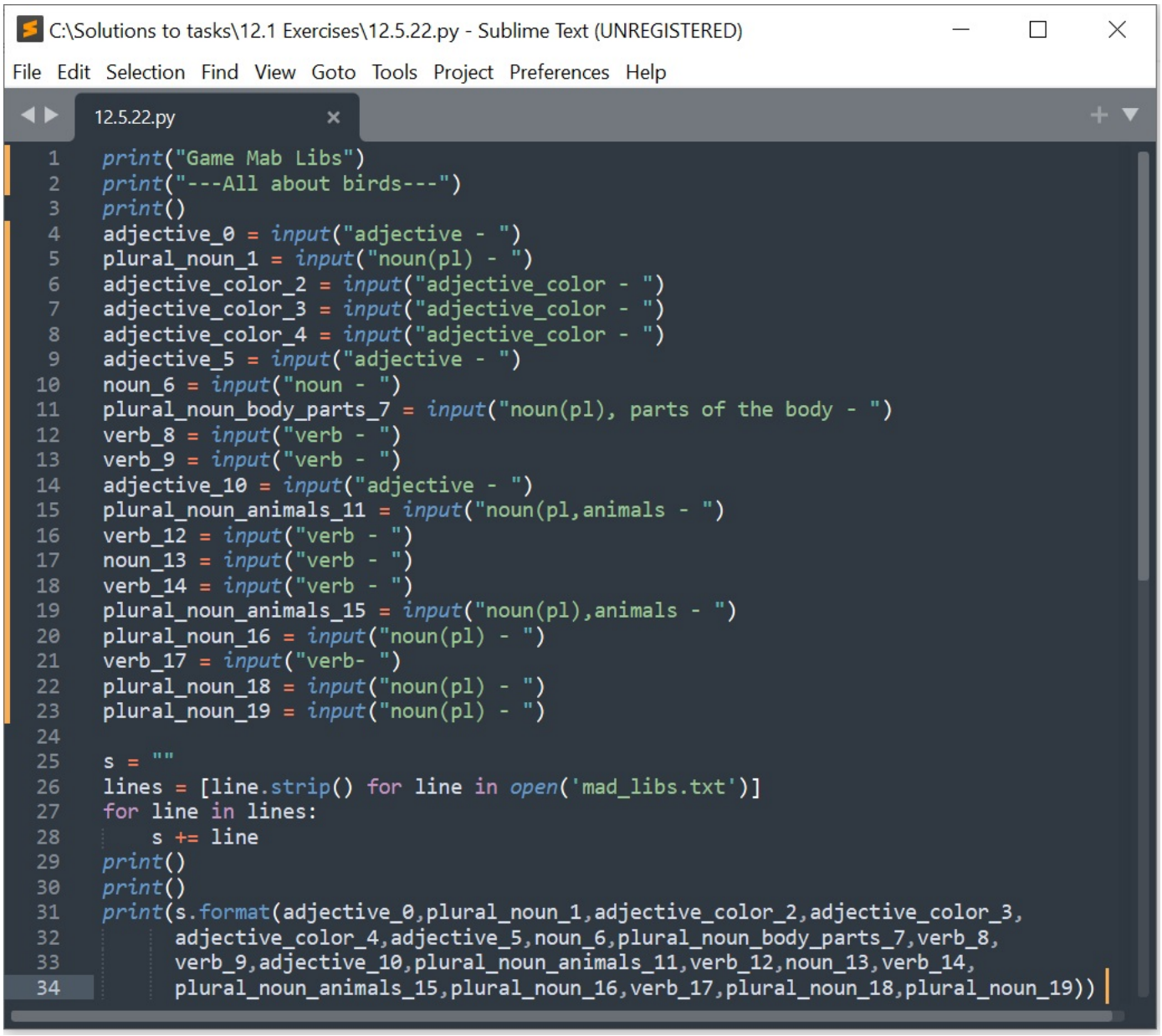


21. The file *high_ temperatures.txt* The file *high_ temperatures.txt* contains the average high temperatures for each day of the year in a certain city. Each line of the file consists of the date, written in the month/day format, followed by a space and the average high temperature for that date. Find the 30-day period over which there is the biggest increase in the average high temperature.



22. In Chapter 6 there was an exercise about the game *Mad Libs*. It asked you to make up a story and leave out some words of the story. Your program should ask the user to enter some words and tell them what types of words to enter. Then print the full story along with the inserted words. Rewrite your program from that exercise to read the story from a file. Reading the story from a file allows people

who do not know how to program to use their own stories with the program without having to change the code.



23. An acronym is an abbreviation that uses the first letter of each word in a phrase. We see them everywhere. For instance, NCAA for National Collegiate Athletic Association or NBC for National Broadcasting Company. Write a program where the user enters an acronym and the program randomly selects words from a wordlist such that the words would fit the acronym. Below is some typical output generated when I ran the program:

Enter acronym: ABC
['addressed', 'better', 'common']

Enter acronym: BRIAN
['bank', 'regarding', 'intending', 'army', 'naive']

C:\Solutions to tasks\12.1 Exercises\12.5.23.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.23.py

```
from random import choice
acronym = input("Enter an acronym:  ")
words = [line.strip() for line in open('wordlist.txt')]
A = []
for i in range(len(acronym)):
    L = []
    for word in words:
        if word[0] == acronym[i].lower():
            L.append(word)
    A.append(choice(L))
print(A)
```

24. This problem is about a version of the game *Jotto*. The computer chooses a random five-letter word with no repeat letters. The player gets several turns to try to guess the computer's word. On each turn, the player guesses a five-letter word and is told the number of letters that their guess has in common with the computer's word.

C:\Solutions to tasks\12.1 Exercises\12.5.24.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.24.py

```
from random import choice
words = [line.strip() for line in open('wordlist.txt')]
while True:
    w = choice(words)
    if len(w) == 5:
        flag = 0
        for i in w:
            if w.count(i) != 1:
                flag = 1
        if flag == 0:
            break
print(w)
print()
print("You have 5 attemps to guess the word")
c = 0
for i in range(5):
    letter = input("Enter a letter:  ")
    if letter in w:
        c += 1
    if i == 4:
        print("The selected word - ", w)
        print("Total number of letters guessed - ", c)
```

25. The word *part* has the interesting property that if you remove its letters one by one, each resulting step is a real word. For instance, *part* $\rightarrow$*pat* $\rightarrow$*pa* $\rightarrow$*a*. You may remove the letters in any order, and the last (single-letter) word needs to be a real word as well. Find all eight-letter words with this property.

C:\Solutions to tasks\12.1 Exercises\12.5.25.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.25.py

```
#Program written by Antonenko Igor
words = [line.strip() for line in open('wordlist.txt')]
Z = []
for word in words:
    if len(word) == 8:
        Z.append(word)
for word_start in Z:
    A = []
    A.append(word_start)
    B = A[:]
    while True:
        C = []
        for word in B:
            for i in range(len(word)):
                if word[:(0 + i)] + word[(1+i):]  in words:
                    C.append( word[:(0 + i)] + word[(1+i):])
        C = list(set(C))
        if len(C) != 0:
            B = C[:]
            A += C
        else:
            break
        s = "".join(A[-1:])
        if len(s) == 1:
            print(A)
```

26. Write a program to cheat at the game *Scrabble*.The user enters a string. Your program should return a list of all the words that can be created from those seven letters.

C:\Solutions to tasks\12.1 Exercises\12.5.26.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

12.5.26.py

```
words = [line.strip() for line in open('wordlist.txt')]

user_letters = input("Enter 7 letters in a row")
possible_words = []
for word in words:
    if len(word) == 7:
        flag = 0
        for letter in word:
            if word.count(letter) != user_letters.count(letter):
                flag = 1
                break
        if flag == 0:
            possible_words.append(word)
print(possible_words)
```

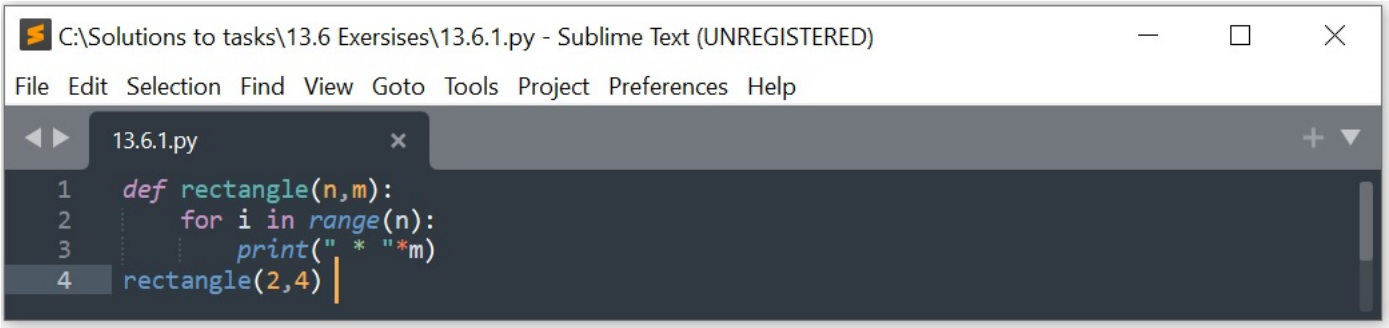
Chapter 13

Functions

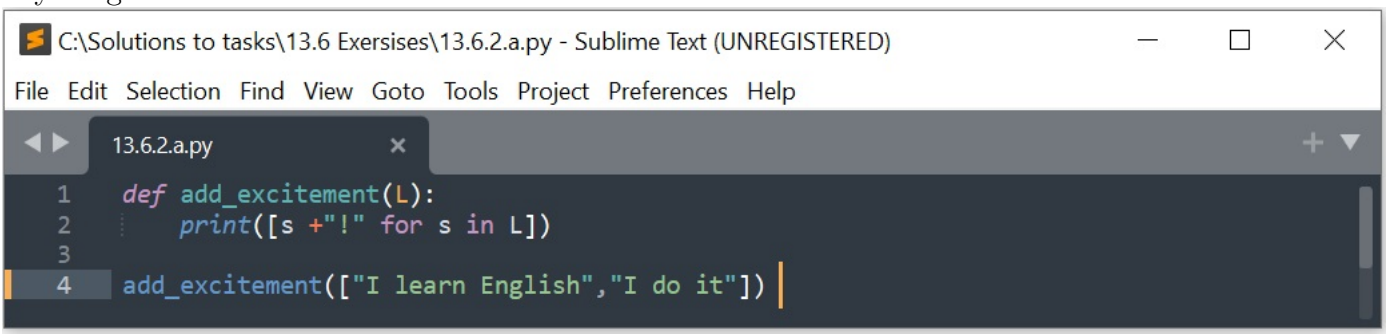
13.1 Exercises 13.6

1. Write a function called *rectangle* that takes two integers m and n as arguments and prints out an m x n consisting of asterisks. Shown below is the output of *rectangle*(2, 4)

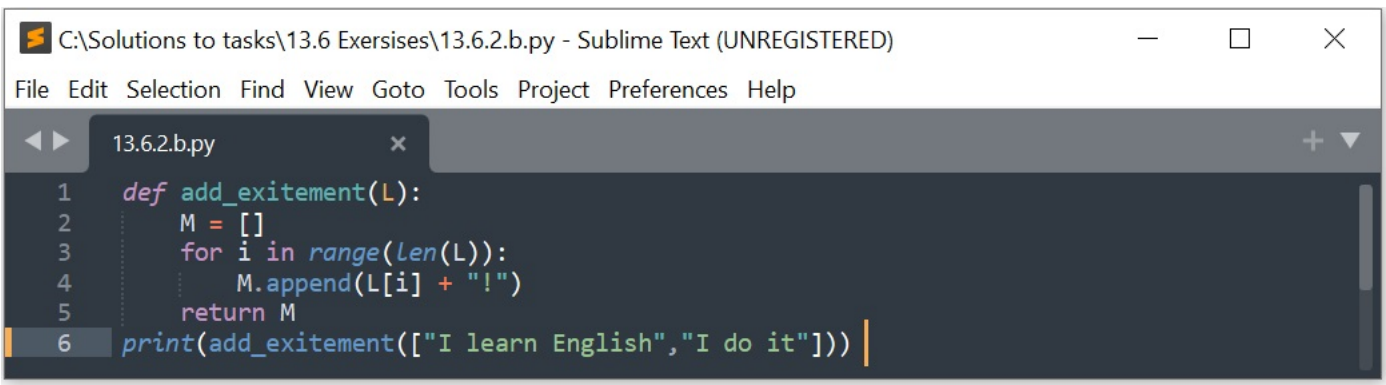
```
* * * *
* * * *
```



2.(a). Write a function called *add_ excitement* that takes a list of strings and adds an exclamation point (!) to the end of each string in the list. The program should modify the original list and not return anything.



(b). Write the same function except that it should not modify the original list and should instead return a new list.



3. Write a function called *sum_ digits* that is given an integer *num* and returns the sum of the digits of *num*.

C:\Solutions to tasks\13.6 Exersises\13.6.3.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.3.py

```
1  def sum_digits(n):
2      s = str(n)
3      n = 0
4      for i in range(len(s)):
5          n += int(s[i])
6      return n
7  print(sum_digits(555))
```

4. The *digital root* of a number n is obtained as follows: Add up the digits n to get a new number. Add up the digits of that to get another new number. Keep doing this until you get a number that has only one digit. That number is the digital root.
For example, if $n = 45893$, we add up the digits to get $4 + 5 + 8 + 9 + 3 = 29$. We then add up the digits of 29 to get $2 + 9 = 11$. We then add up the digits of 11 to get $1 + 1 = 2$. Since 2 has only one digit, 2 is our digital root.
Write a function that returns the digital root of an integer n. [Note: there is a shortcut, where the digital root is equal to n mod 9, but do not use that here.]

C:\Solutions to tasks\13.6 Exersises\13.6.4.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.4.py

```
1  def digital_root(number):
2      s = str(number)
3      while True:
4          n = 0
5          for i in range(len(s)):
6              n += int(s[i])
7          s = str(n)
8          if len(s) == 1:
9              return s
10             break
11 print(digital_root(45893))
```

5. Write a function called *first_ diff* that is given two strings and returns the first location in which the strings differ. If the strings are identical, it should return -1.

C:\Solutions to tasks\13.6 Exersises\13.6.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.5.py

```
1  def first_diff(string1,string2):
2      if len(string1) < len(string2):
3          l = len(string2)
4      else:
5          l = len(string1)
6      for i in range(l):
7          if string1 == string2:
8              return -1
9          elif string1[i] != string2[i]:
10             return ("The first location in which the strings differ. ",i)
11 print(first_diff("python","pythun"))
```

6. Write a function called *binom* that takes two integers n and k and returns the binomial coefficient $\binom{n}{k}$. The definition is $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

C:\Solutions to tasks\13.6 Exersises\13.6.6.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.6.py

```
1  from math import factorial as f
2
3  def binom(n,k):
4      bin = (f(n)/(f(k)*(f(n-k))))
5      return bin
6  print(binom(20,20))
```

7. Write a function that takes an integer n and returns a random integer with exactly n digits. For instance, if n is 3, then 125 and 593 would be valid return values, but 093 would not because that is really 93, which is a two-digit number.

C:\Solutions to tasks\13.6 Exersises\13.6.7.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.7.py

```
1  from random import randint
2
3  def ran_int(n):
4      while True:
5          x = ""
6          for i in range(n):
7              x += str(randint(0,9))
8          if x[0] != "0":
9              break
10     return x
11 print(ran_int(3))
```

8. Write a function *called number_ of_factors* that takes an integer and returns how many factors the number has.

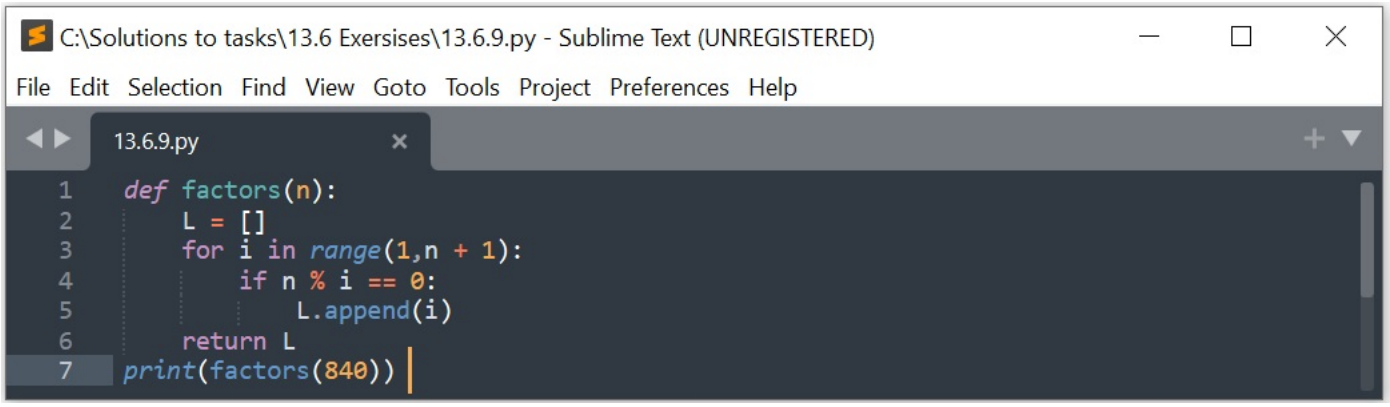
C:\Solutions to tasks\13.6 Exersises\13.6.8.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

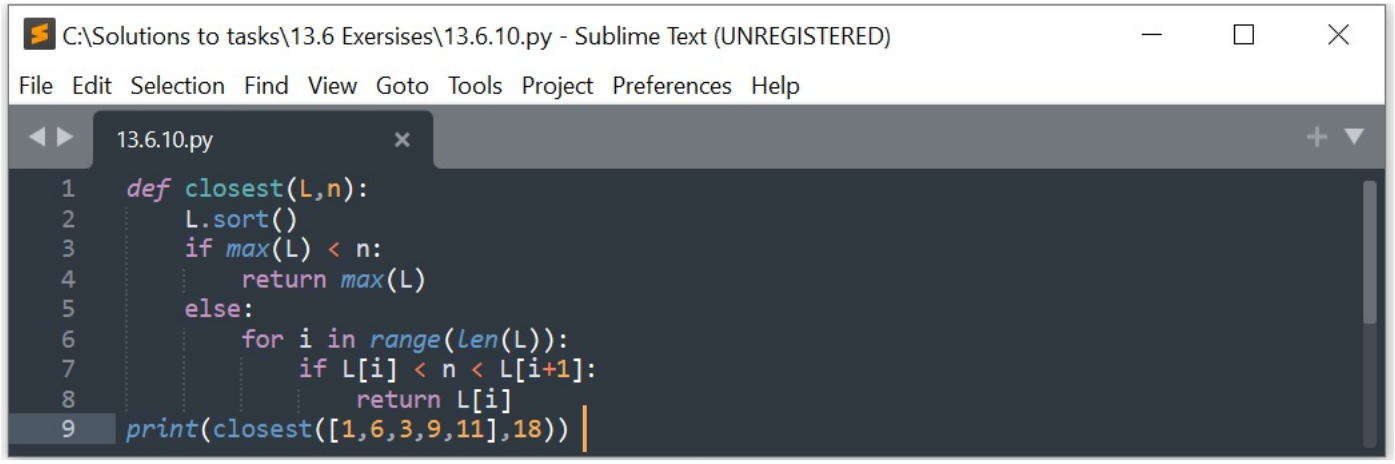
13.6.8.py

```
1  def number_of_factors(n):
2      count = 0
3      for i in range(1,n + 1):
4          if n % i == 0:
5              count += 1
6      return count
7  print(number_of_factors(840))
```

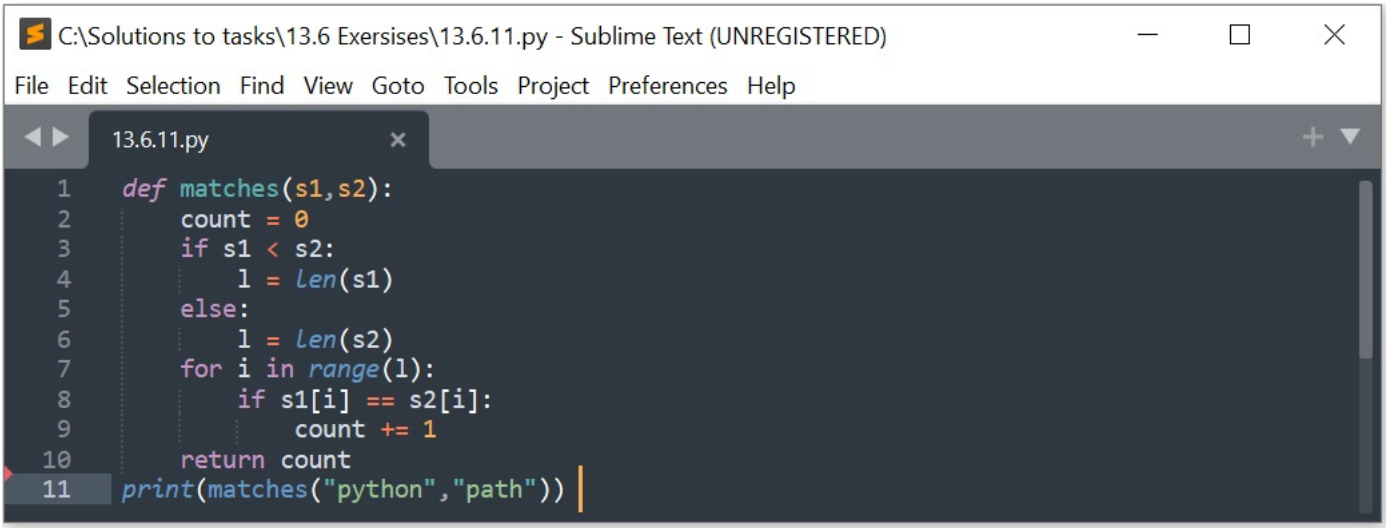
9.Write a function called *factors* that takes an integer and returns a list of its factors.



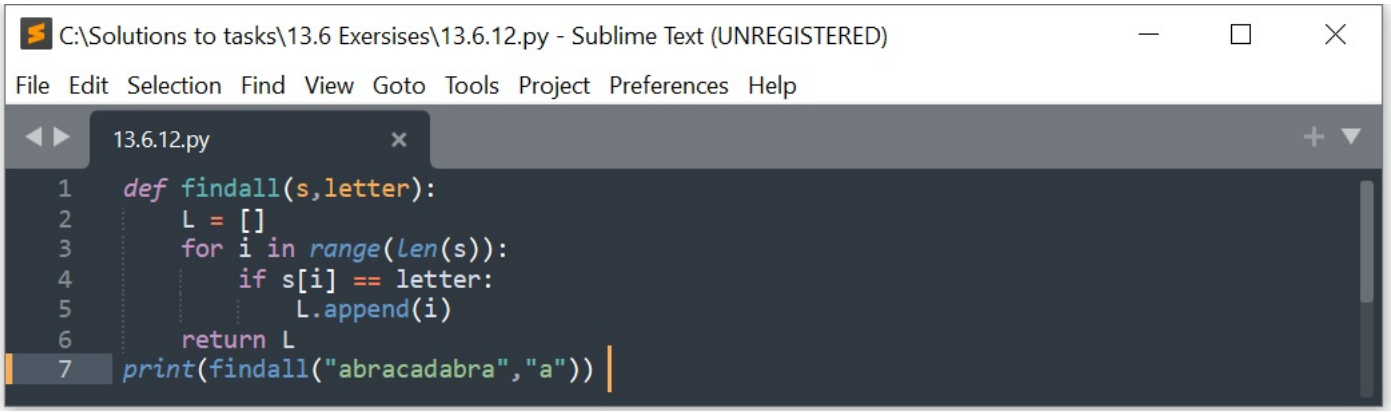
10. Write a function called *closest* that takes a list of numbers L and a number n and returns the largest element in L that is not larger than n. For instance, if $L = [1, 6, 3, 9, 11]$ and $n = 8$, then the function should return 6, because 6 is the closest thing in L to 8 that is not larger than 8. Don't worry about if all of the things in L are smaller than n.



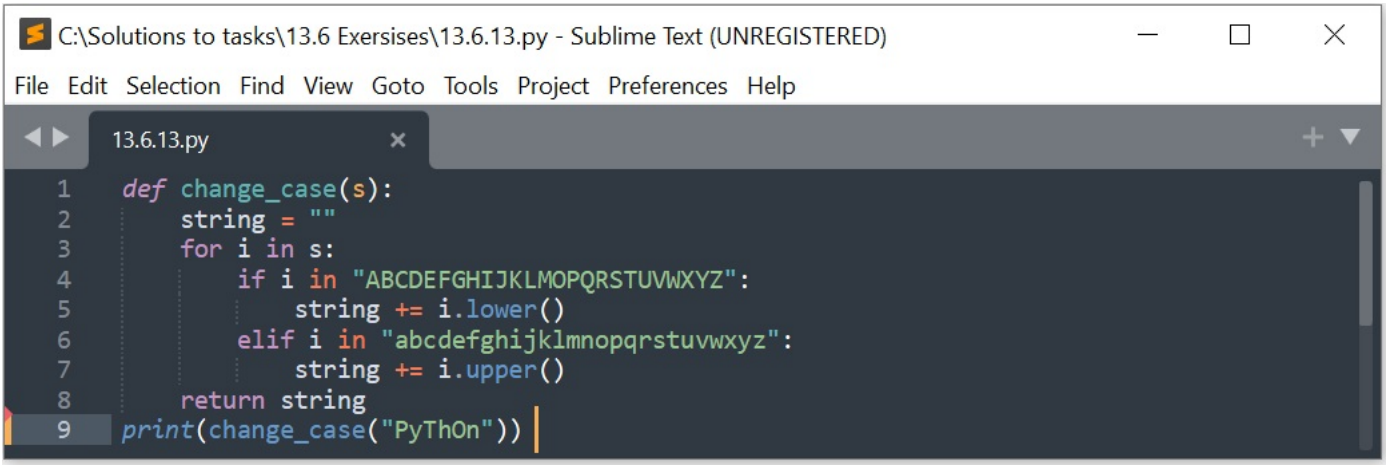
11. Write a function called *matches* that takes two strings as arguments and returns how many matches there are between the strings. A match is where the two strings have the same character at the same index. For instance, 'python' and 'path' match in the first, third, and fourth characters, so the function should return 3.



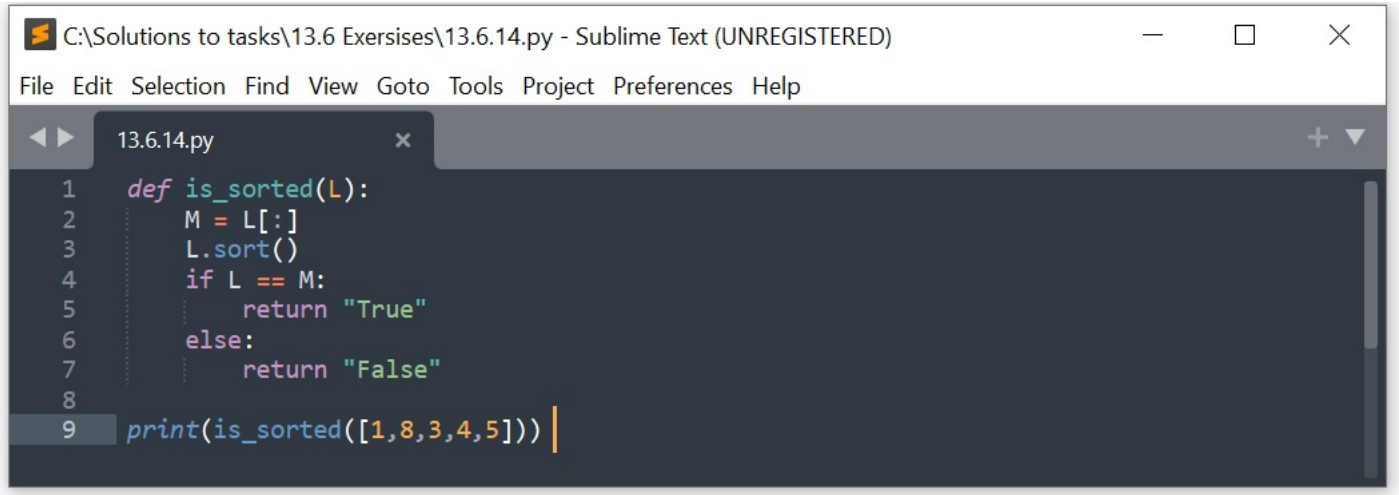
12. Recall that if s is a string, then $s.find('a')$ will find the location of the *first* a in s. The problem is that it does not find the location of every a. Write a function called *findall* that given a string and a single character, returns a list containing all of the locations of that character in the string. It should return an empty list if there are no occurrences of the character in the string.



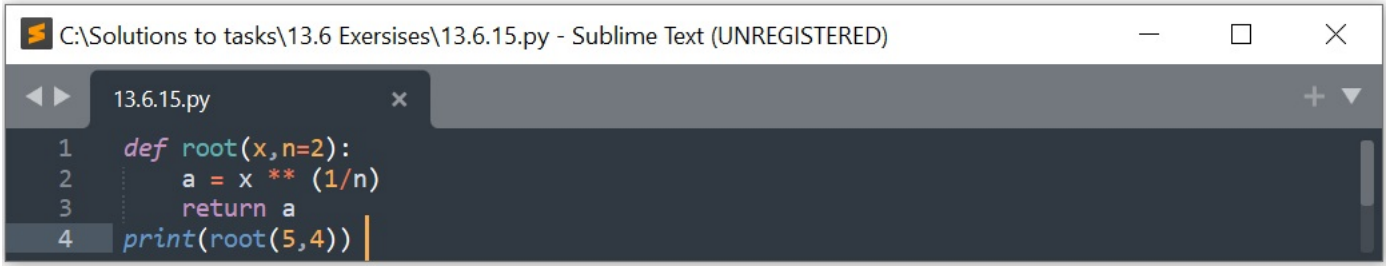
13. Write a function called *change_ case* that given a string, returns a string with each upper case letter replaced by a lower case letter and vice-versa.



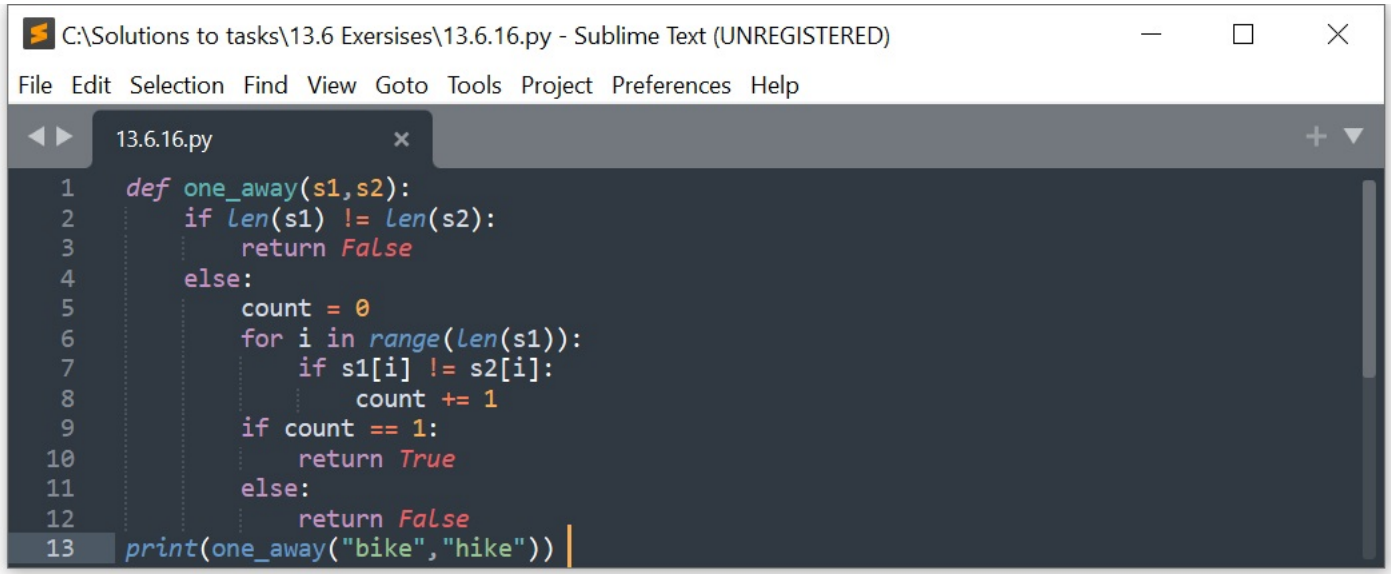
14. Write a function called*is_ sorted* that is given a list and returns **True** e if the list is sorted and **False** otherwise.



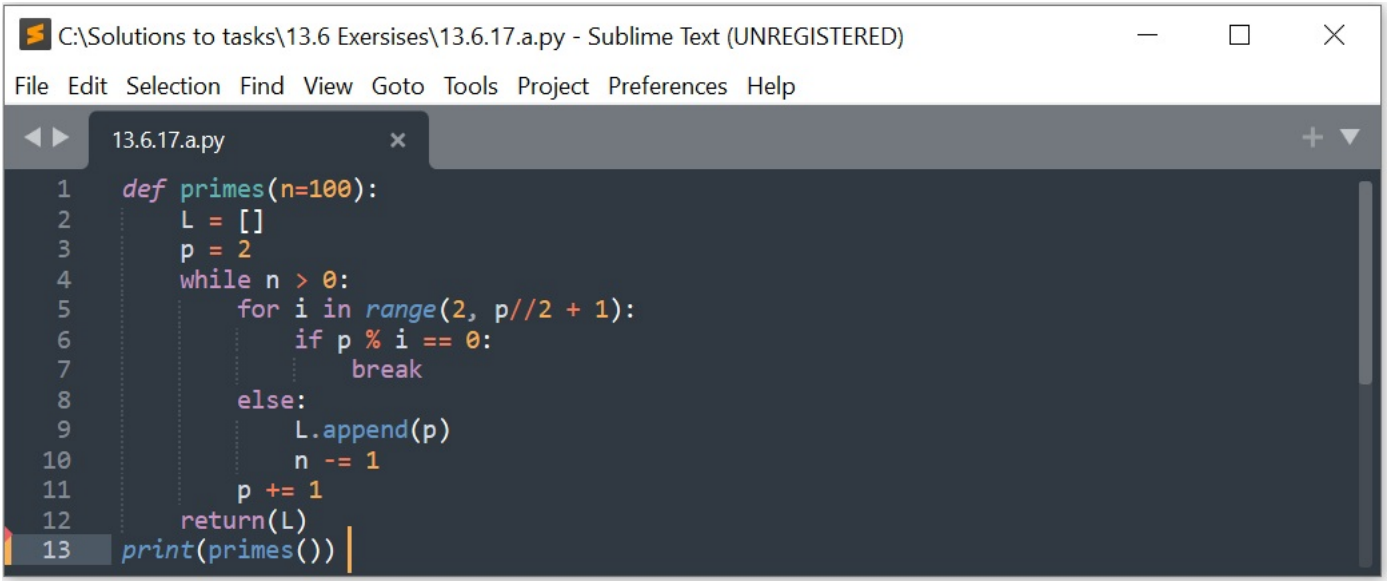
15. Write a function called *root* that is given a number x and an integer n and returns $x^{1\backslash n}$. . In the function definition, set the default value of n to 2.



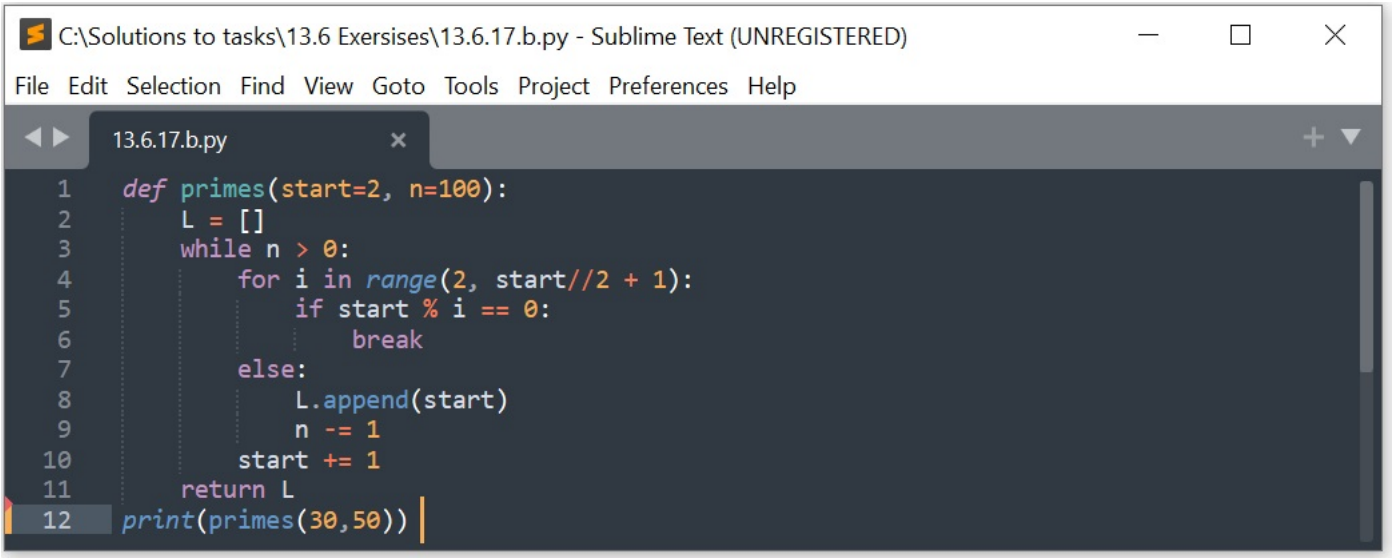
16. Write a function called *one_away* that takes two strings and returns **True** if the strings are of the same length and differ in exactly one letter, like *bike/hike* or *water/wafer*.



17.(a). Write a function called *primes* that is given a number n and returns a list of the first n primes. Let the default value of n be 100.



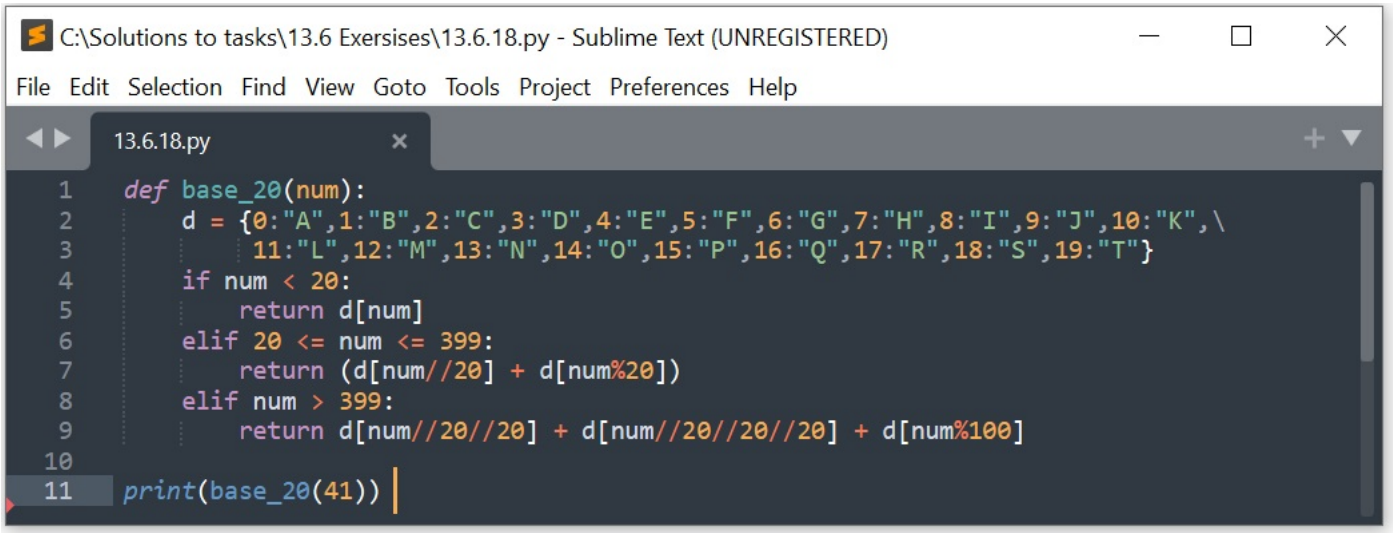
(b). Modify the function above so that there is an optional argument called *start* that allows the list to start at a value other than 2. The function should return the first n primes that are greater than or equal to *start*. The default value of *start* should be 2.



18. Our number system is called *base 10* because we have ten digits: 0, 1, . . . , 9. Some cultures, including the Mayans and Celts, used a base 20 system. In one version of this system, the 20 digits are represented by the letters A through T. Here is a table showing a few conversions:

10	20	10	20	10	20	10	20
0	A	8	I	16	Q	39	BT
1	B	9	J	17	R	40	CA
2	C	10	K	18	S	41	CB
3	D	11	L	19	T	60	DA
4	E	12	M	20	BA	399	TT
5	F	13	N	21	BB	400	BAA
6	G	14	O	22	BC	401	BAB
7	H	15	P	23	BD	402	BAC

Write a function called *base20* that converts a base 10 number to base 20. It should return the result as a string of base 20 digits. One way to convert is to find the remainder when the number is divided by 20, then divide the number by 20, and repeat the process until the number is 0. The remainders are the base 20 digits in reverse order, though you have to convert them into their letter equivalents.



19. Write a function called *verbose* that, given an integer less than 10^{15} , returns the name of the integer in English. As an example, *verbose(123456)* should return *one hundred twenty-three thousand, four hundred fifty-six.*

C:\Solutions to tasks\13.6 Exersises\13.6.19.py - Sublime Text (UNREGISTERED)

13.6.19.py

```
def verbose(num):
    d = { 0 : 'zero', 1 : 'one', 2 : 'two', 3 : 'three', 4 : 'four', 5 : 'five',
          6 : 'six', 7 : 'seven', 8 : 'eight', 9 : 'nine', 10 : 'ten',
          11 : 'eleven', 12 : 'twelve', 13 : 'thirteen', 14 : 'fourteen',
          15 : 'fifteen', 16 : 'sixteen', 17 : 'seventeen', 18 : 'eighteen',
          19 : 'nineteen', 20 : 'twenty',
          30 : 'thirty', 40 : 'forty', 50 : 'fifty', 60 : 'sixty',
          70 : 'seventy', 80 : 'eighty', 90 : 'ninety' }
    if (num < 20):
        return d[num]
    if (num < 10**2):
        if num % 10 == 0:
            return d[num]
        else:
            return d[num // 10 * 10] + '-' + d[num % 10]
    if (num < 10**3):
        if num % 100 == 0:
            return d[num // 100] + ' hundred'
        else:
            return d[num // 100] + ' hundred  ' + verbose(num % 100)
    if (num < 10**6):
        if num % 10**3 == 0:
            return verbose(num // 10**3) + ' thousand'
        else:
            return verbose(num // 10**3) + ' thousand, ' + verbose(num % 10**3)
    if (num < 10**9):
        if (num % 10**6) == 0:
            return verbose(num // 10**6) + ' million'
        else:
            return verbose(num // 10**6) + ' million, ' + verbose(num % 10**6)
    if (num < 10**12):
        if (num % 10**9) == 0:
            return verbose(num // 10**9) + ' billion'
        else:
            return verbose(num // 10**9) + ' billion, ' + verbose(num % 10**9)
    if (num % 10**12 == 0):
        return verbose(num // 10**12) + ' trillion'
    else:
        return verbose(num // 10**12) + ' trillion, ' + verbose(num % 10**12)
print(verbose(1234569854213))
```

20. Write a function called *merge* that takes two already sorted lists of possibly different lengths, and merges them into a single sorted list.

(a). Do this using the *sort* method.

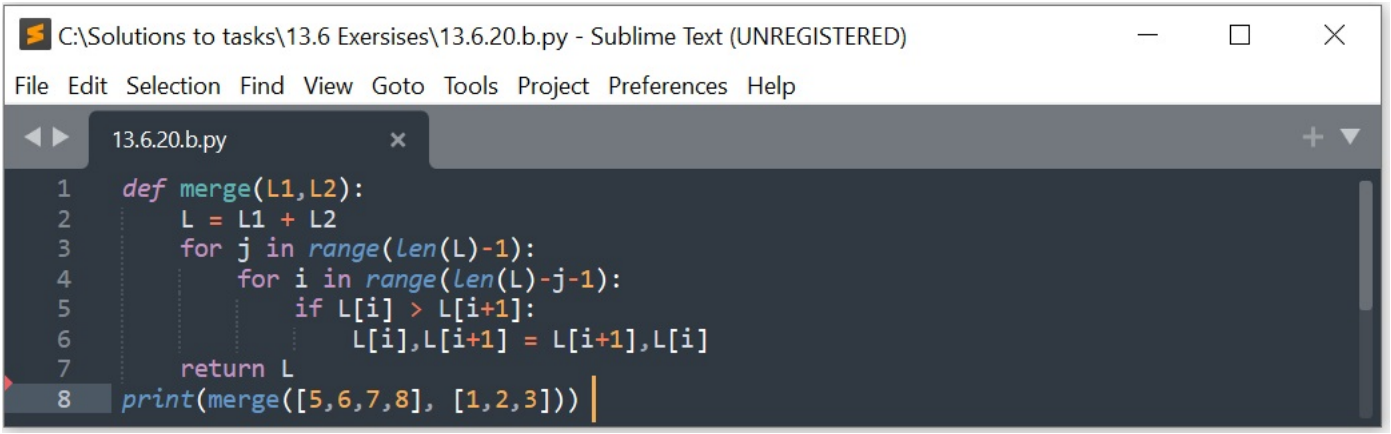
C:\Solutions to tasks\13.6 Exersises\13.6.20.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

13.6.20.a.py

```
def merge(L1,L2):
    L = L1 + L2
    L.sort()
    return L
print(merge([5,6,7,8,9], [1,2,3]))
```

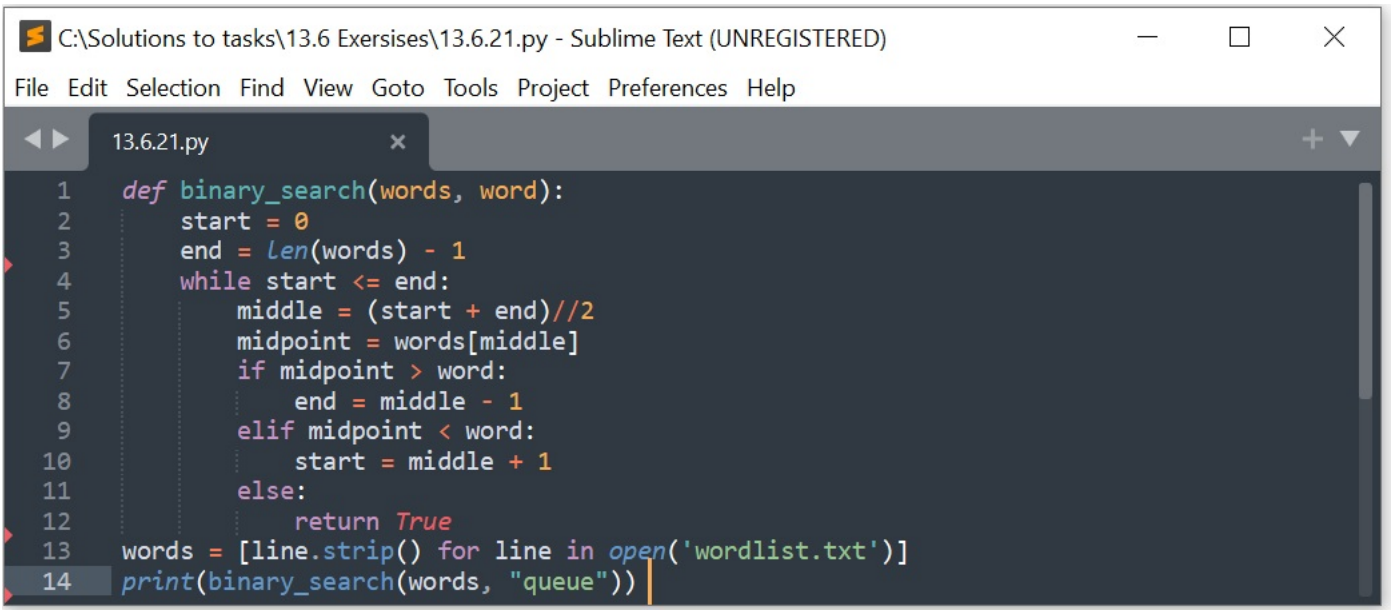
(b). Do this without using the *sort* method.



21. In Chapter 12, the way we checked to see if a word w was a real word was:

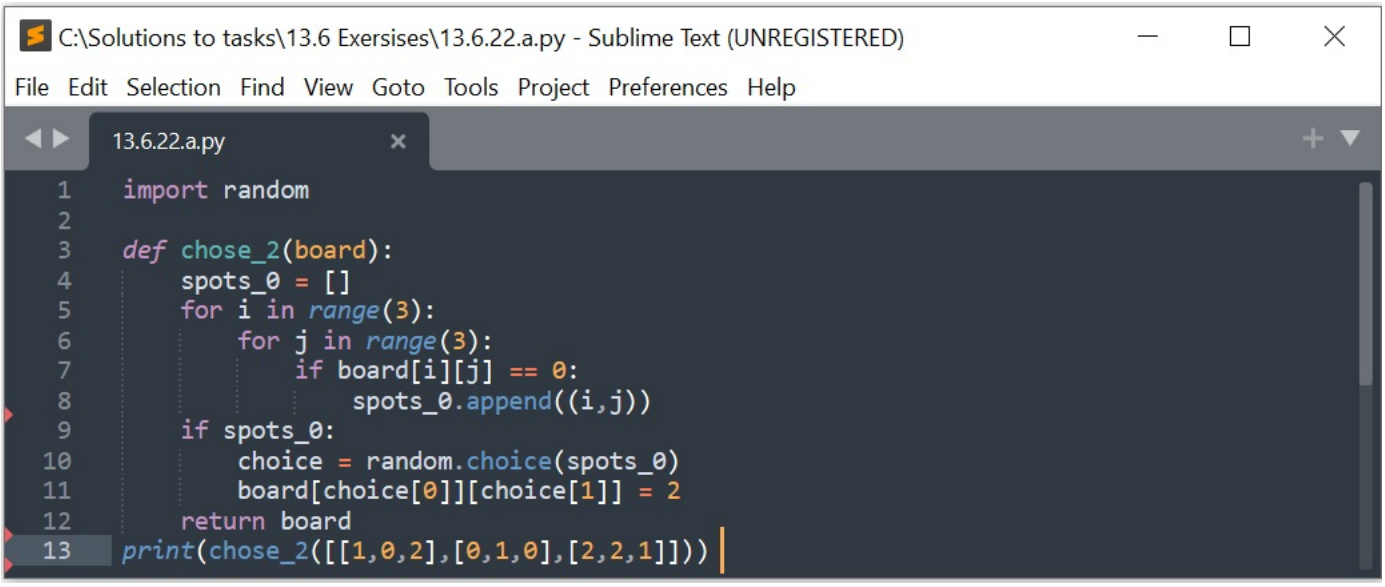
```
if w in words:
```

where *words* was the list of words generated from a wordlist. This is unfortunately slow, but there is a faster way, called a *binary search*. To implement a binary search in a function, start by comparing w with the middle entry in *words*. If they are equal, then you are done and the function should return **True**. On the other hand, if w comes before the middle entry, then search the first half of the list. If it comes after the middle entry, then search the second half of the list. Then repeat the process on the appropriate half of the list and continue until the word is found or there is nothing left to search, in which case the function short return **False**. The $<$ and $>$ operators can be used to alphabetically compare two strings.

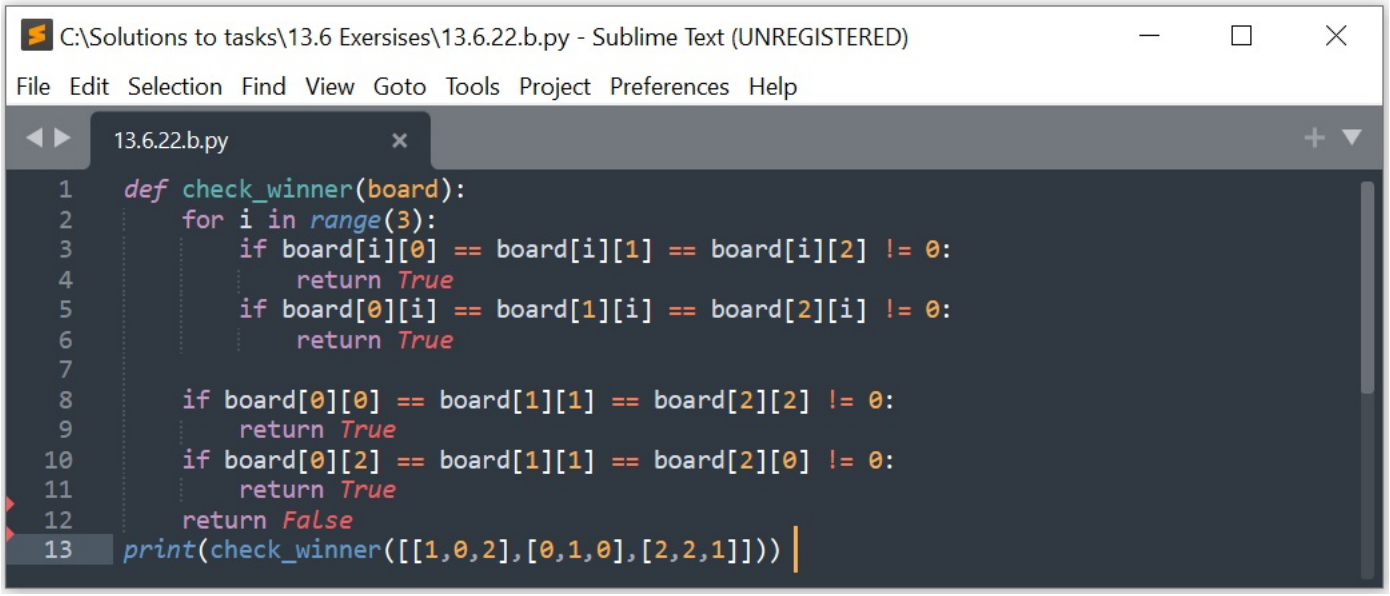


22. A Tic-tac-toe board can be represented be a 3×3 two-dimensional list, where zeroes stand for empty cells, ones stand for X's and twos stand for O's.

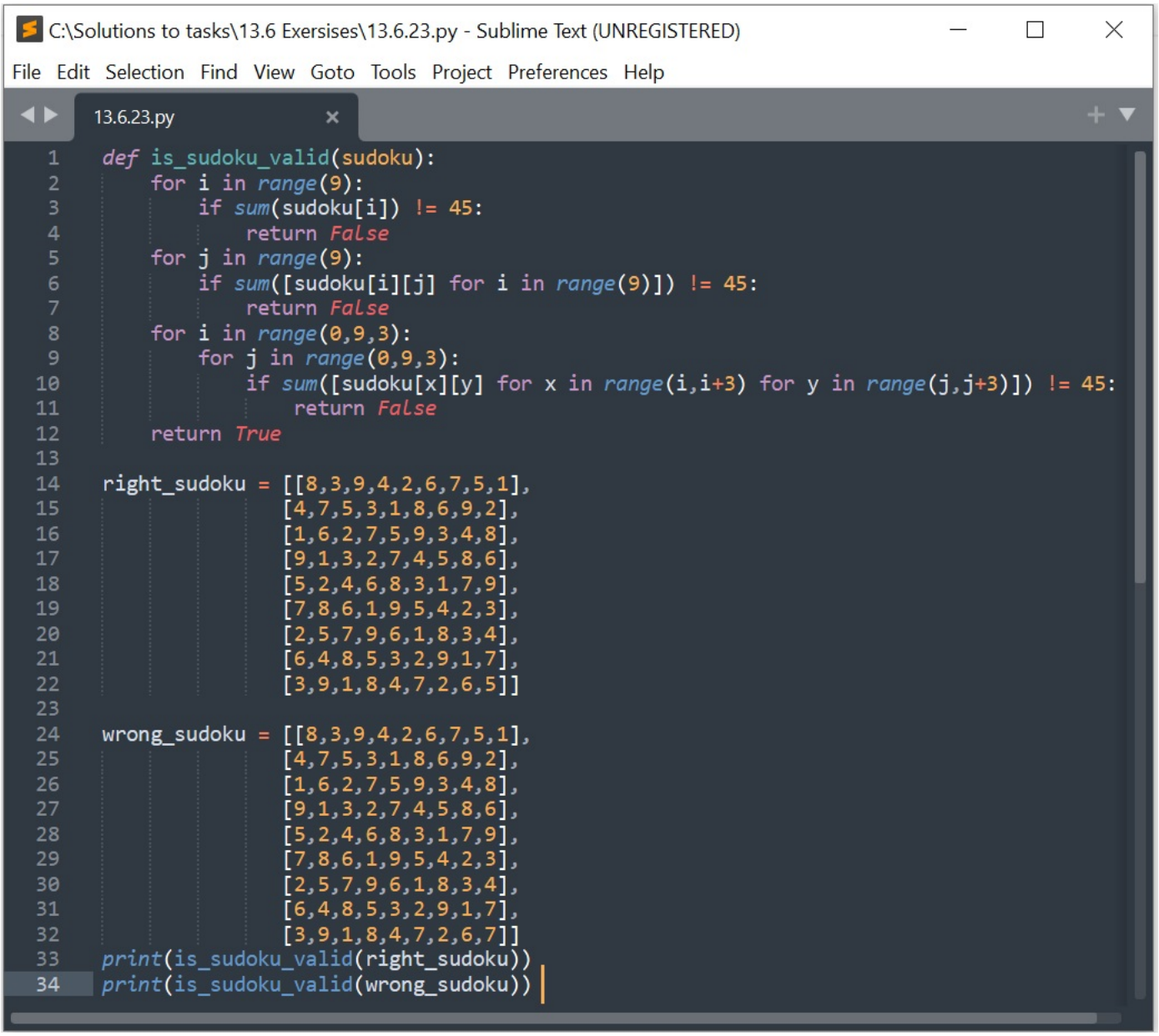
(a). Write a function that is given such a list and randomly chooses a spot in which to place a 2. The spot chosen must currently be a 0 and a spot must be chosen.



(b). Write a function that is given such a list and checks to see if someone has won. Return **True** if there is a winner and **False** otherwise.



23. Write a function that is given a 9×9 potentially solved Sudoku and returns **True** if it is solved correctly and **False** if there is a mistake. The Sudoku is correctly solved if there areno repeated numbers in any row or any column or in any of the nine "blocks."


C:\Solutions to tasks\13.6 Exersises\13.6.23.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
13.6.23.py
def is_sudoku_valid(sudoku):
for i in range(9):
if sum(sudoku[i]) != 45:
return False
for j in range(9):
if sum([sudoku[i][j] for i in range(9)]) != 45:
return False
for i in range(0,9,3):
for j in range(0,9,3):
if sum([sudoku[x][y] for x in range(i,i+3) for y in range(j,j+3)]) != 45:
return False
return True
right_sudoku = [[8,3,9,4,2,6,7,5,1],
[4,7,5,3,1,8,6,9,2],
[1,6,2,7,5,9,3,4,8],
[9,1,3,2,7,4,5,8,6],
[5,2,4,6,8,3,1,7,9],
[7,8,6,1,9,5,4,2,3],
[2,5,7,9,6,1,8,3,4],
[6,4,8,5,3,2,9,1,7],
[3,9,1,8,4,7,2,6,5]]
wrong_sudoku = [[8,3,9,4,2,6,7,5,1],
[4,7,5,3,1,8,6,9,2],
[1,6,2,7,5,9,3,4,8],
[9,1,3,2,7,4,5,8,6],
[5,2,4,6,8,3,1,7,9],
[7,8,6,1,9,5,4,2,3],
[2,5,7,9,6,1,8,3,4],
[6,4,8,5,3,2,9,1,7],
[3,9,1,8,4,7,2,6,7]]
print(is_sudoku_valid(right_sudoku))
print(is_sudoku_valid(wrong_sudoku))

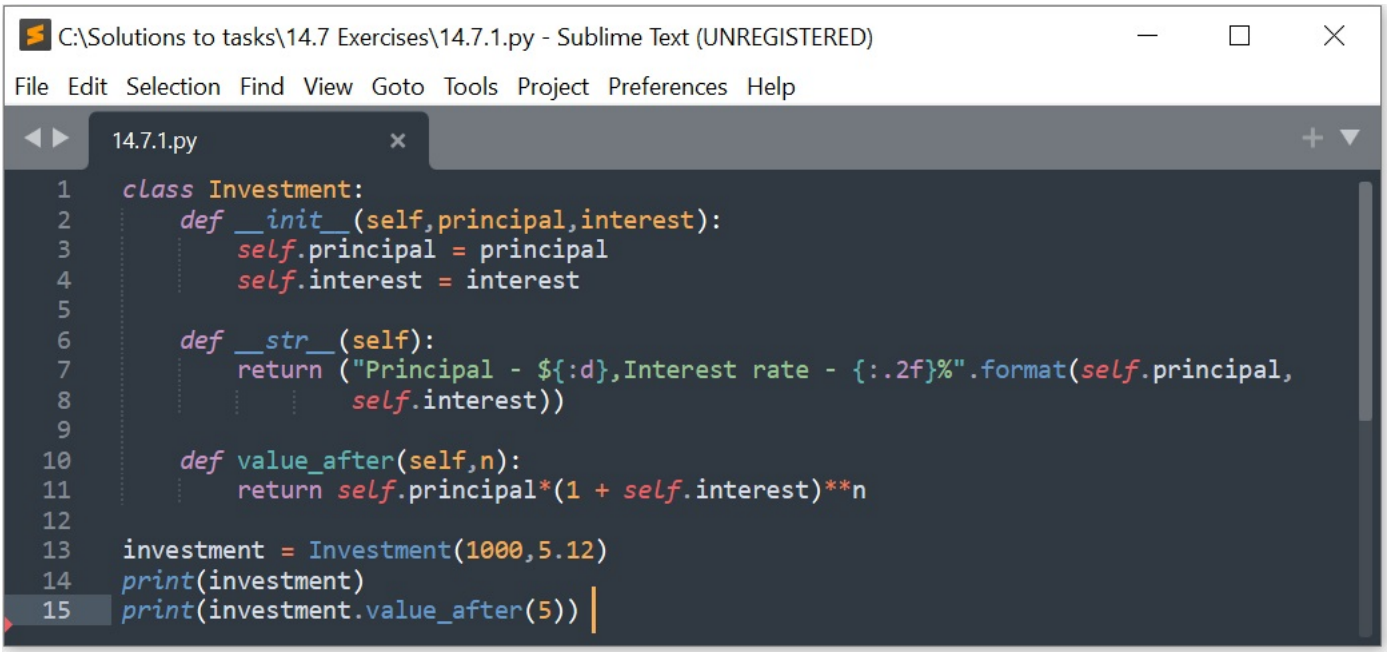
Chapter 14

Object-Oriented Programming

14.1 Exercises 14.7

1. Write a class called *Investment* with fields called *principal* and *interest* . The constructor should set the values of those fields. There should be a method called *value_after* that returns the value of the investment after n years. The formula for this is $p(1+i)^n$, where p isthe principal, and i is the interest rate. It should also use the special method *__str__* so that printing the object will result in something like below:

Principal - $1000.00, Interest rate - 5.12%



2. Write a class called *Product*. The class should have fields called *name, amount* and *price*, holding the product's name, the number of items of that product in stock, and the regular price of the product. There should be a method *get_price*, that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called *make_purchase* that receives the number of items to be bought and decreases *amount* by that much.

C:\Solutions to tasks\14.7 Exercises\14.7.2.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

14.7.2.py

```
class Product:
    def __init__(self,name,amount,price):
        self.name = n
        self.amount = a
        self.price = p

    def get_price(self):
        return f"The regular price - {round(self.amount * self.price, 2)}"

    def make_purchase(self):
        if self.amount < 10:
            return f"The discounted price - {round(a*p, 2)}"

        if self.amount >= 10 and self.amount < 100:
            return f"The discounted price - {round(a * p - (int(a*p*10/100)), 2)}"

        if self.amount > 100:
            return f"The discounted price - {round(a*p - (a*p*20/100), 2)}"

n = input("The product's name: ")
a = eval(input("The number of items of that product: "))
p = eval(input("The regular price: "))

c = Product(n,a,p)
print(c.get_price())
print(c.make_purchase())
```

3.Write a class called *Password_manager*. The class should have a list called *old_passwords* that holds all of the user's past passwords. The last item of the list is the user's current password. There should be a method called *get_password* that returns the current password and a method called *set_password* that sets the user's password. The *set_password* method should only change the password if the attempted password is different from all the user's past passwords. Finally, create a method called *is_correct* that receives a string and returns a boolean **True** or **False** depending on whether the string is equal to the current password or not.

C:\Solutions to tasks\14.7 Exercises\14.7.3.py - Sublime Text (UNREGISTERED)

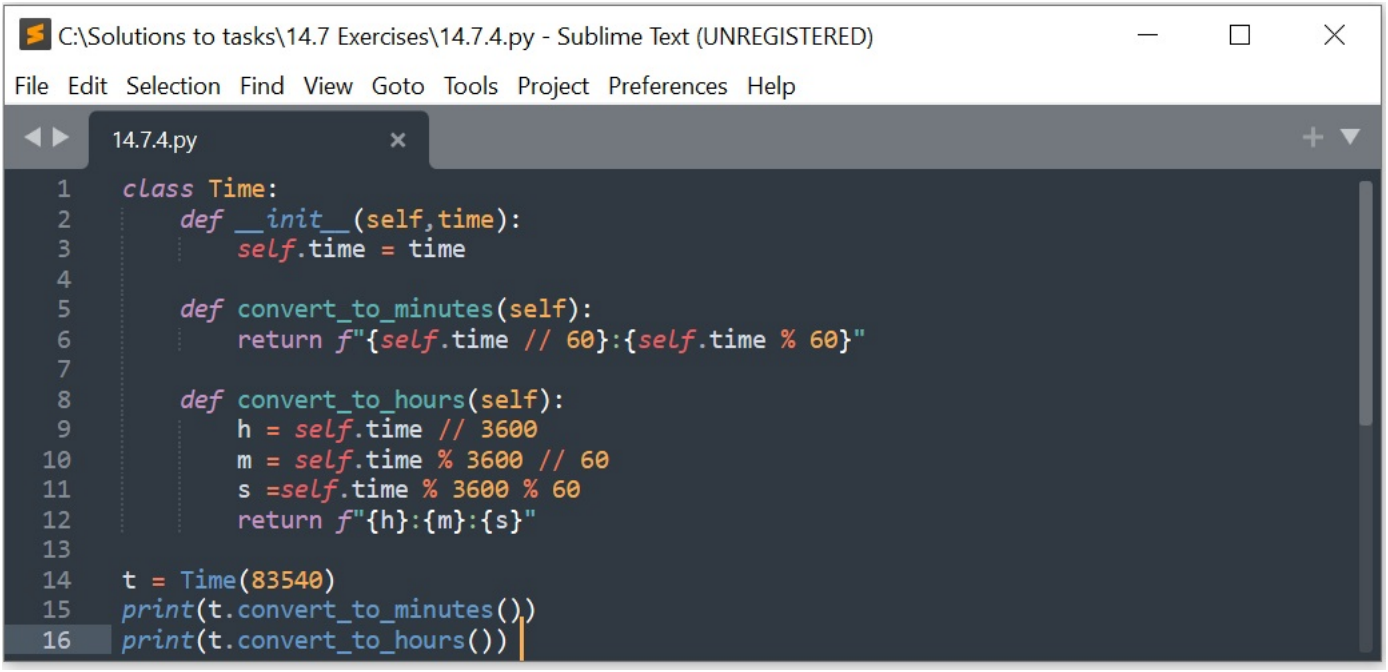
File Edit Selection Find View Goto Tools Project Preferences Help

14.7.3.py

```
class Password_manager:
    def __init__(self):
        self.old_passwords = []

    def get_password(self):
        return self.old_passwords[-1]

    def set_passwords(self,new_pass):
        if new_pass in self.old_passwords:
            return f"The password is in the list. Enter a new password."
        else:
            self.old_passwords.append(new_pass)
            return f"The password has been set."

    def is_correct(self, your_password):
        if your_password == self.old_passwords[-1]:
            return True
        else:
            return False
```

```
20
21  a = Password_manager()
22  print(a.set_passwords("password123")) #Output - The password has been set.
23  print(a.set_passwords("password123")) #Output - The password is in the list.
24                                        #Enter a new password.
25  print(a.get_password()) #Output - password123
26  print(a.is_correct("password123")) #Output - True
27  print(a.is_correct("password")) #Output - False
```

4. Write a class called *Time* whose only field is a time in seconds. It should have a method called *convert_to_minutes* that returns a string of minutes and seconds formatted as in the following example: if seconds is 230, the method should return '5:50'. It should also have a method called *convert_to_hours* that returns a string of hours, minutes, and seconds formatted analogously to the previous method.



5. Write a class called *Wordplay*. It should have a field that holds a list of words. The user of the class should pass the list of words they want to use to the class. There should be the following methods:

- *words_with_length(length)* — returns a list of all the words of length *length*
- *starts_with(s)* — returns a list of all the words that start with *s*
- *ends_with(s)* — returns a list of all the words that end with *s*
- *palindromes()* — returns a list of all the palindromes in the list
- *only(L)* — returns a list of the words that contain only those letters in *L*
- *avoids(L)* — returns a list of the words that contain none of the letters in *L*

C:\Solutions to tasks\14.7 Exercises\14.7.5.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

14.7.5.py

```
class Wordplay:
    def __init__(self, words):
        self.words = words

    def words_with_length(self, length):
        return [w for w in self.words if len(w) == length]

    def starts_with(self, s):
        return [w for w in self.words if w[0] == s]

    def ends_with(self, s):
        return [w for w in self.words if w[-1] == s]

    def palindromes(self):
        return [w for w in self.words if w[:] == w[: : -1]]

    def only(self, L1):
        M = []
        for w in self.words:
            flag = 0
            for i in L1:
                if i not in w:
                    flag = 1
            if flag == 0:
                M.append(w)
        return M

    def avoids(self, L2):
        M = []
        for w in self.words:
            flag = 0
            for i in L2:
                if i in w:
                    flag = 1
            if flag == 0:
                M.append(w)
        return M
```

6. Write a class called *Converter*. The user will pass a length and a unit when declaring an object from the class—for example, $c = Converter(9, \text{'inches'})$.The possible units are inches, feet, yards, miles, kilometers, meters, centimeters, and millimeters. For each of these units there should be a method that returns the length converted into those units. For example, using the *Converter* object created above, the user could call *c.feet* and should get 0.75 as the result..

C:\Solutions to tasks\14.7 Exercises\14.7.6.py - Sublime Text (UNREGISTERED)

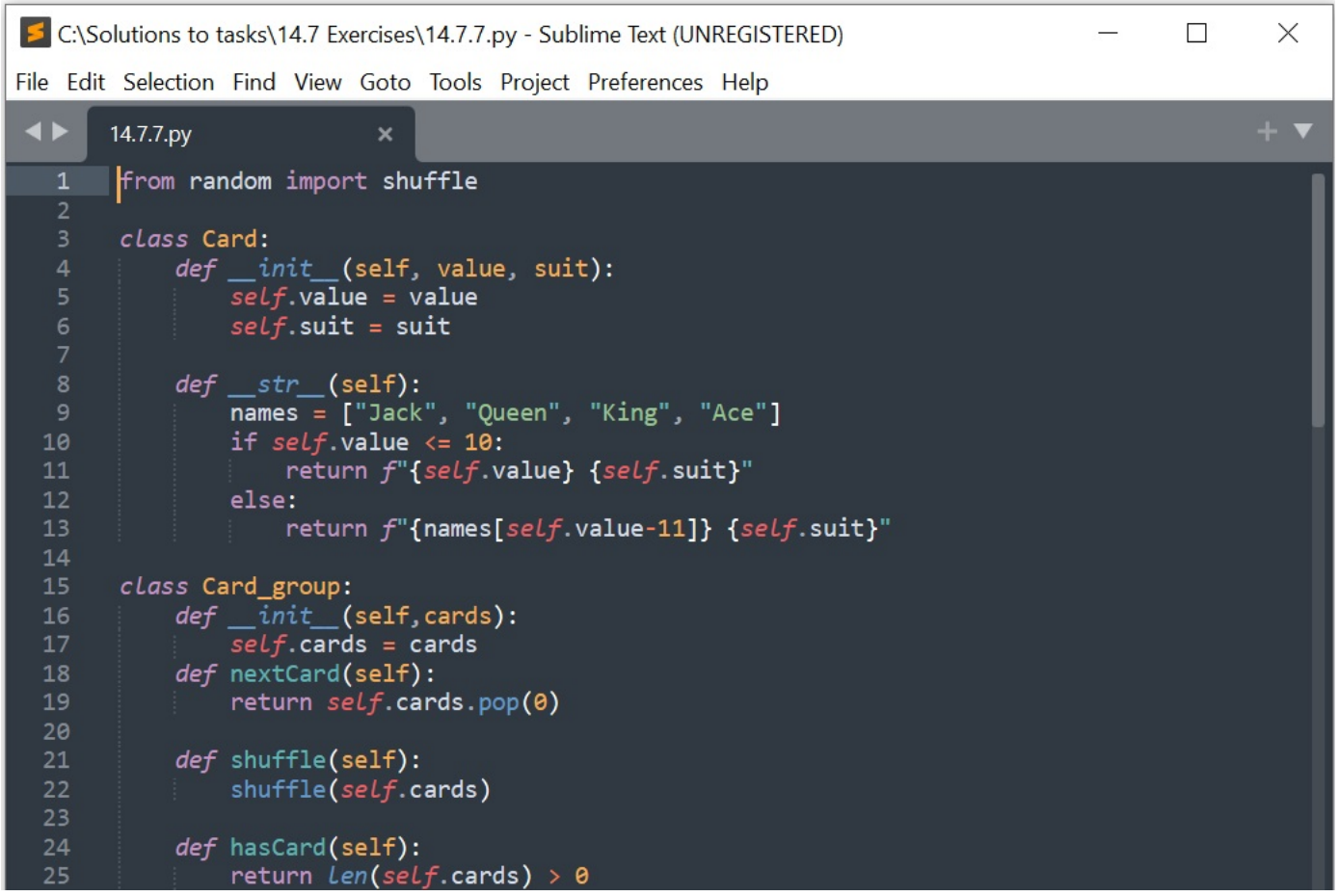
File Edit Selection Find View Goto Tools Project Preferences Help

14.7.6.py

```
class Converter:

    def __init__(self,length,unit):
        self.length = length
        self.unit = unit

    def transform(self, unit_transform):
        unit_list = ["inches","feet","yards","miles","kilometrs","meters",\
        "centimetrs","millimetrs"]
```

```
            value_in_foot = [12,1,1/3,1/5280,1/3281,1/3.281,1/30.48,1/304.8]
            if unit_transform in unit_list:
                length_transform = round(self.length*value_in_foot[unit_list.index(
                                         unit_transform)]/value_in_foot
                                         [unit_list.index(self.unit)], 6)
            return f"{length_transform} {unit_transform}"

        def inches(self):
            return self.transform("inches")
        def feet(self):
            return self.transform("feet")
        def yards(self):
            return self.transform("yards")
        def miles(self):
            return self.transform("miles")
        def kilometrs(self):
            return self.transform("kilometrs")
        def meters(self):
            return self.transform("meters")
        def centimetrs(self):
            return self.transform("centimetrs")
        def millimetrs(self):
            return self.transform("millimetrs")
    c = Converter(9,"inches")
    print(c.inches())
    print(c.feet())
    print(c.millimetrs())
```

7.Use the *Standard_ deck* class of this section to create a simplified version of the game *War*. In this game, there are two players. Each starts with half of a deck. The players each deal the top card from their decks and whoever has the higher card wins the other player's cards and adds them to the bottom of his deck. If there is a tie, the two cards are eliminated from play (this differs from the actual game, but is simpler to program). The game ends when one player runs out of cards.



```
26
27      def size(self):
28          return len(self.cards)
29
30  class Standart_deck(Card_group):
31      def __init__(self):
32          self.cards = []
33
34          for s in ["♣", "♦", "♥", "♠"]:
35              for v in range(2,15):
36                  self.cards.append(Card(v, s))
37
38  class Player(Card_group):
39      def __init__(self):
40          self.cards = []
41
42  player_1 = Player()
43  player_2 = Player()
44
45
46  deck = Standart_deck()
47  deck.shuffle()
48  player_1.cards = deck.cards[:26]
49  player_2.cards = deck.cards[26:]
50
51
52  print("For the next move, press Enter")
53  sign = ""
54  while player_1.hasCard() and player_2.hasCard():
55      card1 = player_1.nextCard()
56      card2 = player_2.nextCard()
57
58      if card1.value == card2.value:
59          sign = "="
60
61      elif card1.value > card2.value:
62          player_1.cards.extend([card1,card2])
63          sign = ">"
64      elif card2.value > card1.value:
65          player_2.cards.extend([card2,card1])
66          sign = "<"
67      print("player_1 -",card1, sign ,card2,"- player_2")
68      print(f"The mumber of the cards : player_1 - {player_1.size()},\
69            player_2 - {player_2.size()}")
70      input()
71  print("Game over.")
```

8.Write a class that inherits from the *Card_group* class of this chapter. The class should represent a deck of cards that contains only hearts and spaces, with only the cards 2 through 10 in each suit. Add a method to the class called *next2* that returns the top two cards from the deck.

C:\Solutions to tasks\14.7 Exercises\14.7.8.py - Sublime Text (UNREGISTERED)

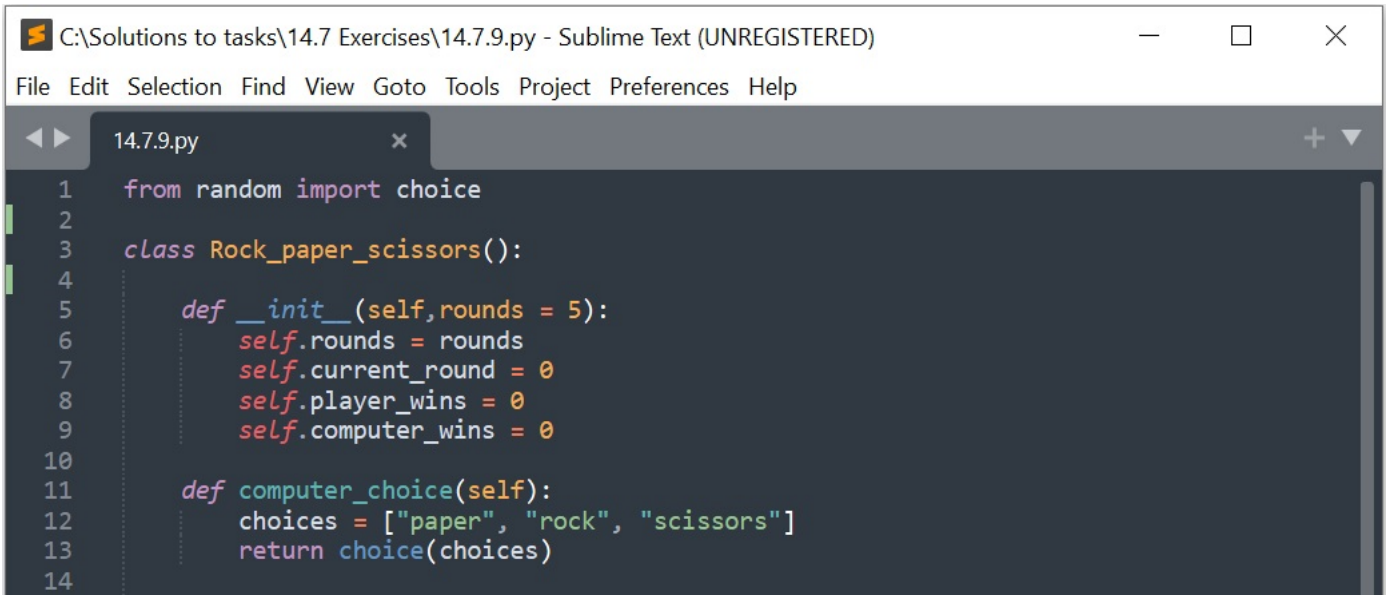
File Edit Selection Find View Goto Tools Project Preferences Help

14.7.8.py

```
1  from random import shuffle
2
3  class Card:
4      def __init__(self, value, suit):
5          self.value = value
6          self.suit = suit
7
```

```
    def __str__(self):
        names = ["Jack", "Queen", "King", "Ace"]
        if self.value <= 10:
            return f"{self.value} {self.suit}"
        else:
            return f"{names[self.value-11]} {self.suit}"

class Card_group:
    def __init__(self,cards):
        self.cards = cards
    def nextCard(self):
        return self.cards.pop(0)

    def shuffle(self):
        shuffle(self.cards)

    def hasCard(self):
        return len(self.cards) > 0

    def size(self):
        return len(self.cards)

class Deck_two_suits(Card_group):
    def __init__(self):
        self.cards = []
        for s in ["♥", "♠"]:
            for v in range(2,10):
                self.cards.append(Card(v,s))

    def next2(self):
        return f"{self.cards[0]} {self.cards[1]}"

deck = Deck_two_suits()
deck.shuffle()
print(deck.next2())
```

9. Write a class called *Rock_paper_scissors* that implements the logic of the game Rock-paper-scissors. For this game the user plays against the computer for a certain number of rounds. Your class should have fields for the how many rounds there will be, the current round number, and the number of wins each player has. There should be methods for getting the computer's choice, finding the winner of a round, and checking to see if someone has one the (entire) game. You may want more methods.



```
    def find_winner(self,player_choice, computer_choice):
        if player_choice == computer_choice:
            return f"Tie"
        elif (player_choice == "rock" and computer_choice == "scissors") \
             or (player_choice == "paper" and computer_choice == "rock") \
             or (player_choice == "scissors" and computer_choice == "paper"):
            self.player_wins += 1
            return f"The player won this round."
        else:
            self.computer_wins += 1
            return f"The computer won this round."

    def check_game_winner(self):
        if self.player_wins == self.computer_wins:
            return f"Tie!"
        elif self.player_wins > self.computer_wins:
            return f"The player won the game!"
        else:
            return f"The computer won the game!"

    def play_round(self,player_choice):
        self.current_round += 1
        computer_choice = self.computer_choice()
        print(f"Round {self.current_round}: The player's choice - {player_choice},\
            The computer's choice - {computer_choice}")

        result = self.find_winner(player_choice, computer_choice)
        print(result)

        if self.current_round == self.rounds:
            game_winner = self.check_game_winner()
            print(game_winner)
game = Rock_paper_scissors(3)
while game.current_round < game.rounds:
    player_choice = input("Your choice from (paper, rock, scissors) - ")
    game.play_round(player_choice)
```

10.(a). Write a class called *Connect4* that implements the logic of a Connect4 game. Use the *Tic_tac_toe* class from this chapter as a starting point.

C:\Solutions to tasks\14.7 Exercises\14.7.10.a.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

14.7.10.a.py

```
class Connect4:
    def __init__(self):
        self.board = [[' ' for _ in range(7)] for _ in range(6)]
        self.players = ['X', 'O']
        self.player = 0

    def print_board(self):
        for row in self.board:
            print("  |  ".join(row) + "  |")
            print("-" * 34)
        print("     ".join([str(i) for i in range(7)]))
        print()

    def get_spots(self, col):
        if 0 <= col < 7 and self.board[0][col] == ' ':
            for i in range(5, -1, -1):
                if self.board[i][col] == ' ':
                    self.board[i][col] = self.players[self.player]
                    return True
        return False
```

```
21
22      def check_winner(self):
23          directions = [(1,0),(0,1),(1,1),(1,-1)]
24
25          for i in range(6):
26              for j in range(7):
27                  if self.board[i][j] != " ":
28                      for d in directions:
29                          x, y = i, j
30                          count = 0
31                          while 0 <= x <= 6 and 0 <= y <= 7 and self.board[x][y] \
32                          == self.board[i][j]:
33                              count += 1
34                              x, y = x + d[0], y + d[1]
35                              if count == 4:
36                                  return True
37          return False
```

(b). Use the *Connect4* class to create a simple text-based version of the game.

C:\Solutions to tasks\14.7 Exercises\14.7.10.b.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

14.7.10.b.py

```
1   """
2   To display it correctly, run the program in  py.exe
3   """
4   class Connect4:
5       def __init__(self):
6           self.board = [[' ' for _ in range(7)] for _ in range(6)]
7           self.players = ['X', 'O']
8           self.player = 0
9       def print_board(self):
10          for row in self.board:
11              print("  |   ".join(row) + "   |")
12              print("-" * 47)
13          print("      ".join([str(i) for i in range(7)]))
14          print()
15      def get_spots(self, col):
16          if 0 <= col < 7 and self.board[0][col] == ' ':
17              for i in range(5, -1, -1):
18                  if self.board[i][col] == ' ':
19                      self.board[i][col] = self.players[self.player]
20                      return True
21          return False
22
23      def check_winner(self):
24          directions = [(1,0),(0,1),(1,1),(1,-1)]
25          for i in range(6):
26              for j in range(7):
27                  if self.board[i][j] != " ":
28                      for d in directions:
29                          x, y = i, j
30                          count = 0
31                          while 0 <= x <= 5 and 0 <= y <= 6 and \
32                                self.board[x][y] == self.board[i][j]:
33                              count += 1
34                              x, y = x + d[0], y + d[1]
35                              if count == 4:
36                                  return True
37          return False
```

```
38
39  game = Connect4()
40
41  while True:
42      game.print_board()
43      col = int(input(f"Player {game.players[game.player]}, choice a row: "))
44      print()
45      if game.get_spots(col):
46          if game.check_winner():
47              game.print_board()
48              print(f"Player {game.players[game.player]} won!")
49              break
50          game.player = (game.player + 1) % 2
51      else:
52          print("Wrong move.Please try again.")
```

11. Write a class called *Poker_hand* that has a field that is a list of *Card* objects. There should be the following self-explanatory methods:

has_royal_flush, has_straight_flush, has_four_of_a_kind,
has_full_house, has_flush, has_straight,
has_three_of_a_kind, has_two_pair, has_pair

There should also be a method called *best* that returns a string indicating what the best hand is that can be made from those cards.

C:\Solutions to tasks\14.7 Exercises\14.7.11.py - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

14.7.11.py

```
1   from random import *
2
3   class Card:
4       def __init__(self, value, suit):
5           self.value = value
6           self.suit = suit
7
8       def __str__(self):
9           names = ["Jack", "Queen", "King", "Ace"]
10          if self.value <= 10:
11              return f"{self.value} {self.suit}"
12          else:
13              return f"{names[self.value - 11]} {self.suit}"
14
15  class Poker_Deck():
16      def __init__(self):
17          self.cards = []
18          for s in ["♣", "♦", "♥", "♠"]:
19              for v in range(2,15):
20                  self.cards.append(Card(v, s))
21
22  class Poker_Hand:
23      def __init__(self, cards):
24          self.cards = cards
25          self.deals = sample(self.cards, 5)
26          self.suits = [self.deals[i].suit for i in range(len(self.deals))]
27          self.values = [self.deals[i].value for i in range(len(self.deals))]
28          self.cards = [self.deals[i].__str__() for i in range(len(self.deals))]
29          self.values.sort()
```

```
            self.values.sort()
            self.cards_count = {i:self.values.count(i) for i in self.values}
            self.value_count = list(self.cards_count.values())
            self.value_count.sort()

        def has_royal_flush(self):
            return self.values == [10,11,12,13,14] and self.has_flush()

        def has_straight_flush(self):
            return self.has_straight() and self.has_flush()

        def has_four_of_a_kind(self):
            return self.value_count == [1, 4]

        def has_full_house(self):
            return self.value_count == [2,3]

        def has_flush(self):
            return self.suits.count(self.suits[0]) == 5

        def has_straight(self):
            return self.values == [self.values[0] + i for i in range(5)]

        def has_three_of_a_kind(self):
            return self.value_count == [1,1,3]

        def has_two_pair(self):
            return self.value_count == [1,2,2]

        def has_pair(self):
            return self.value_count == [1,1,1,2]

        def best(self):
            if self.has_royal_flush():
                return "royal flush"
            elif self.has_straight_flush():
                return "straight flush"
            elif self.has_four_of_a_kind():
                return "four of a kind"
            elif self.has_full_house():
                return "full house"
            elif self.has_flush():
                return "flush"
            elif self.has_straight():
                return "straight"
            elif self.has_three_of_a_kind():
                return "three of a kind"
            elif self.has_two_pair():
                return " two pair"
            elif self.has_pair():
                return "pair"
            else:
                return "high cards"
    print("To continue, press - ENTER, to end, press - 0 ")
    enter = "1"
    while enter != "0":
        deck = Poker_Deck()
        hand = Poker_Hand(deck.cards)
        print(" | ".join(hand.cards) + "|")
        print(hand.best())
        enter = input()
```