

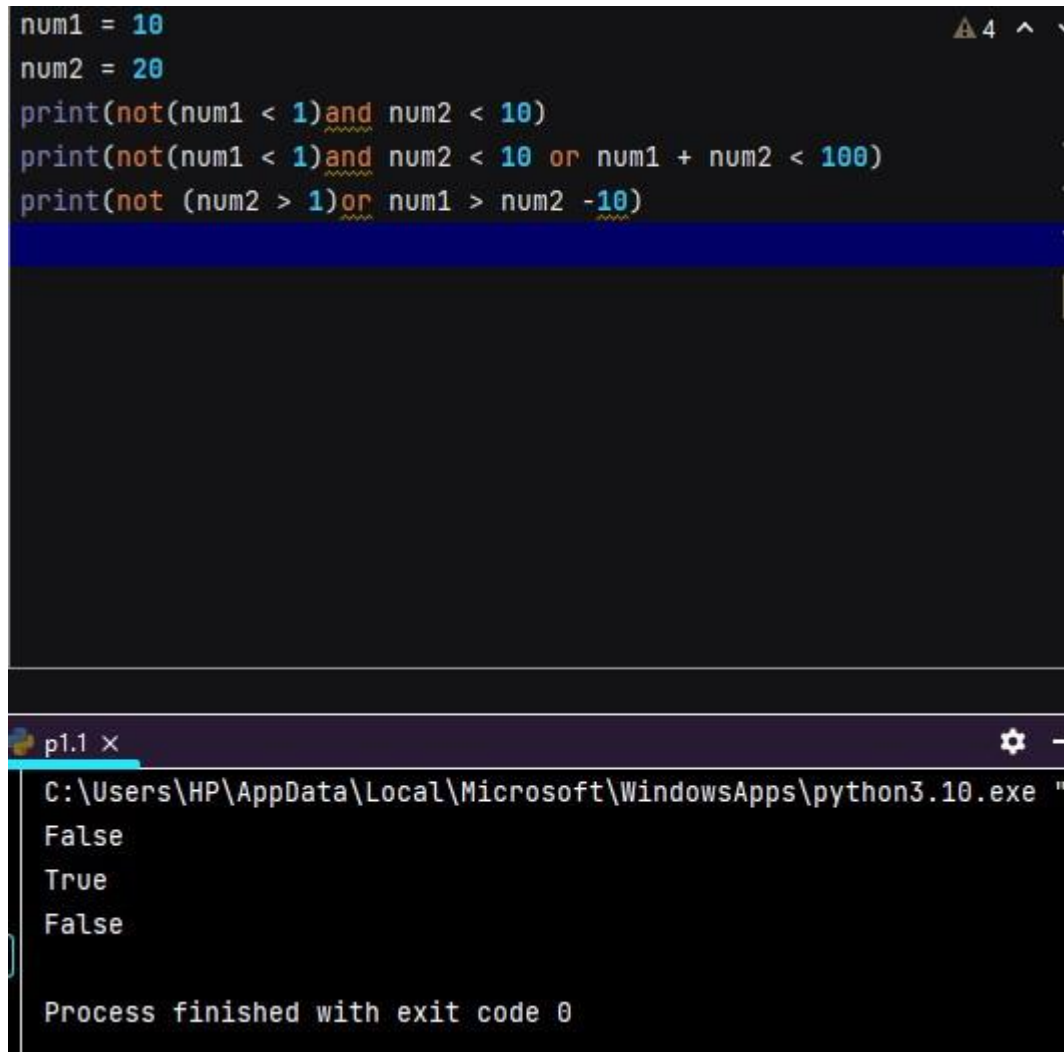
Part 1

1. Evaluate the following expressions for $\text{num1} = 10$ and $\text{num2} = 20$.

(a) $\text{not } (\text{num1} < 1) \text{ and } \text{num2} < 10$

(b) $\text{not } (\text{num1} < 1) \text{ and } \text{num2} < 10 \text{ or } \text{num1} + \text{num2} < 100$

(c) $\text{not } (\text{num2} > 1) \text{ or } \text{num1} > \text{num2} - 10$



```
num1 = 10
num2 = 20
print(not(num1 < 1) and num2 < 10)
print(not(num1 < 1) and num2 < 10 or num1 + num2 < 100)
print(not (num2 > 1) or num1 > num2 - 10)
```

p1.1 x

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "
False
True
False

Process finished with exit code 0
```

2. Give a proper if statement for each of the following (the value of num is not important):

(a) Displays 'within range' if num is between 0 and 100, inclusive.

```
def range():  
    num = int(input("enter any number"))  
    if ( 0 <= num <= 100):  
        print("within range")  
range()
```

p1.2a x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python
enter any number 99
within range

Process finished with exit code 0

(b) Displays 'within range' if num is between 0 and 100, inclusive, and displays 'out of range' otherwise.

```
def range():  
    num=int(input("enter any number"))  
    if (0<=num<=100):  
        print("within range")  
    else:  
        print("out of range")  
range()  
  
p1.2b x  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe  
enter any number 101  
out of range  
  
Process finished with exit code 0  
|
```

3. Rewrite the following if-else statements using a single if statement and elif:

```
if temperature >= 85 and humidity > 60:  
    print ('muggy day today')  
else:  
    if temperature >= 85:  
        print ('warm, but not muggy today')  
    else:  
        if temperature >= 65:  
            print ('pleasant today')  
        else:  
            if temperature <= 45:  
                print ('cold today')  
            else:  
                print ('cool today')
```

```
def temp_humi():  
    temperature=int(input("enter the temperature: "))  
    humidity=int(input("enter the humidity: "))  
    if (temperature >= 85 and humidity > 60 ):  
        print('muggy day today')  
    elif (temperature >= 85):  
        print('warm, but not muggy today')  
    elif(temperature >= 65):  
        print('pleasant today')  
    elif(temperature <= 45):  
        print('cold today')  
    else:  
        print('cool today')  
temp_humi()
```

p1.3 x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.e
enter the temperature: 27
enter the humidity: 56
cold today

Process finished with exit code 0

4. Write a Python program in which:

(a) The user enters either 'A', 'B', or 'C'. If 'A' is entered, the program should display the word 'Apple'; if 'B' is entered, it displays 'Banana'; and if 'C' is entered, it displays 'Coconut'. Use nested if statements for this.

```
def fruits():  
    fruit=input("enter either 'A', 'B', 'C': ")  
    if fruit == 'A':  
        print("Apple")  
    else:  
        if fruit == 'B':  
            print("Banana")  
        else:  
            if fruit == 'C':  
                print("Coconut")  
            else:  
                print("Invalid Input.")  
fruits()  
|  
  
p1.4a x  
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.e  
enter either 'A', 'B', 'C': N  
Invalid Input.  
  
Process finished with exit code 0
```

(b) Repeat question (a) using an if statement with `elif` headers instead.

```
def fruits():  
    fruit=input("enter either 'A', 'B', 'C': ")  
    if fruit == 'A':  
        print("Apple")  
    elif (fruit == 'B'):  
        print("Banana")  
    elif (fruit == 'C'):  
        print("Coconut")  
    else:  
        print("Invalid input")  
fruits()
```

p1.4b x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe
enter either 'A', 'B', 'C': C
Coconut
Process finished with exit code 0

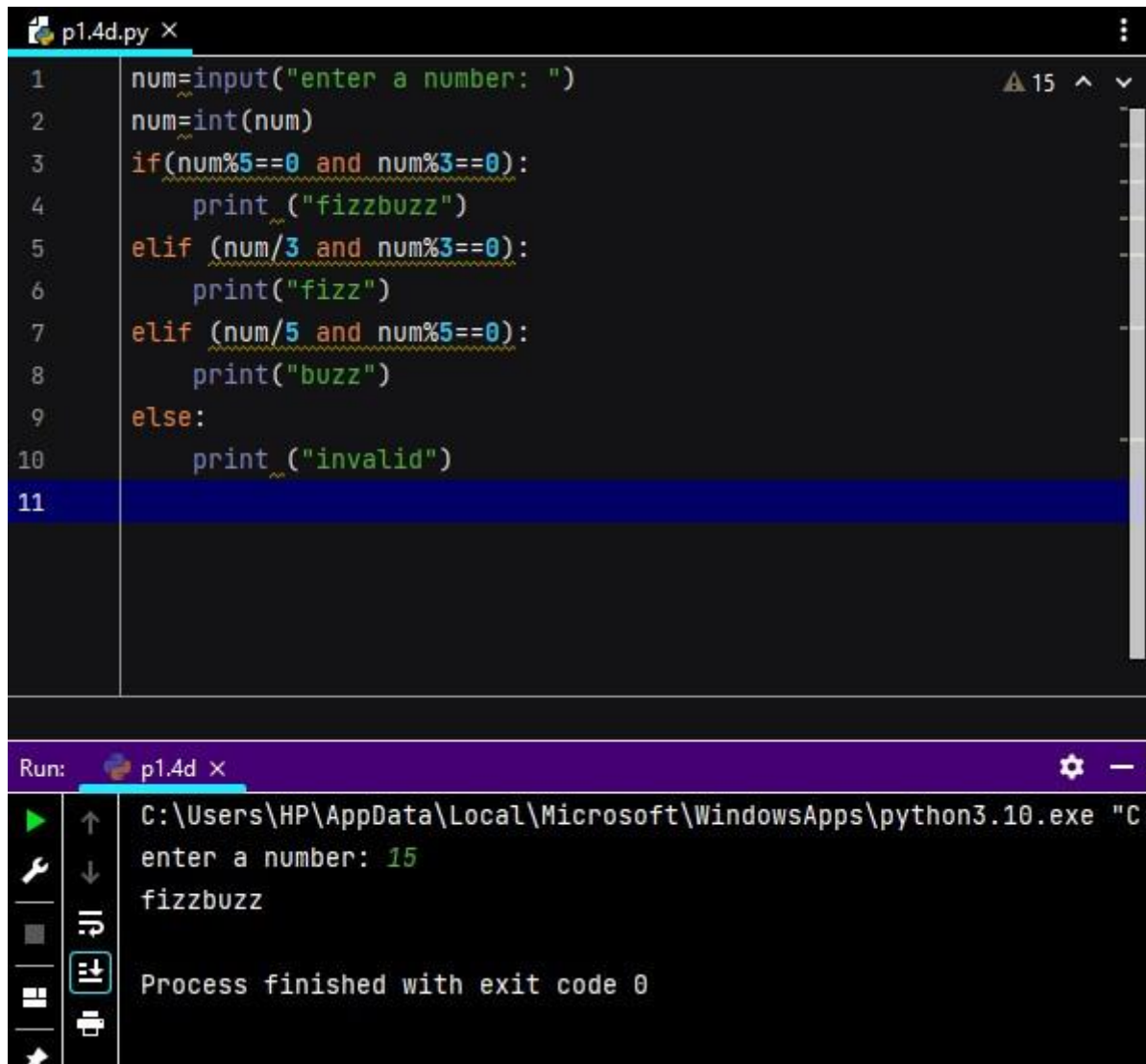
(c) A student enters the number of college credits earned. If the number of credits is greater than or equal to 90, 'Senior Status' is displayed; if greater than or equal to 60, 'Junior Status' is displayed; if greater than or equal to 30, 'Sophomore Status' is displayed; else, 'Freshman Status' is displayed.

```
p1.4c.py ×
1 credit = input("enter the number of college credits ")
2 credit = int(credit)
3 if credit >=90:
4     print("Senior status")
5 elif credit >=60 and credit <90:
6     print("junior status")
7 elif credit >=30 and credit <60:
8     print("Sophmore status")
9 else:
10    print("freshman status")
11

Run: p1.4c ×
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C
enter the number of college credits you earned: 60
junior status

Process finished with exit code 0
```

(d) The user enters a number. If the number is divisible by 3, the word 'Fizz' should be displayed; if the number is divisible by 5 the word 'Buzz' should be displayed and if the number is divisible by both 'FizzBuzz' should be displayed.

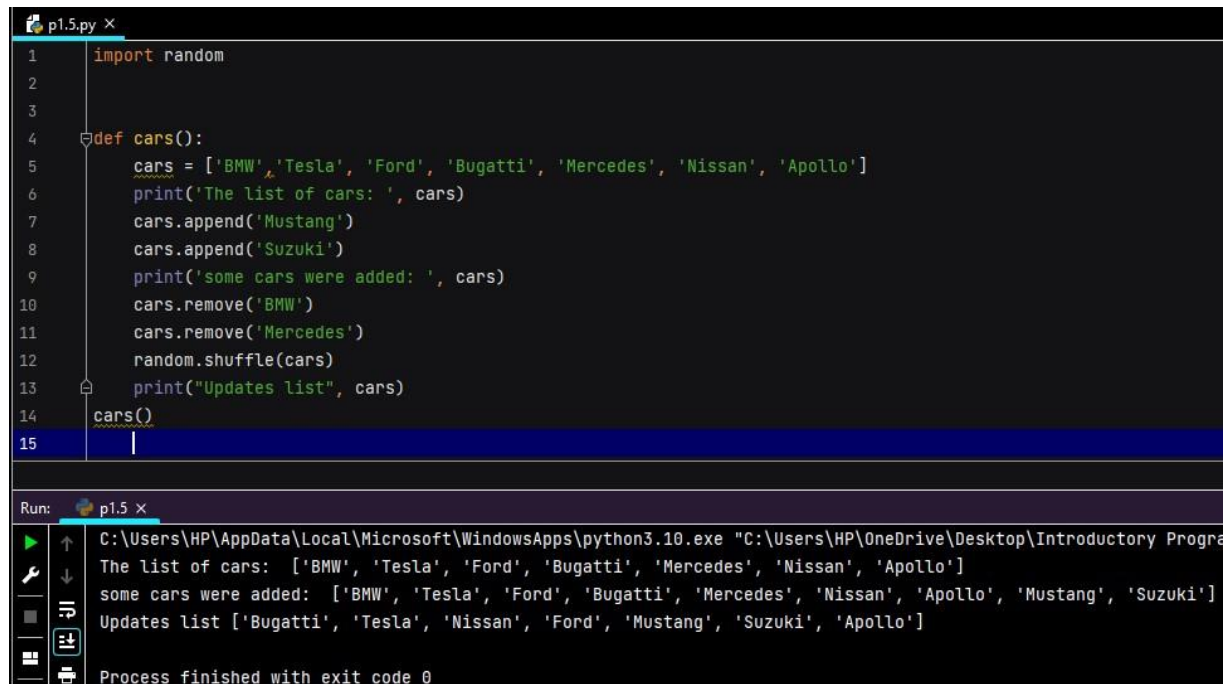


The screenshot shows a Python IDE with a file named `p1.4d.py`. The code is a FizzBuzz program that takes an input number and prints "fizzbuzz", "fizz", "buzz", or "invalid" based on divisibility rules. The code is as follows:

```
1 num=input("enter a number: ")
2 num=int(num)
3 if(num%5==0 and num%3==0):
4     print("fizzbuzz")
5 elif (num/3 and num%3==0):
6     print("fizz")
7 elif (num/5 and num%5==0):
8     print("buzz")
9 else:
10    print("invalid")
11
```

Below the code editor is a terminal window titled "Run: p1.4d x". It shows the execution of the program with the input "15" and the output "fizzbuzz". The terminal also indicates that the process finished with exit code 0.

5. Sam wants to store his series of car to a list. The list of a car are: (up to you). After creating a list he add some car and delete some car and at last there are still 5 cars left in his list. Additionally, he wants his car to be shuffled every time when the list is being displayed. [Hint: shuffle from random]



```
p1.5.py X
1  import random
2
3
4  def cars():
5      cars = ['BMW', 'Tesla', 'Ford', 'Bugatti', 'Mercedes', 'Nissan', 'Apollo']
6      print('The list of cars: ', cars)
7      cars.append('Mustang')
8      cars.append('Suzuki')
9      print('some cars were added: ', cars)
10     cars.remove('BMW')
11     cars.remove('Mercedes')
12     random.shuffle(cars)
13     print("Updates list", cars)
14     cars()
15
```

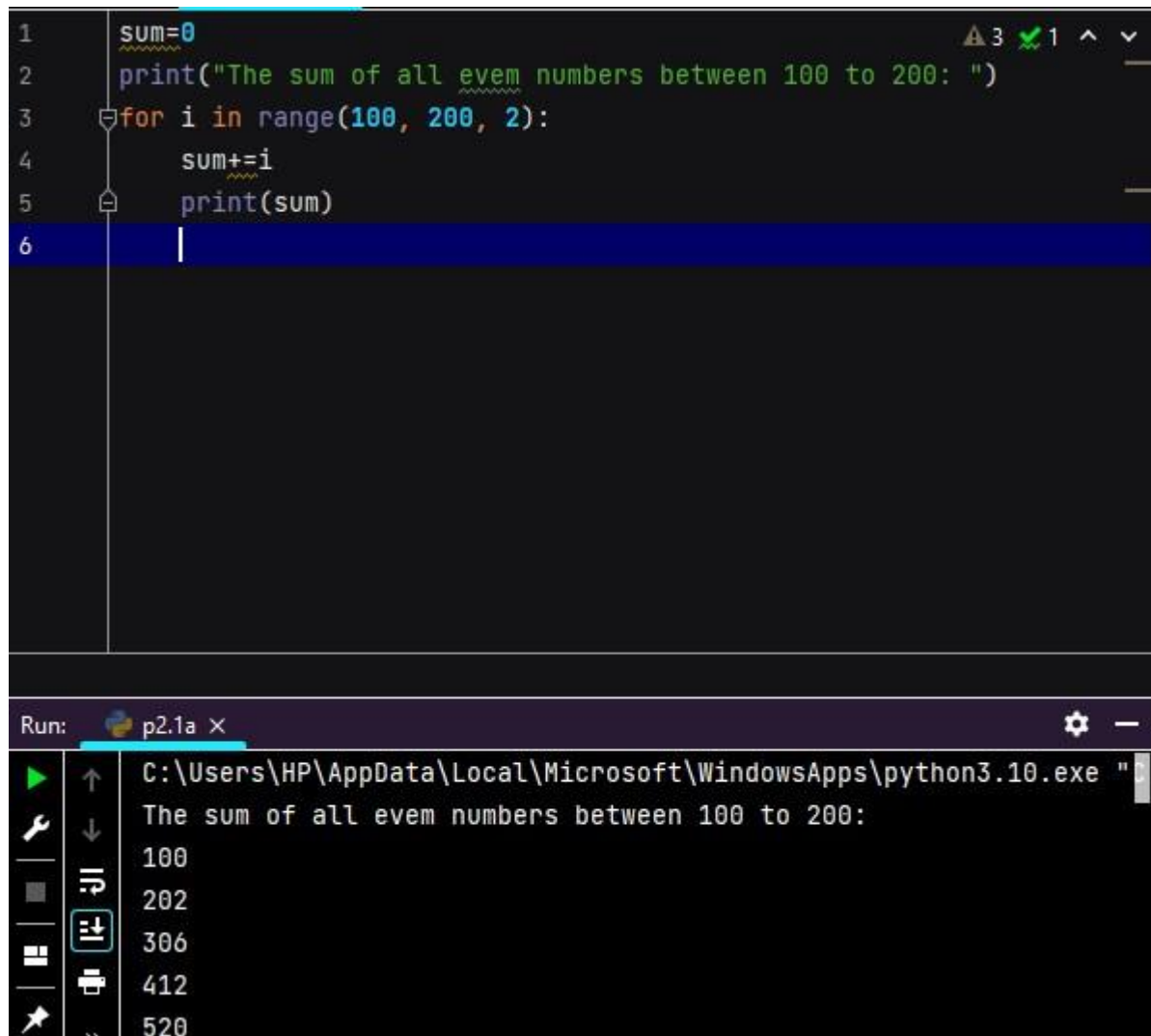
Run: p1.5 X

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneDrive\Desktop\Introductory Program"
The list of cars: ['BMW', 'Tesla', 'Ford', 'Bugatti', 'Mercedes', 'Nissan', 'Apollo']
some cars were added: ['BMW', 'Tesla', 'Ford', 'Bugatti', 'Mercedes', 'Nissan', 'Apollo', 'Mustang', 'Suzuki']
Updates list ['Bugatti', 'Tesla', 'Nissan', 'Ford', 'Mustang', 'Suzuki', 'Apollo']
Process finished with exit code 0
```

Part 2

1. Write a program that:

- (a) Uses a loop to add up all the even numbers between 100 and 200, inclusive.



```
1  sum=0
2  print("The sum of all even numbers between 100 to 200: ")
3  for i in range(100, 200, 2):
4      sum+=i
5      print(sum)
6  |
```

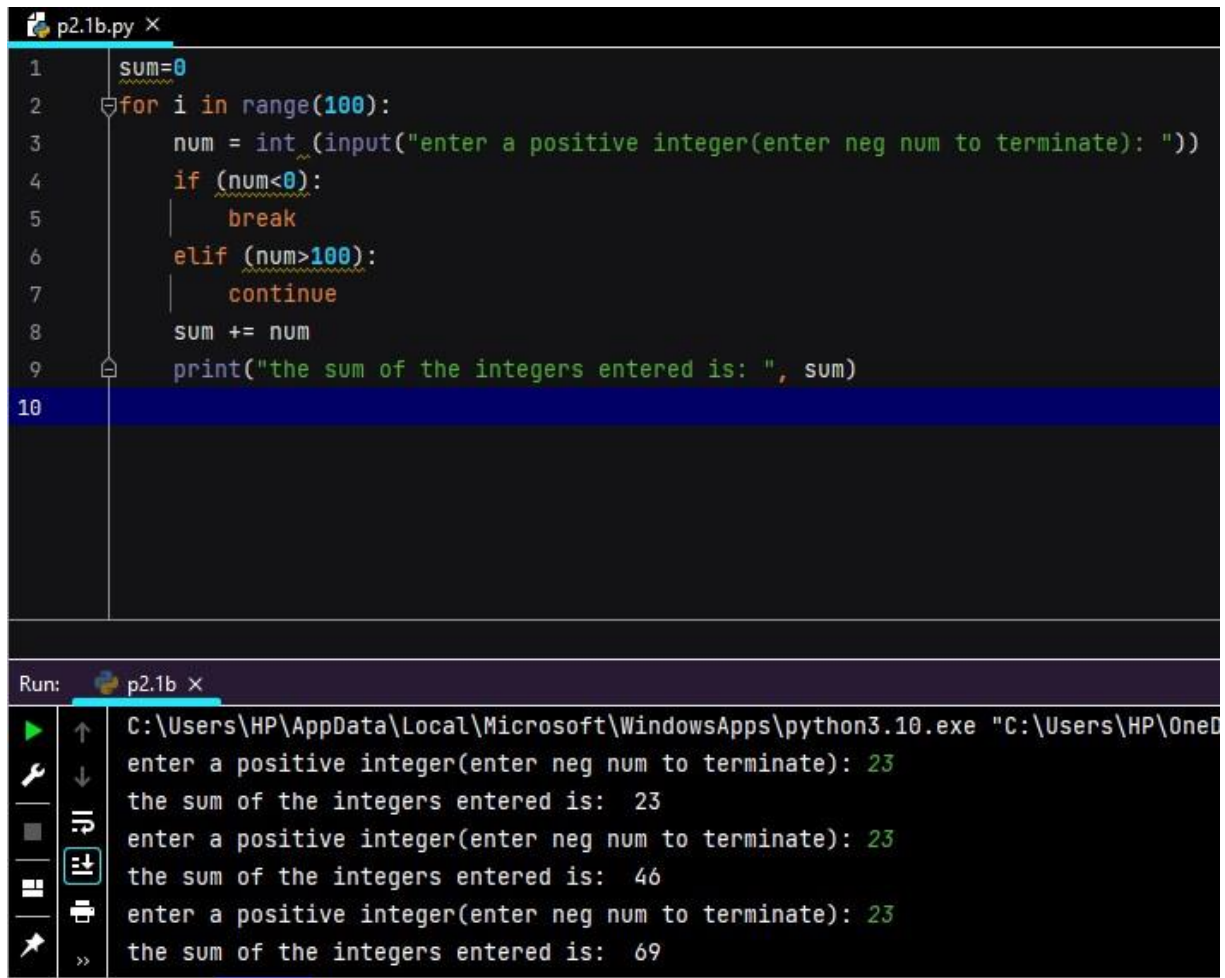
Run: p2.1a x

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "

The sum of all even numbers between 100 to 200:

100
202
306
412
520

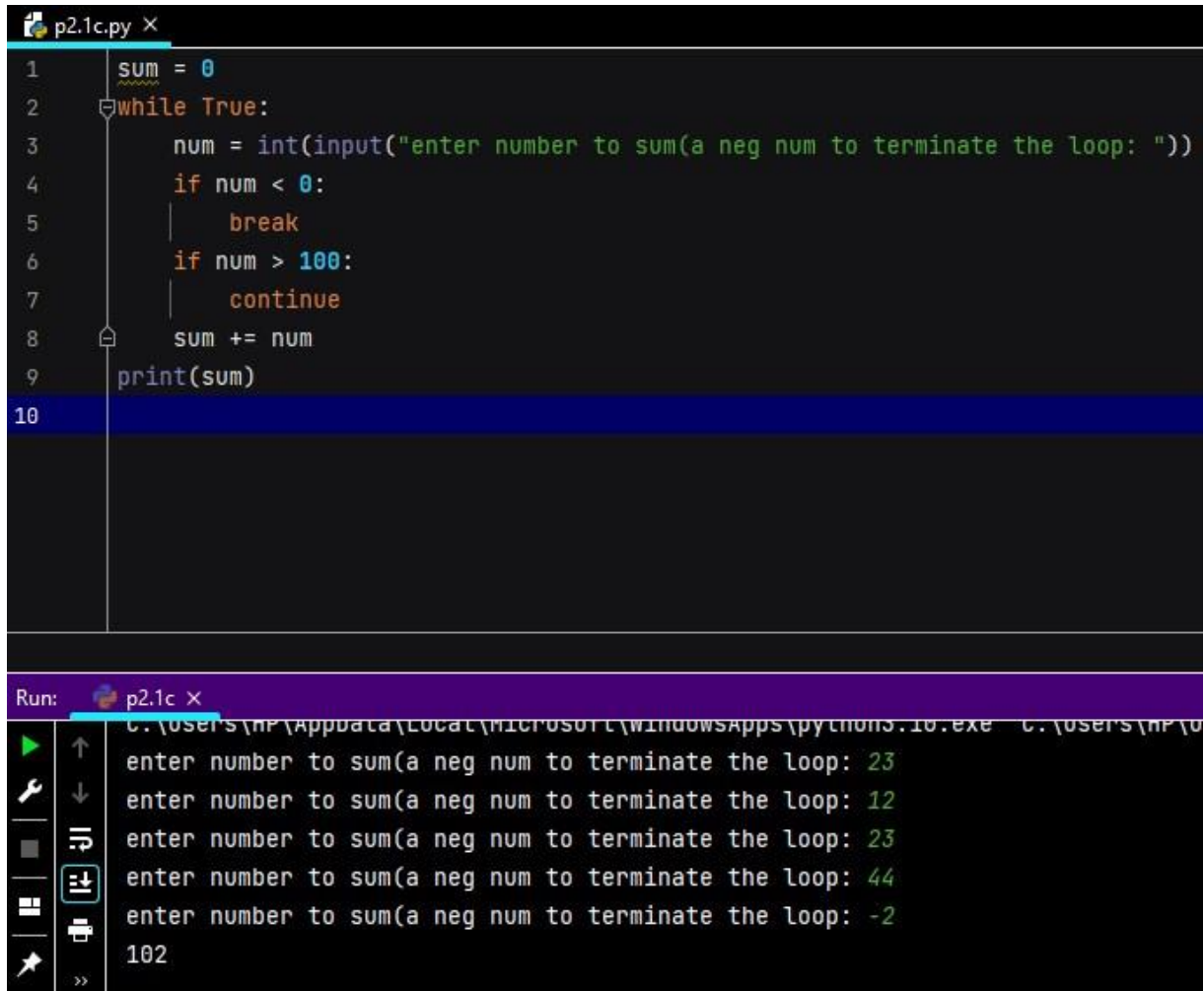
(b) Sums a series of (positive) integers entered by the user, excluding all numbers that are greater than 100.



```
p2.1b.py X
1  sum=0
2  for i in range(100):
3      num = int(input("enter a positive integer(enter neg num to terminate): "))
4      if (num<0):
5          break
6      elif (num>100):
7          continue
8      sum += num
9  print("the sum of the integers entered is: ", sum)
10

Run: p2.1b X
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneD
enter a positive integer(enter neg num to terminate): 23
the sum of the integers entered is: 23
enter a positive integer(enter neg num to terminate): 23
the sum of the integers entered is: 46
enter a positive integer(enter neg num to terminate): 23
the sum of the integers entered is: 69
```

(c) Solves Q2 but this time using an infinite loop, break and continue statements.



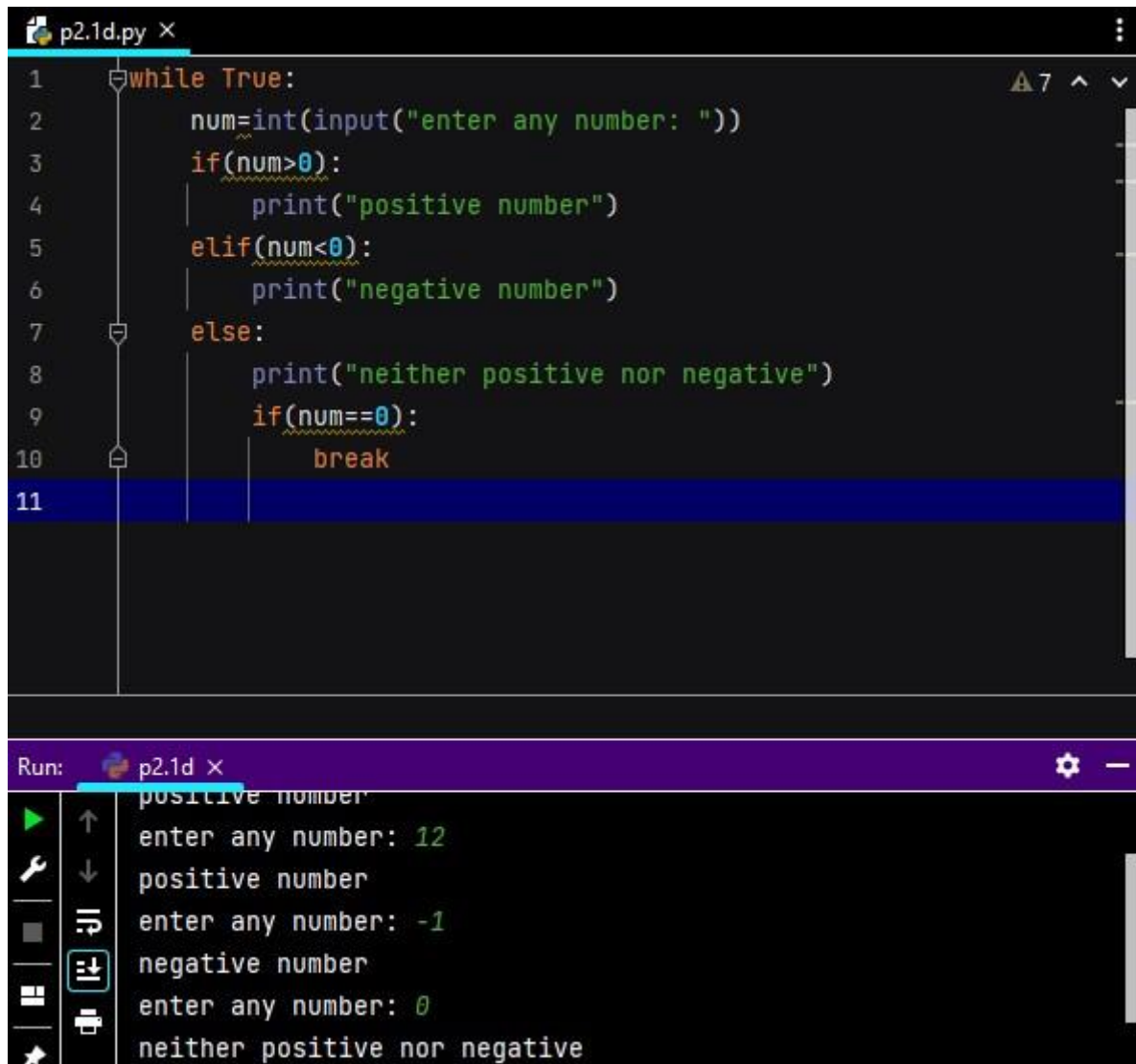
The image shows a Python IDE with a file named `p2.1c.py`. The code is as follows:

```
1 sum = 0
2 while True:
3     num = int(input("enter number to sum(a neg num to terminate the loop: "))
4     if num < 0:
5         break
6     if num > 100:
7         continue
8     sum += num
9 print(sum)
```

Line 10 is highlighted in blue. Below the code editor is a terminal window titled "Run: p2.1c X" showing the program's execution:

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe C:\Users\HP\p2.1c.py
enter number to sum(a neg num to terminate the loop: 23
enter number to sum(a neg num to terminate the loop: 12
enter number to sum(a neg num to terminate the loop: 23
enter number to sum(a neg num to terminate the loop: 44
enter number to sum(a neg num to terminate the loop: -2
102
```

(d) Prompts the user to enter any number of positive and negative integer values, then displays the number of each type that were entered.



The screenshot shows a Python IDE with a file named `p2.1d.py`. The code is as follows:

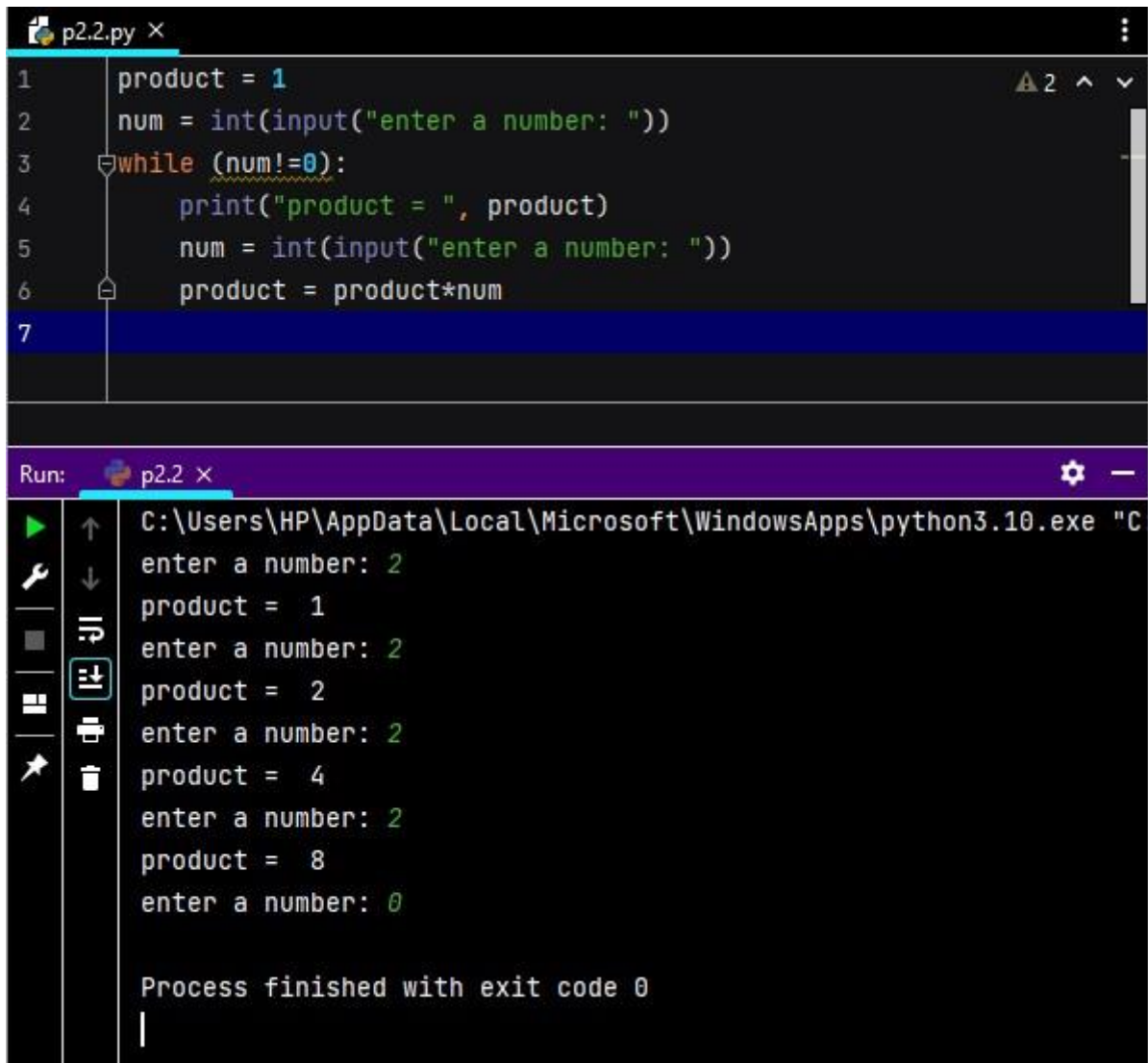
```
1 while True:
2     num=int(input("enter any number: "))
3     if(num>0):
4         print("positive number")
5     elif(num<0):
6         print("negative number")
7     else:
8         print("neither positive nor negative")
9         if(num==0):
10            break
11
```

Below the code editor is a "Run" window showing the output of the program. The output is as follows:

```
Run: p2.1d x
positive number
enter any number: 12
positive number
enter any number: -1
negative number
enter any number: 0
neither positive nor negative
```

2. The following while loop is meant to multiply a series of integers input by the user, until a sentinel value of 0 is entered. Indicate any errors in the code given. See if you can fix the program and get it running.

```
product = 1
num = input('Enter first number: ')
while num != 0:
    num = input('Enter first number: ')
    product = product * num
    print('product = ', product)
```



The screenshot shows a Python IDE with a file named `p2.2.py`. The code is as follows:

```
1 product = 1
2 num = int(input("enter a number: "))
3 while (num!=0):
4     print("product = ", product)
5     num = int(input("enter a number: "))
6     product = product*num
7
```

Below the code editor is a 'Run' window showing the execution of the program. The output is:

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe"
enter a number: 2
product = 1
enter a number: 2
product = 2
enter a number: 2
product = 4
enter a number: 2
product = 8
enter a number: 0
Process finished with exit code 0
```

3. For each of the following, indicate which the definite loop is, and which an indefinite loop, explain your reasoning. (a)

```
num = input('Enter a non-zero value:')
while num == 0:
    num = input('Enter a non-zero value: ')
```

= In the above example, we can see an indefinite loop which is the program loop. The loop will iterate number of times which also mean that cannot be figured out before the loop is executed.

b)

```
num = 0
while n < 10:
    print 2 ** n
    n = n + 1
```

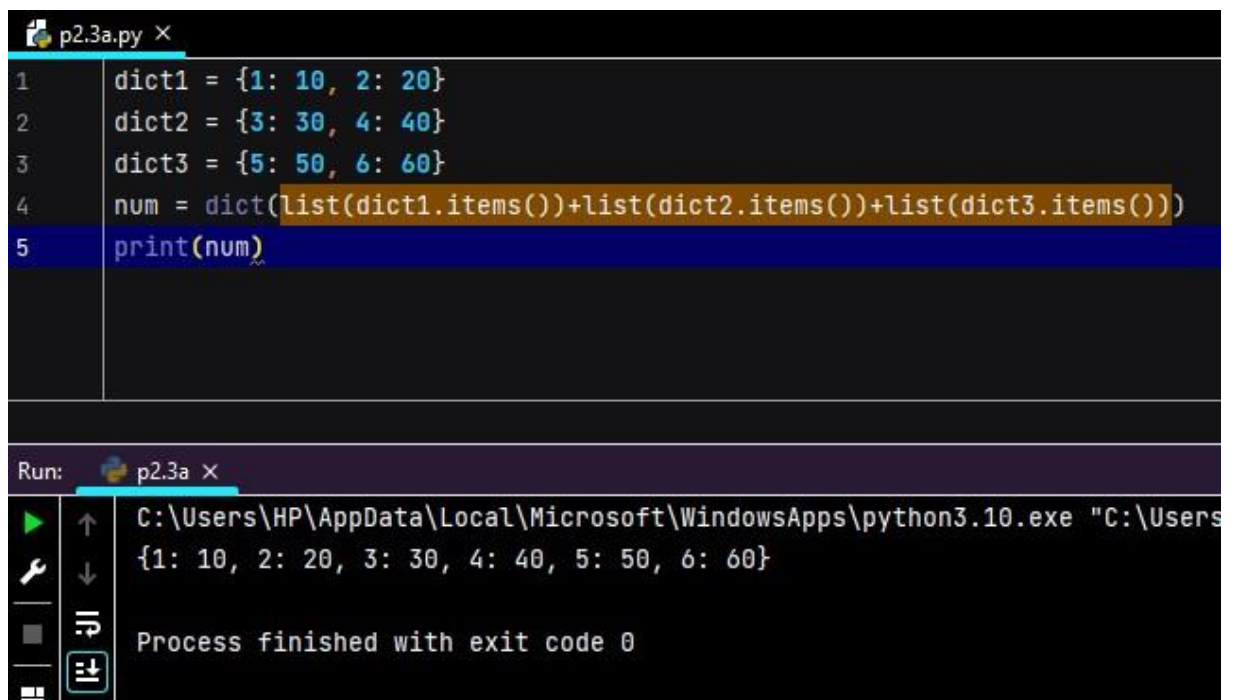
=

Part 3

1. Create three dictionaries:

```
dic1 = {1:10, 2:20}
dic2 = {3:30, 4:40}
dic3 = {5:50, 6:60}
```

- (a) Write code to concatenate these dictionaries to create a new one. Create a variable called `nums` to store the resulting dictionary. There are multiple ways to do this, however, one of the easiest is to convert each of the dictionaries items to a list (which can be added together) and pass them to the `dict()` constructor.

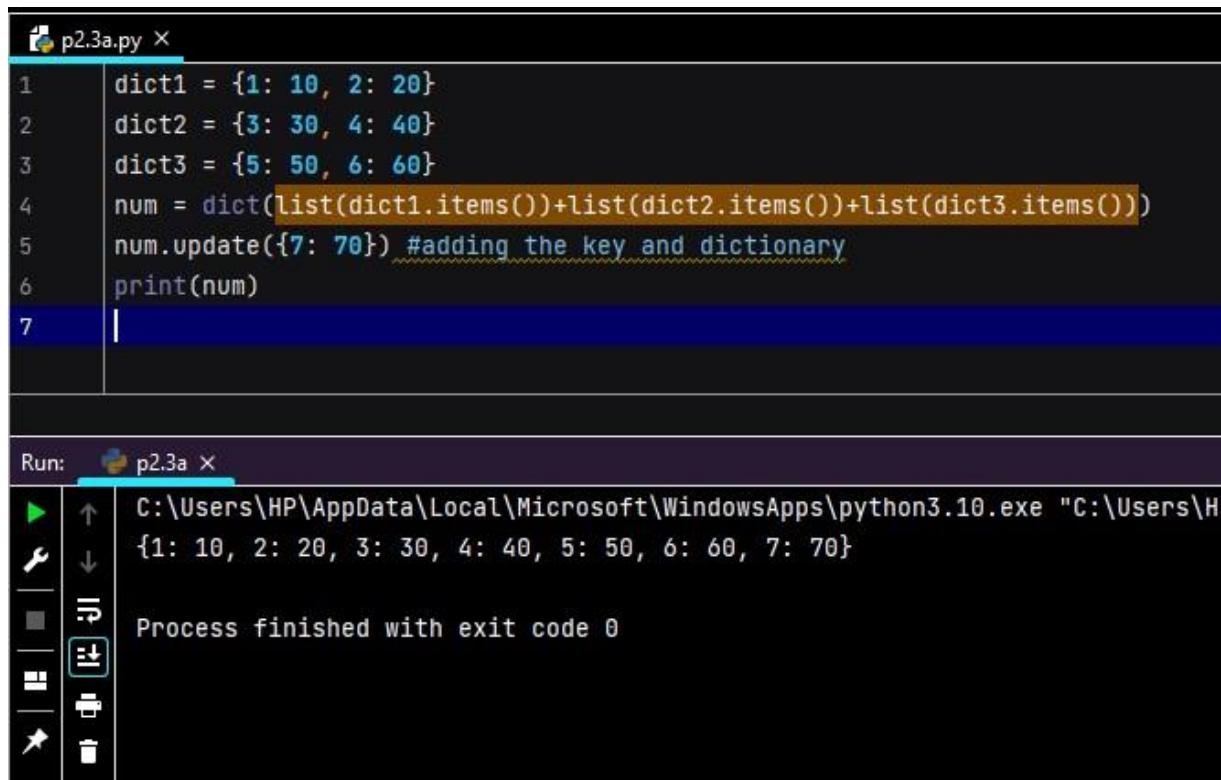


```
p2.3a.py ×
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items())+list(dict2.items())+list(dict3.items()))
5 print(num)

Run: p2.3a ×
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

Process finished with exit code 0
```

- (b) Write code to add a new key/value pair to the dictionary `nums`: `(7, 70)`



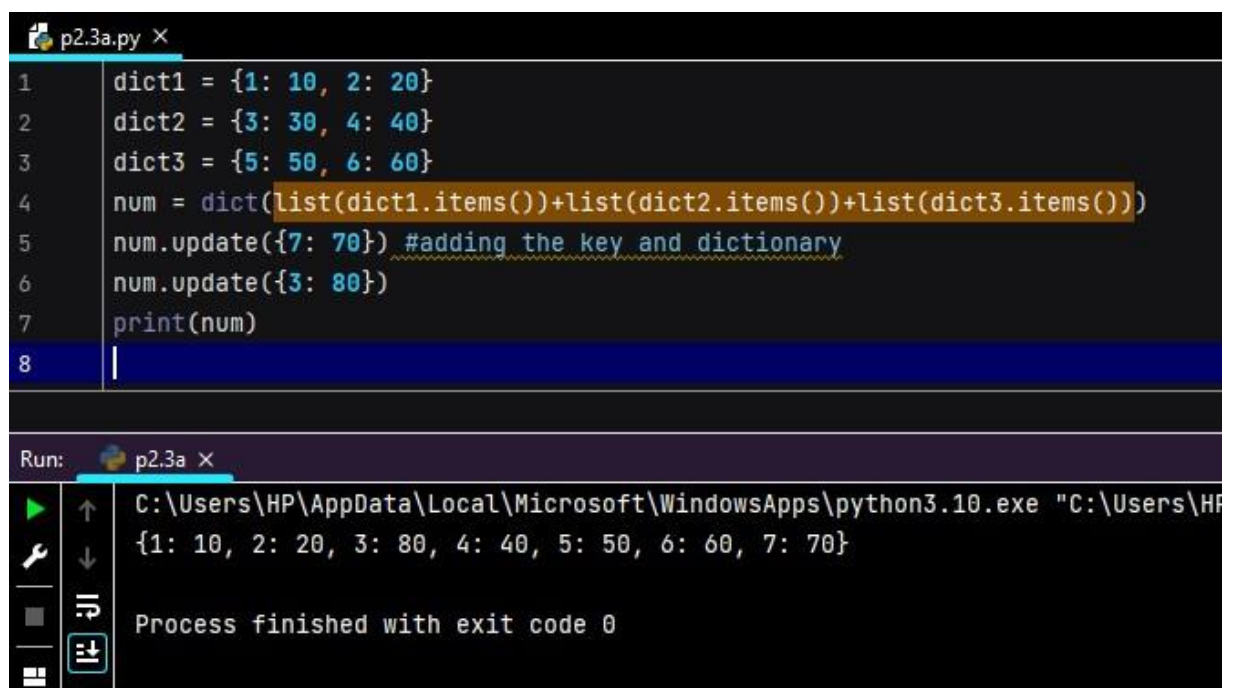
```
p2.3a.py X
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items())+list(dict2.items())+list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 print(num)
7
```

Run: p2.3a X

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\H
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60, 7: 70}

Process finished with exit code 0
```

(c) Write code to update the value of the item with key 3 in nums to 80



```
p2.3a.py X
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items())+list(dict2.items())+list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 num.update({3: 80})
7 print(num)
8
```

Run: p2.3a X

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\H
{1: 10, 2: 20, 3: 80, 4: 40, 5: 50, 6: 60, 7: 70}

Process finished with exit code 0
```


(d) Write code to remove the third item from dictionary `nums`.

```
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items()) + list(dict2.items()) + list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 del num[3]
7 print(num)
8
```

Run: p2.3a ×

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe" "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe" {1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}

Process finished with exit code 0

(e) Write code to sum all the items in the dictionary `nums`

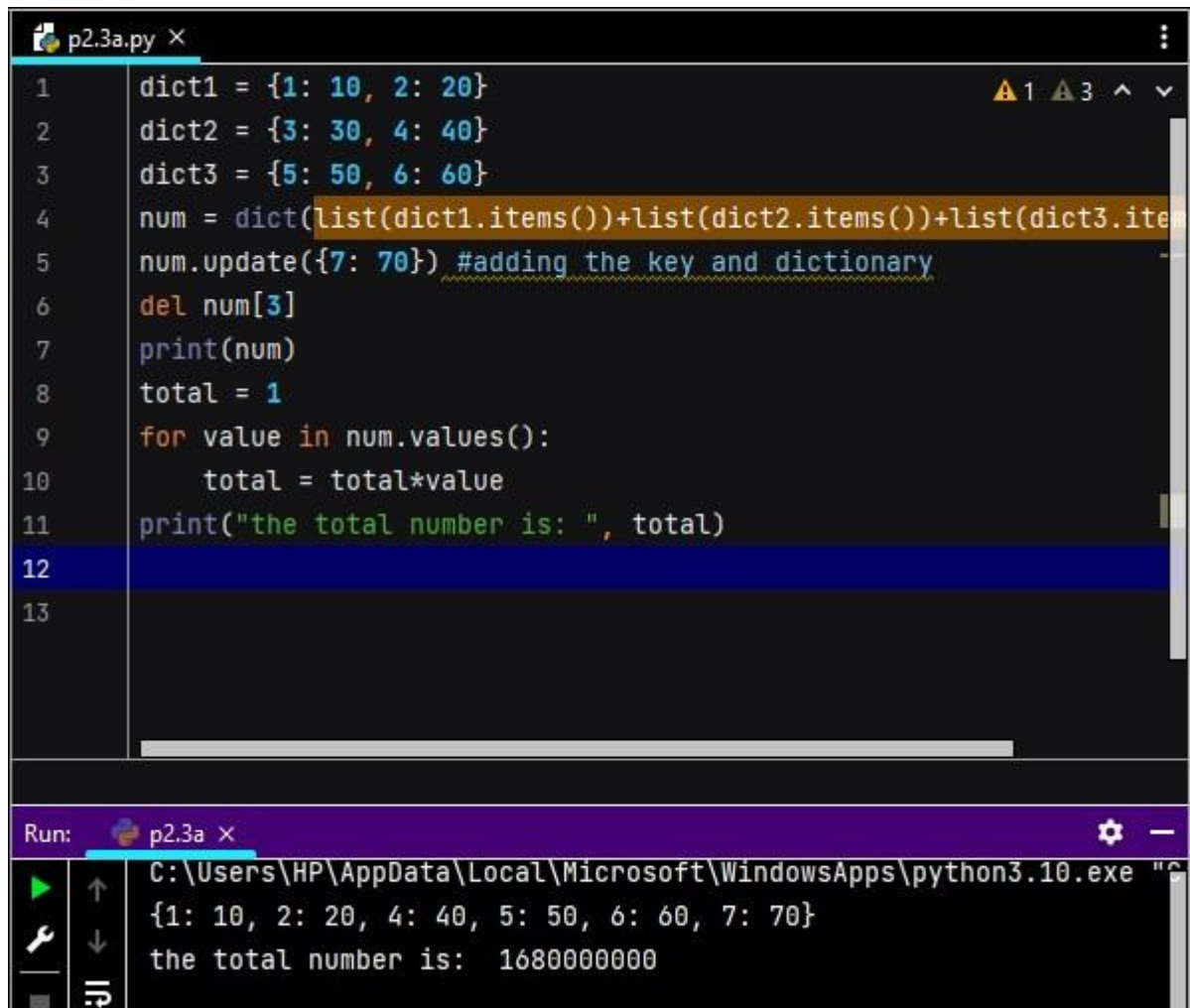
```
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items()) + list(dict2.items()) + list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 del num[3]
7 print(num)
8 total = sum(num.values())
9 print("the total number is: ", total)
10
```

Run: p2.3a ×

C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe" "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe" {1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}

Process finished with exit code 0

(f) Write code to multiply all the items in the dictionary `nums`



```
p2.3a.py X
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items()) + list(dict2.items()) + list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 del num[3]
7 print(num)
8 total = 1
9 for value in num.values():
10     total = total * value
11 print("the total number is: ", total)
12
13
```

Run: p2.3a X

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe"
{1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}
the total number is: 16800000000
```

(g) Write code to retrieve the maximum and minimum values in `nums`.

```
p2.3a.py x
1 dict1 = {1: 10, 2: 20}
2 dict2 = {3: 30, 4: 40}
3 dict3 = {5: 50, 6: 60}
4 num = dict(list(dict1.items())+list(dict2.items())+list(dict3.items()))
5 num.update({7: 70}) #adding the key and dictionary
6 del num[3]
7 print(num)
8 max_value = max(num.values())
9 min_value = min(num.values())
10 print("the maximum value is: ", max_value)
11 print("the minimum value is: ", min_value)
12
13
```

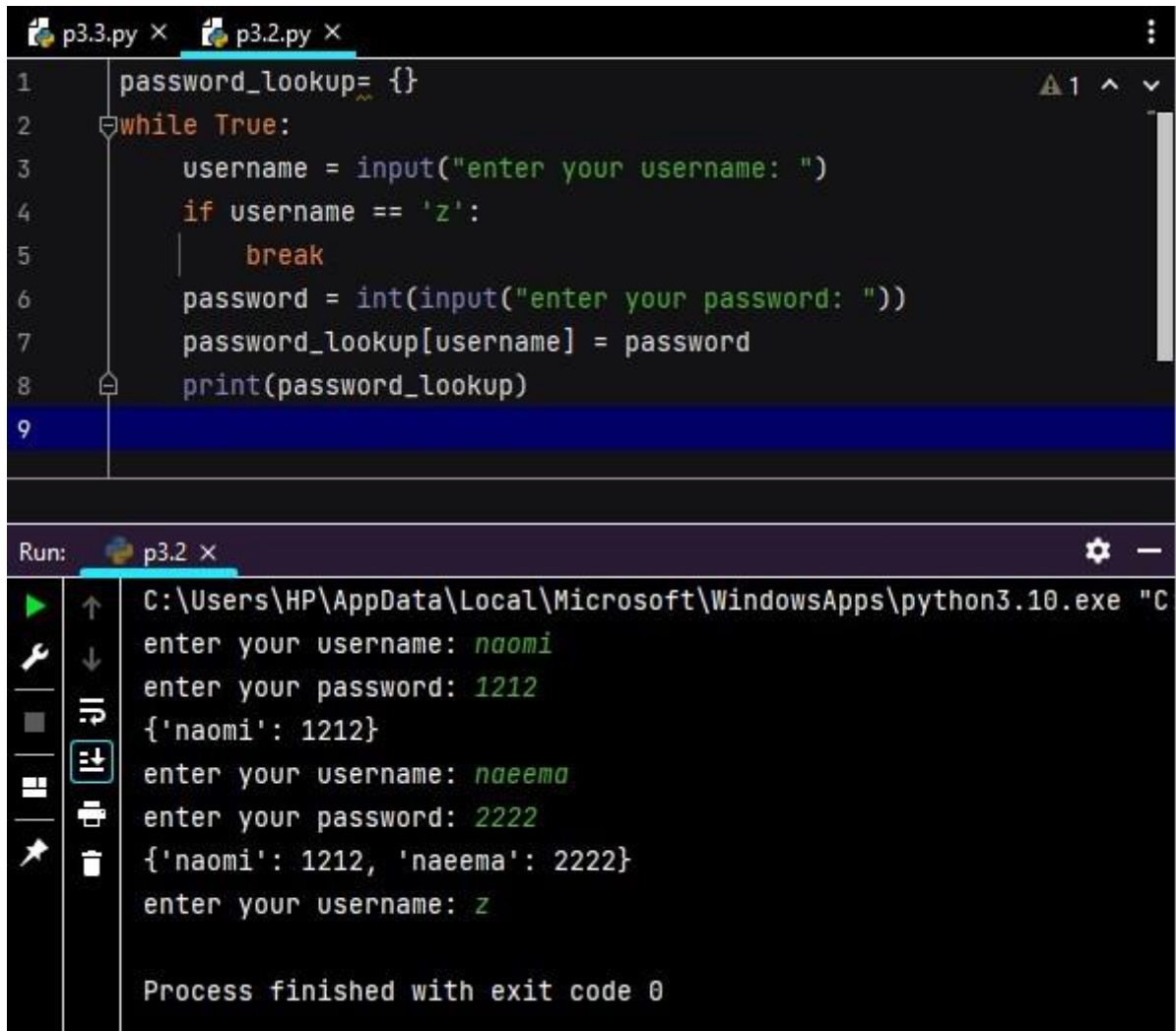
```
Run: p2.3a x
{1: 10, 2: 20, 4: 40, 5: 50, 6: 60, 7: 70}
the maximum value is: 70
the minimum value is: 10
Process finished with exit code 0
```

2. Create a dictionary named `password_lookup` that contains usernames as keys and passwords as associated string values. Make up data for five entries.

```
p3.3.py x
1 #creating dictionary key containing username and value containing password
2 password_lookup={"ram": 2222, "rekha": 4444, "naomi": 1111, "naeema": 3333, "ravi": 4565}
3 print(password_lookup)
4
```

```
Run: p3.3 x
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\Users\HP\OneDrive\Desktop"
{'ram': 2222, 'rekha': 4444, 'naomi': 1111, 'naeema': 3333, 'ravi': 4565}
Process finished with exit code 0
```

3. Write a program that creates an initially empty dictionary named `password_lookup`, prompting one-by-one for usernames and passwords (until a username of 'z' is read) entering each into the dictionary.



```
1 password_lookup= {}
2 while True:
3     username = input("enter your username: ")
4     if username == 'z':
5         break
6     password = int(input("enter your password: "))
7     password_lookup[username] = password
8     print(password_lookup)
9
```

Run: p3.2 x

```
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C
enter your username: naomi
enter your password: 1212
{'naomi': 1212}
enter your username: naeema
enter your password: 2222
{'naomi': 1212, 'naeema': 2222}
enter your username: z

Process finished with exit code 0
```

4. Create a dictionary named `password_hint` that contains email addresses as keys, and associated values that contain both the users' "password security question," and the answer to the question. Make up data for dictionary entries.

```
p3.4.py x
1 password_hint={"gmail.com": 2222}
2 username = input("what is the email address?")
3 if username == "gmail.com":
4     print("Welcome.")
5 else:
6     print("invalid!!!")
7     password=int(input("enter the password: "))
8     if (password == 2222):
9         print("correct password")
10    else:
11        print("incorrect password")
12        if(username == "gmail.com" and password == 2222):
13            password_hint[username]=password
14            print(password_hint)
15        else:
16            print("5 mins till you can try again")
17
18
```

```
p3.4 x
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C:\U
what is the email address?thing@gmail.com
invalid!!!
enter the password: 1222
incorrect password
5 mins till you can try again

Process finished with exit code 0
```

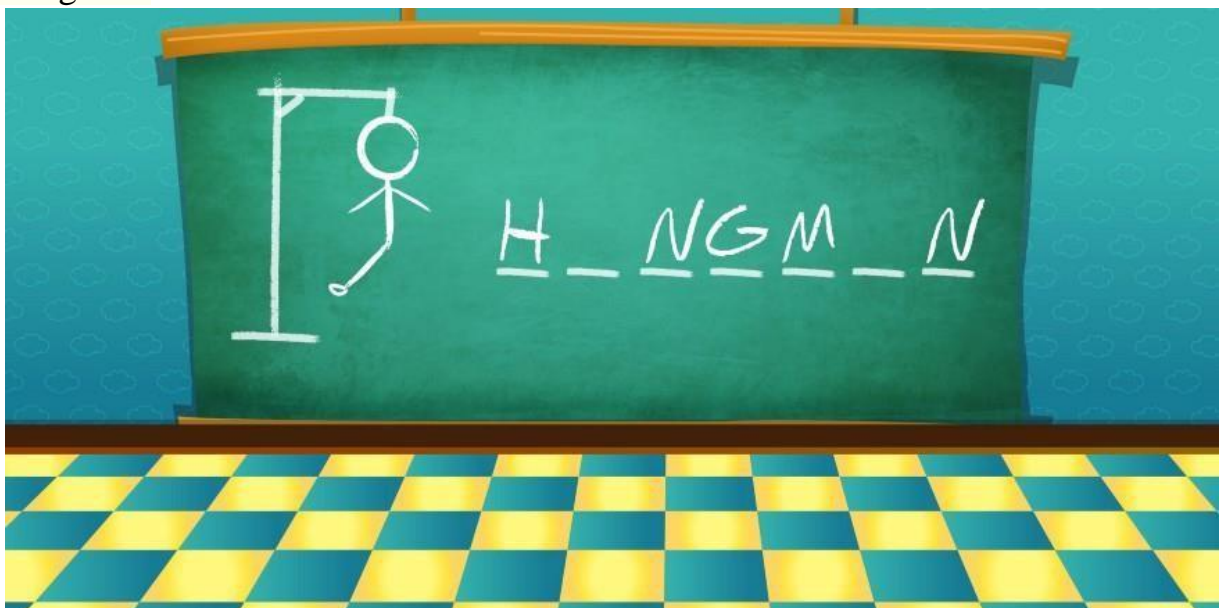
5. Create a dictionary named `member_table` that contains users' email addresses as keys, and answers to their password hints as the associated values, and a function that generates a temporary new password and stored in the table.


```
p3.5.py ×
1 member_table = {}
2 username = input("enter your email address or username: ")
3 password = int(input("enter your password: "))
4 member_table.update({username: password})
5 print(member_table)
6

Run: p3.5 ×
C:\Users\HP\AppData\Local\Microsoft\WindowsApps\python3.10.exe "C
enter your email address or username: naomithing
enter your password: 2222
{'naomithing': 2222}
Process finished with exit code 0
```

Part 4 (Home Task)

1. The hangman game introduces many new concepts like *methods*, which are functions attached to values. You'll also need to learn about a data type called a *list*. Once you understand these concepts, it will be much easier to program Hangman.



Naomi Thing
Group 4
Workshop 9

1. You will need *random* module.
2. You will need to use the concept of *list*.

```

1  import random
2  def hangman():
3      random_word = random.choice(['Python', 'Java', 'Command', 'Academics', 'Naomi'])
4      random_word = random_word.upper()
5      word_letters = set(random_word)
6      alphabet = set('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
7      use_letters = set()
8      tries = 5
9      while len(word_letters) > 0 and tries > 0:
10         print("you have", tries, "tries left")
11         print("letters used", ''.join(use_letters))
12         print("word: ", ''.join(letter if letter in use_letters else '_' for letter in random_word))
13         guess = input("guess a letter: ").upper()
14         if guess in alphabet - use_letters:
15             use_letters.add(guess)
16             if guess in word_letters:
17                 word_letters.remove(guess)
18             else:
19                 tries -= 1
20         else:
21             print("you have already tried this letter.")
22             if tries == 0:
23                 print("you lost your game! the word was: ", random_word)
24             else:
25                 print("you won the game!! the word was: ", random_word)
26     hangman()

```

Run: 1.1 x 1.1 x

```

you have 5 tries left
letters used
word:  _ _ _ _ _
guess a letter: P
you have 4 tries left
letters used P
word:  _ _ _ _ _
guess a letter: Y
you have 3 tries left
letters used YP
word:  _ _ _ _ _
guess a letter: C
you have 3 tries left
letters used YPC
word:  _C_ _ _C_

```