# Exercise: Basic Syntax, Conditional Statements, and Loops

Problems for exercise and homework for the [Python Fundamentals Course @SoftUni](https://softuni.bg/trainings/3953/programming-fundamentals-with-python-january-2023).

Submit your solutions in the SoftUni judge system at <https://judge.softuni.org/Contests/1719>.

## Jenny's Secret Message

Jenny studies programming with Python and wants to create a program that **greets the user** when he/she gives his/her **name**.The greeting should be in the format **"Hello, {name}!**". However, Jenny is in love with **Johnny** and would like to **greet him differently: "Hello, my love!"**. Could you help her?

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Peter | Hello, Peter! |
| Amy | Hello, Amy! |
| Johnny | Hello, my love! |

name= input()  
if name =='Johnny':  
 print(f"Hello, my love!")  
else:  
 print(f"Hello, {name}!")

## Drink Something

**Kids** drink **toddy**, **teens** drink **coke**, **young adults** drink **beer**, and **adults** drink **whisky**. Create a program that receives a **person's age** and prints what he/she drinks.

**Rules:**

**A kid** is defined as someone **under or at** **the age of** **14**.

**A teen** is defined as someone **under** **or at** **the age of 18**.

**A young adult** is defined as someone **under or at** **the age of 21**.

**An adult** is defined as someone **above the age of 21**.

***Note:*** All the values are **inclusive** except the **last** **one**!

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 13 | drink toddy |
| 17 | drink coke |
| 21 | drink beer |
| 30 | drink whisky |

age = int(input())  
  
if age <= 14:  
 print(f'drink toddy')  
elif age <=18:  
 print(f'drink coke')  
elif age <= 21:  
 print(f'drink beer')  
else:  
 print(f'drink whisky')

## Chat Codes

Peter is a programming enthusiast who wants to create a chat where people will send messages via number codes. He starts by creating a program for only **four** messages.

Create a program that receives the **n** number of messages sent. On the following **n** lines, it will receive **integer** numbers. For **each number**, the program should print a **different message**:

* If the number **is** **88** - **"Hello"**
* If the number **is** **86** - **"How are you?"**
* If the number is **not** **88** **nor** **86**, and it is **below 88** - **"GREAT!"**
* If the number is **over 88** - **"Bye."**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  88  86  2  105 | Hello  How are you?  GREAT!  Bye. |
| 3  88  88  89 | Hello  Hello  Bye. |

n = int(input())  
  
for i in range(0,n):  
 current\_number = int(input())  
  
 if current\_number == 88:  
 print("Hello")  
 elif current\_number == 86:  
 print(f"How are you?")  
 elif current\_number < 88:  
 print("GREAT!")  
 elif current\_number >88:  
 print("Bye.")

## Maximum Multiple

On the first line, you will be given a positive number, which will serve as a **divisor**. On the second line, you will receive a positive number that will be the **boundary**. You should find the **largest** integer **N**, that is:

* **divisible by the given divisor**
* **less than or equal to the given bound**
* **greater than 0**

***Note:*** it is guaranteed that **N** **is found**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  7 | 6 |
| 10  50 | 50 |
| 37  200 | 185 |

import sys  
  
first\_number = int(input())  
second\_number = int(input())  
max\_number = - sys.maxsize  
  
for i in range(first\_number, second\_number+1):  
 if i > 0 and i % first\_number == 0:  
 max\_number =i  
  
print(max\_number)

## Orders

You work at a coffee shop, and your job is to **place orders** to the distributors. Thus, you want to know **the price** **of each order**. On the first line, you will receive integer **N** - the **number of orders** the shop will receive. For each order, you will receive the following information:

* Price per capsule - a **floating-point number** in the range **[0.01…100.00]**
* Days - **integer** in the range **[1…31]**
* Capsules, needed per day - **integer** in the range **[1…2000]**

For **each order**, you should **print a single line** in the following format:

* **"The price for the coffee is: ${price}"**

If you do **not receive a correct order** (one or more of the values are not in the given range),you should **ignore it** and **move to the next one.**

After you **go through all orders**, you need to **print the** **total price** in the following format:

* **"Total: ${**total\_p**rice}"**

Both the **price of a coffee** andthe **total price must be formatted** to the **second decimal place**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1  1.53  30  8 | The price for the coffee is: $367.20  Total: $367.20 |
| 2  4.99  31  3  0.35  31  5 | The price for the coffee is: $464.07  The price for the coffee is: $54.25  Total: $518.32 |
| 2  9.223  31  0  0.05  10  30 | The price for the coffee is: $15.00  Total: $15.00 |

number\_of\_orders = int(input())  
total\_price = 0  
  
for i in range(0,number\_of\_orders):  
 price\_per\_capsule = float(input())  
 days = int(input())  
 capsules = int(input())  
  
 if 0.01 <= price\_per\_capsule <= 100.00 and 1 <= days <=31 and 1<= capsules <= 2000:  
 price = price\_per\_capsule \* days \* capsules  
 total\_price += price  
 print(f"The price for the coffee is: ${price:.2f}")  
  
 else:  
 continue  
  
print(f"Total: ${total\_price:.2f}")

## String Pureness

You will be given number **n**. After that, you'll receive different strings **n** times. Your task is to check if the given strings are pure, meaning that they do **NOT** consist of any of the characters: **comma** **","**, **period** **"**.**"**, or **underscore** **"\_"**:

* If a **string is pure**, print **"{string} is pure."**
* Otherwise, print **"{string} is not pure!"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  pure string  not\_pure\_string | pure string is pure.  not\_pure\_string is not pure! |
| 3  SoftUni  12345  string.pureness | SoftUni is pure.  12345 is pure.  string.pureness is not pure! |

n = int(input())  
  
for i in range(0,n):  
 text = input()  
  
 if "," in text:  
 print(f"{text} is not pure!")  
 elif "." in text:  
 print(f"{text} is not pure!")  
 elif "\_" in text:  
 print(f"{text} is not pure!")  
 else:  
 print(f"{text} is pure.")

## Double Char

You will be given **strings until you receive the command "End"**. For each string given, you should **print** a **string** in which **each character** (case-sensitive) is **repeated twice**. Note that if you receive the **string** **"SoftUni"**, you should **NOT** print it!

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hello World  Repeat  End | HHeelllloo WWoorrlldd  RReeppeeaatt |
| 1234!  SoftUni  softuni  End | 11223344!!  ssooffttuunnii |

result = ''  
while True:  
 text = input()  
  
 if text =='SoftUni':  
 continue  
 if text == 'End':  
 break  
  
 for letter in text:  
 result +=letter \* 2  
 print(result)  
 result = ''

## How Much Coffee Do You Need?

*Everybody knows that you spend too much time awake during nighttime.*

Your task is to define how much coffee you need to stay awake. Until you receive the command **"END"**, you need to read **commands on different lines**. According to the commands, **calculate the number of coffees** you need to drink to stay awake during the daytime.

The list of **events** can contain the following:

* You have homework to do (**"coding"**).
* You have a dog or a cat that decided to wake you up too early (**"dog"** or **"cat"**).
* You watch a movie (**"movie"**).
* If **other** **events** are present, they will be represented by arbitrary strings. Just **ignore** them!

Each event can be **lowercase** or **uppercase**:

* If it is **lowercase,** you need **1 coffee** by an event.
* If it is **uppercase,** you need **2 coffees** by an event.

In the end, print the **number of coffees** you will need. **If the count has exceeded 5,** justprint **"You need extra sleep"**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| dog  CAT  gaming  END | 3 |
| movie  CODING  MOVIE  CLEANING  cat  END | You need extra sleep |

number\_of\_coffees = 0  
  
while True:  
 command = input()  
  
 if command == 'END':  
 print(f'{number\_of\_coffees}')  
 break  
   
 if command =='coding' or command =='cat' or command =='dog' or command =='movie':  
 number\_of\_coffees +=1  
 elif command =='CODING' or command =='CAT' or command =='DOG' or command =='MOVIE':  
 number\_of\_coffees +=2  
 else:  
 continue  
  
 if number\_of\_coffees >5:  
 print(f"You need extra sleep")  
 break

## Sorting Hat

Help out the sorting hat to sort the new students in the houses of Hogwarts. You will be receiving **names** until the command **"Welcome!"**. The length of each name determines in which house the student is going:

* If the name is less than 5 chars, the student is going into Gryffindor
  + Print **"{name} goes to Gryffindor."**
* If the name is exactly 5 chars, the student is going into Slytherin
  + Print **"{name} goes to Slytherin."**
* If the name is exactly 6 chars, the student is going into Ravenclaw
  + Print **"{name} goes to Ravenclaw."**
* If the name is more than 6 chars, the student is going into Hufflepuff
  + Print **"{name} goes to Hufflepuff."**

While receiving names, if you receive **"Voldemort"**, print **"You must not speak of that name!"** and end the program. No more sorting for today!

If **all students** are sorted successfully, print **"Welcome to Hogwarts."**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Harry  Ron  Ginny  Draco  Welcome! | Harry goes to Slytherin.  Ron goes to Gryffindor.  Ginny goes to Slytherin.  Draco goes to Slytherin.  Welcome to Hogwarts. |
| Luna  Hermione  Neville  Voldemort | Luna goes to Gryffindor.  Hermione goes to Hufflepuff.  Neville goes to Hufflepuff.  You must not speak of that name! |

while True:  
 name = input()  
  
 if name =="Welcome!":  
 print(f"Welcome to Hogwarts.")  
 break  
  
 if name == "Voldemort":  
 print(f"You must not speak of that name!")  
 break  
  
 if len(name) < 5:  
 print(f"{name} goes to Gryffindor.")  
 elif len(name) == 5:  
 print(f"{name} goes to Slytherin.")  
 elif len(name) == 6:  
 print(f"{name} goes to Ravenclaw.")  
 elif len(name) > 6:  
 print(f"{name} goes to Hufflepuff.")

## \* Mutate Strings

You will be given **two strings**. **Transform the first** string into **the second** one, **letter** by letter, starting from the first one.After **each interaction**, print the **resulting** string **only if it is unique**.

***Note:*** the strings will have the same length.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| bubble gum  turtle hum | tubble gum  turble gum  turtle gum  turtle hum |
| Kitty  Doggy | Ditty  Dotty  Dogty  Doggy |

first\_string = input()  
second\_string = input()  
  
result =first\_string  
  
for letter in range(len(first\_string)):  
 if first\_string[letter] == second\_string[letter]:  
 continue  
 result = second\_string[:letter+1] + first\_string[letter+1:]  
 print(result)

## \* Easter Bread

*Since it is Easter, you have decided to make some loaves of Easter* *bread and exchange them for eggs.*

Create a program that **calculates** how many **loaves** you can make **(according to the recipe**) with the **budget** you **have**.

Here is the **recipe** for **one** loaf:

|  |  |
| --- | --- |
| **Eggs** | **1 pack** |
| **Flour** | **1 kg** |
| **Milk** | **0.250 l** |

**First**, you will **receive** your **budget**. Then, you will **receive** the **price** for **1 kg flour**. The **price for 1 pack of eggs** is **75%** of the **price** **for 1 kg flour**. The **price** for **1L** **milk** is **25%** **more** than the price for **1 kg flour**. Keep in mind that you use only **250ml milk for a bread**.

**Start** cooking the loaves and **keep making** them until you have **enough budget**. Keep in mind that:

* For **every** loaf of bread that you make, you will receive **3 colored eggs**.
* For **every** **3rd** bread you make, you will lose some of your **colored** eggs **after receiving** the usual **3 colored eggs** for your bread. The count of eggs you will lose is calculated when you **subtract** **2** from your **current** **count** of loaves - **({current\_bread\_count} - 2)**

In the end, print the loaves of bread you made, the eggs you have collected, and the money you have **left**, **formatted** to the **2nd decimal place**, in the following format:

**"You made {number\_of\_loaves} loaves of Easter bread! Now you have {colored\_eggs} eggs and {money\_left}BGN left."**

### Input / Constraints

* On the **1st line**, you will receive the budget - a **real number** in the range [0.0…100000.0]
* On the **2nd line**, you will receive the price for **1 kg flour** - a **real number** in the range [0.0…100000.0]
* The input will always be in the correct format
* You will **always** have a **remaining** **budget**
* There will **not** be a case in which the **eggs** become a **negative** **count**

### Output

* In the end, print the **number** of loaves of **Easter bread** you have made, the colored **eggs** you have gathered, and the **money** **formatted** to **the 2nd** decimal place in the format described above.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20.50  1.25 | You made 7 loaves of Easter bread! Now you have 16 eggs and 2.45BGN left. |
| 15.75  1.4 | You made 5 loaves of Easter bread! Now you have 14 eggs and 1.31BGN left. |

budget= float(input())  
flour\_price\_per\_kg = float(input())  
colored\_eggs = 0  
number\_of\_loaves = 0  
  
eggs\_price\_per\_pack = 0.75 \* flour\_price\_per\_kg  
milk\_price\_per\_litre = flour\_price\_per\_kg \* 1.25  
milk\_price\_for\_recipe = milk\_price\_per\_litre / 4  
  
price\_per\_bread = flour\_price\_per\_kg +eggs\_price\_per\_pack + milk\_price\_for\_recipe  
  
while budget > price\_per\_bread:  
 colored\_eggs += 3  
 number\_of\_loaves += 1  
 budget -= price\_per\_bread  
  
 if number\_of\_loaves>0 and number\_of\_loaves % 3 ==0:  
 colored\_eggs -=(number\_of\_loaves -2)  
  
print(f"You made {number\_of\_loaves} loaves of Easter bread! Now you have {colored\_eggs} eggs and {budget:.2f}BGN left.")

## \* Christmas Spirit

*It is time to get in a Christmas mood. You need to decorate the house in time for the big event, but you have limited days to do so.*

Write a program that calculates **how much money you will need to spend** on Christmas decorations and **how much your Christmas spirit will improve**.

On the first line, you will receive the **quantity of decorations** you should buy **each time** you go shopping.

On the second line, you will receive the **days** **left** **until Christmas**.

There are **4 types** of decorations, and each **piece** costs a certain **price.** Also, **each time you go shopping** for a concrete type of decoration,your Christmas spirit **is improved by some points:**

|  |  |  |
| --- | --- | --- |
| **Decoration** | **Price/Piece** | **Points/Shopping** |
| Ornament Set | 2$ | 5 |
| Tree Skirt | 5$ | 3 |
| Tree Garland | 3$ | 10 |
| Tree Lights | 15$ | 17 |

Until Christmas, you go shopping for a certain decoration as follows:

* Every **second day** you buy **Ornament Sets**.
* Every **third day** you buy **Tree Skirts** and **Tree Garlands**.
* Every **fifth day** you buy **Tree Lights**.
  + If you have bought Tree Skirts and Tree Garlands on the **same day,** you **additionally increase** your spirit by **30**.

That's not all! You have a cat at home that really likes to mess around with the decoration:

* Every **tenth day** your cat ruins all tree decorations, and you **lose 20 points of the spirit:**
  + Because of that, you go shopping (for a second time during the day) to buy **one** piece of tree **skirt**, **garlands,** and **lights**, but you do **NOT** earn additional spirit points for them.
* Also, because of the cat - at the **beginning** of **every eleventh day**, you are forced to **increase** the **quantity** of decorations needed to be bought each time you go shopping **by 2**.
* If the **last day** is a **tenth day**, the cat demolishes even more and ruins the Christmas turkey, and you **lose** an additional **30 points of spirit**.

In the end, you must print the **total cost** and the **gained spirit**.

### Input / Constraints

The input will consist of **exactly 2 lines**:

* quantity - **integer in the range [1…100]**
* days **- integer in the range [1…100]**

### Output

In the end, print the **total cost** and the total gained **spirit** in the following format:

* **"Total cost: {budget}"**
* **"Total spirit: {totalSpirit}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1  7 | Total cost: 37  Total spirit: 58 |
| 3  20 | Total cost: 558  Total spirit: 156 |

quantity\_decorations = int(input())  
remaining\_days = int(input())  
total\_cost = 0  
total\_spirit =0  
  
for day in range(1,remaining\_days+1):  
  
 if day % 11 == 0:  
 quantity\_decorations +=2  
  
 if day % 2 == 0:  
 total\_cost += 2\* quantity\_decorations  
 total\_spirit += 5  
 if day % 3 == 0:  
 total\_cost += (5 \* quantity\_decorations) + (3 \* quantity\_decorations)  
 total\_spirit += 13  
 if day % 5 == 0:  
 total\_cost += 15 \* quantity\_decorations  
 total\_spirit += 17  
 if day % 3 == 0:  
 total\_spirit +=30  
  
 if day % 10 == 0:  
 total\_spirit -=20  
 total\_cost += 5 + 3 + 15  
if remaining\_days % 10 ==0:  
 total\_spirit -=30  
  
print(f"Total cost: {total\_cost}")  
print(f"Total spirit: {total\_spirit}")