# Exercise: Conditional Statements

Problems for in-class and homework exercises for the course ["Programming Basics" @ SoftUni](https://softuni.org/).

Test your solutions in the **Judge** system: <https://judge.softuni.org/Contests/Compete/Index/3490>

## Sum Seconds

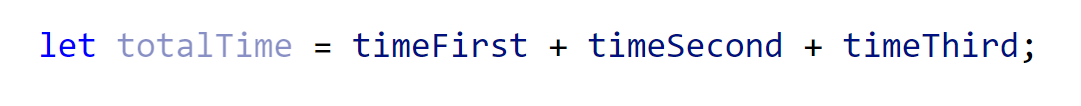
Three athletes finish in some **number of seconds** (between **1** and **50**). Write a function solve(seconds1, seconds2, seconds3) that calculates their **total time** in the format "**minutes:seconds**". The seconds are to be output with **leading zero** (2 "02", 7 "07", 35 "35").

**Sample Input and Output**

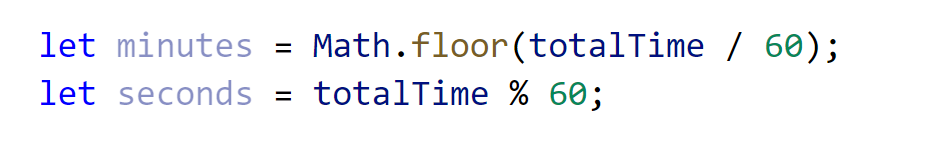
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| solve(35,45,44) | 2:04 | solve(22,7,34) | 1:03 | solve(50,50,49) | 2:29 | (14, 12,  10) | 0:36 |

### Hints and Guidelines

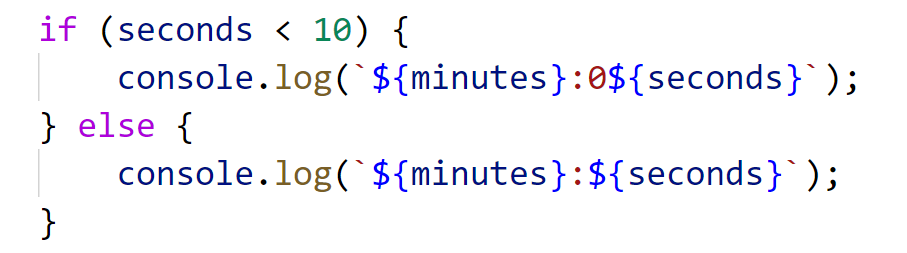
1. Convert the given seconds from strings to numbers.
2. Create a **new variable** to store the **sum of the seconds of the three athletes**:



1. After you have found the **sum of the seconds** you need to **convert them into minutes and seconds** (for example, if the sum is **85 seconds that is 1 minute and 25 seconds because 1 minute has 60 seconds**). Create **two new variables**. In the first, calculate **how many minutes the sum of seconds is** by **dividing the sum by 60**. In the second variable, **calculate the seconds by dividing with the remainder (%).** Use **division with remainder (%)** to take the **remainder when divided by 60**, which is the remaining seconds. For example, you have a total of 134 seconds (2 minutes and 14 seconds) **after integer division (/) by 60 you will get 2, and after the division with remainder (%) you will get the remaining seconds (14).**



1. Once you know **how many minutes and seconds** the total sum is, you need to print them in the correct format (**minutes:seconds**). If the seconds are **less than 10** you need to print **0 before the seconds**, otherwise just print the **result in the given format**. To do this, use a **conditional statement (if)**. You can use a **placeholder** for printing.



## Bonus Score

**An integer** is received - an initial number of points. **Bonus points** are added to it according to the rules described below. Write a function bonus(points) that calculates the **bonus points the number receives** and **the total number of points** (the number + the bonus).

* If the number is **up to 100** (inclusive), the bonus points are **5**.
* If the number is **greater** **than 100**, bonus points are **20%** of the number.
* If the number is **greater** **than 1000**, the bonus points are **10%** of the number.
* Additional bonus points (added separately from the previous ones):
  + For an **even** number 🡪 + 1 point.
  + For a number **ending in 5** 🡪 + 2 points.

**Sample Input and Output**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| bonus(20) | 6  26 | bonus(175) | 37  212 | bonus(2703) | 270.3  2973.3 | bonus(15875) | 1589.5  17464.5 |

### Hints and Guidelines

1. Convert your input data from string to number.
2. Create a **new variable** in which you will calculate the **accumulated bonus points** by giving it a starting **value of 0.0**.



1. Make an **if-else-if** construction for the first **three conditional statements** to check the size of the number and calculate the bonus.



1. Make a new **if-else-if construction** to perform the checks and **calculate the additional bonus**. If the number **is even add 1 to the bonus accumulated so far**, and if **it ends in 5 add 2 to the bonus**. To check if a number **is even you need to divide it by 2 and if you get a remainder when dividing by 0**, then the number is **even**, but if you **get a remainder of 1**, it means the number is **odd**. For example, the number 34 is even because 34 / 2 = 17 and the remainder is 0 and the number 35 is odd because 35 / 2 = 17 with a remainder of 1. To check if a number ends in 5 you have to **divide the number by 10** and if you **get a remainder when dividing by 5**, then the number ends in 5. For example the number 245 / 10 = 24 with remainder 5.



1. Print the results **in two lines**. On the first line is **the accumulated bonus**, and on the second line is **the final number**, which you will find by **adding the initial number of points to the bonus**.



## Time + 15 Minutes

Write a function solve(hours, minutes) that **receives the hour and minutes** of a 24-hour day and calculates what **will be the time in 15 minutes**. Print the result in **hours:minutes** format. The hours are always between 0 and 23, and the minutes are always between 0 and 59. The hours are written in one or two digits. Minutes are always written in two digits, with a **leading zero** where necessary.

### Sample Input and Output

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| solve(1, 46) | 2:01 | solve(0, 01) | 0:16 | solve(23, 59) | 0:14 | solve(11, 08) | 11:23 | solve(12, 49) | 13:04 |

# Sample Exam Problems

## Toy Shop

Sarah has a toy shop. She receives a big order that she has to complete. With the money she will earn she wants to go on a trip.

**Toy prices:**

* **Puzzle - 2.60 USD**
* **Talking doll – 3 USD**
* **Teddy bear - 4.10 USD**
* **Minion - 8.20 USD**
* **Truck – 2 USD**

If the ordered toys are **50 or more**, the store makes a **discount of 25% of the total price**. From the earned money Sarah must give **10% for the store rent**. Calculate whether the money will be enough for her to go on a trip.

### Input Data

Write a function solve(price, puzzles, dolls, bears, minions, trucks) with the following arguments:

1. **Price of the trip – a floating-point number in the range 1.00 … 10000.00**
2. **Number of puzzles – an integer in the range 0...**  **1000**
3. **Number of talking dolls – an integer in the range 0 ...**  **1000**
4. **Number of teddy bears – an integer in the range 0 ...**  **1000**
5. **Number of minions – an integer in the range 0 ...**  **1000**
6. **Number of trucks – an integer in the range 0 ...**  **1000**

### Output Data

Print on the console:

* If **the money is enough,** print:
  + **"Yes! {the remaining money} USD left."**
* If there is **NOT enough money**, print:
  + **"Not enough money! {the needed money} USD needed."**

**Format the result to 2 digits after the decimal point**.

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve(40.8,  20,  25,  30,  50,  10) | Yes! 418.20 USD left. | **Sum**: 20 \* 2.60 + 25 \* 3 + 30 \* 4.10 + 50 \* 8.20 + 10 \* 2 = **680** USD  **Number of toys**: 20 + 25 + 30 + 50 + 10 = **135**  **135 > 50 => 25% discount**; 25% of 680 = **170 USD discount**  **Final price**: 680 – 170 = **510** USD  **Rent**: 10% of 510 USD = **51** USD  **Profit**: 510 – 51 = **459** USD  **459 > 40.8** =>459 – 40.8= **418.20** USD **left** |
| **Input** | **Output** | **Comments** |
| solve(320,  8,  2,  5,  5,  1) | Not enough money! 238.73 USD needed. | **Sum**: 8 \* 2.60 + 2 \* 3 + 5 \* 4.10 + 5 \* 8.20 + 1 \* 2 = **90.3** USD  **Number of toys**: 8 + 2 + 5 + 5 + 1 = **21**  **21 < 50 => no discount**  **Rent**: 10% of 90.3 = **9.03** USD  **Profit**: 90.3 – 9.03 = **81.27** USD  **81.27 < 320** => 320 – 81.27= **238.73** USD **needed** |

## Godzilla vs. Kong

Filming for the highly anticipated “Godzilla vs. Kong” movie has begun. Screenwriter Adam Wingard asks you to **write a** function solve(budget, people, pricePerClothes) that will calculate **if the budget is enough** to make the film. The filming will require a **certain number of statisticians, clothing** for each statistician, and **scenery**.

It is known that:

* + The scenery for the film **is worth 10% of the budget.**
  + With **more than 150 statisticians, there is a 10% discount on clothing.**

### Input Data

The function receives **3 arguments**:

1. **Budget for the film – a floating-point number in the range 1.00 … 1000000.00**
2. **Number of statisticians – an integer in the range 1 … 500**
3. **Price for clothing of a statistician – a floating-point number in the range 1.00 … 1000.00**

### Output Data

**Two lines** must be printed on the console:

* If the money for the scenery and clothing **is more than the budget**:
  + "Not enough money!"
  + "Wingard needs {the needed money} USD more."
* If the money for the scenery and clothing **is less or equal to the budget**:
  + "Action!"
  + "Wingard starts filming with {the remaining money} USD left."

**Format the result to 2 digits after the decimal point**.

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve(20000,  120,  55.5) | Action!  Wingard starts filming with 11340.00 USD left. | Amount for scenery: 10% от 20000 = 2000 USD  Amount for clothing: 120 \* 55.5 = 6660 USD  Total amount for the film: 2000 + 6660 = 8660 USD  20000 – 8660 = 11340 USD left |
| solve(15437.62,  186,  57.99) | Action!  Wingard starts filming with 4186.33 USD left. | Amount for scenery: 10% от 15437.62 = 1543.762 USD  Amount for clothing: 186 \* 57.99 = 10786.14 USD  There are more than 150 statisticians, therefore there is a 10% discount on clothing.  10% of 10786.14 is 1078.614  10786.14 – 1078.614 = 9707.526 USD for clothing  Total amount for the film: 1543.762 + 9707.526 = 11251.288  15437.62 – 11251.288 = 4186.331 USD left |
| solve(9587.88,  222,  55.68) | Not enough money!  Wingard needs 2495.77 USD more. | Amount for scenery: 10% от 9587.88 = 958.788 USD  Amount for clothing: 11124.864 USD  Total amount for the film: 958.788 + 11124.864 = 12083.652  9587.88 – 12083.652 = 2495.77 USD needed |

## World Swimming Record

Andrew decides to improve the World Record in long-distance swimming. **The** function solve(seconds, meters, secondPerMeter) **receives the record in seconds that Andrew** **has to improve, the distance in meters that he has to swim, and the time in seconds for which he swims a distance of 1 m.** Write a function that calculates whether he has done the task, given that: **the resistance of the water slows him down every 15 m by 12.5 seconds.** When calculating how many times Andrew will slow down because of the water resistance, **the result should be rounded down to the nearest integer number.**

**Calculate the time in seconds for Andrew** **to swim the distance and the difference to the World Record.**

### Input Data

The function receives **3 arguments**:

1. **The record in seconds – a floating-point number in the range 0.00 ...**  **100000.00**
2. **The distance in meters – a floating-point number in the range 0.00 ...**  **100000.00**
3. **The time in seconds for which he swims a distance of 1 m. – a floating-point number in the range 0.00 ...**  **1000.00**

### Output Data

Print on the console the output, depending on the result:

* If **Andrew** **has improved the World Record (his time is less than the record)** we print:
  + **" Yes, he succeeded! The new world record is {Andrew's time} seconds."**
* If **he has NOT improved the record (his time is greater than or equal to the record)** we print:
  + **"No, he failed! He was {the** **failing seconds} seconds slower."**

**Format the result to 2 digits after the decimal point**.

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve(10464,  1500,  20) | No, he failed! He was 20786.00 seconds slower. | **Andrew** **must swim 1500 m.: 1500 \* 20 = 30000 sec.**  **Every 15 meters. 12.5 sec. is added to his** **time:**  **1500 / 15 = 100 \* 12.5 = 1250 sec.**  **Total time: 30000 + 1250 = 31250 sec.**  **10464 < 31250**  **The time he didn't have to improve the record:**  **31250 – 10464 = 20786 sec.** |
| **Input** | **Output** | **Comments** |
| solve(55555.67,  3017,  5.03) | Yes, he succeeded! The new world record is 17688.01 seconds. | **Andrew** **must swim 3017 m.: 3017 \* 5.03 = 15175.51 sec.**  **Every 15 meters. 12.5 sec. is added to his** **time:**  **3017/ 15 = 201 \* 12.5 = 2512.50 sec.**  **Total time: 15175.51 + 2512.50 = 17688.01 sec.**  **The record is improved: 55555.67 > 17688.01** |

## 7. Shopping

Peter wants to buy **N** video cards, **M** processors, and **P** number of RAM. If the number of video cards is greater than the number of processors, he gets a **15% discount** on the final bill.

The following prices apply:

* Video card – **250 USD/per piece**.
* Processor – **35% of the price of purchased video cards/per piece**.
* RAM – **10% of the price of purchased video cards/per piece**.

Calculate the amount needed to purchase the materials and calculate if the budget will be enough.

### Input Data

The function calculate(budget, videoCards, processors, RAMs) receives **4 arguments**:

1. Peter's budget – **a floating-point** number in the range **0.0... 100000.0**
2. The number of video cards – **an integer** in the range **0... 100**
3. The number of processors – **an integer** in the range **0... 100**
4. The number of RAM – **an** **integer** in the range **0... 100**

### Output Data

One line is printed on the console:

* If the budget is enough:

"**You have {the remaining budget} USD left!**"

* If the amount exceeds the budget:

"**Not enough money! You need {the needed money} USD more!**"

**Format the result to 2 digits after the decimal point.**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| calculate(900,  2,  1,  3) | You have 198.75 USD left! | Budget: 900 USD  Amount for video cards: 2 \* 250 = 500 USD  The price for processor: 35% от 500 = 175 USD  Amount for processors: 1 \* 175 = 175 USD  The price for RAM: 10% от 500 = 50 USD  Amount for RAM: 3 \* 50 = 150 USD  Total amount: 500 + 175 + 150 = 825 USD  The number of video cards is greater than the number of processors, so Peter receives 15% discount from the final price: 825 – 15% = 701.25 USD  701.25 <= 900  => the money is enough  => remain 900 – 701.25 = 198.75 USD |
| calculate( (920.45,  3,  1,  1) | Not enough money! You need 3.92 USD more! | Budget: 920.45 USD  Amount for video cards: 3 \* 250 = 750 USD  The price for processor: 35% от 750 = 262.50 USD  Amount for processors: 1 \* 262.50 = 262.50 USD  The price for RAM: 10% от 750 = 75 USD  Amount for RAM: 1 \* 75 = 75 USD  Total amount: 750 + 262.50 + 75 = 1087.50 USD  The number of video cards is greater than the number of processors, so Peter receives 15% discount from the final price: 1087.50 – 15% = 924.37 USD  924.37 > 920.45  => the money is not enough  => needed 924.375 – 920.45 = 3.92 USD |

## 8. Lunch Break

During your lunch break, you want to watch an episode of your favorite series. Your task is to write a program to find out if you **have enough time** to watch the episode. During the break, you take **time for lunch** and **time for relaxation. The lunchtime** will be **1/8 of the break time** and **the relaxation time** will be **1/4 of the break time**.

### Input Data

The function solve(name, episodeDuration, breakDuration) receives **3 arguments**:

1. **Series name** – **text**
2. **Episode duration** – **an integer** in the range **10… 90**
3. **Break duration** – **an integer** in the range **10… 120**

### Output Data

Print one line on the console:

* If **time is enough** to watch the episode:

"**You have enough time to watch {the series name} and left with {the remaining time} minutes free time.**"

* If **you don't have enough time**:

"**You don't have enough time to watch {the series name}, you need {the needed time} more minutes.**"

**Time should be rounded up to the nearest integer number.**

### Sample Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| solve("Game of Thrones",  60,  96) | You have enough time to watch Game of Thrones and left with 0 minutes free time. | Time for lunch: 96 \* 1/8 = 12.0  Time for relaxation: 96 \* 1/4 = 24.0  Remaining time: 96 - 12 - 24 = 60  The remaining time is greater than or equal to the duration of the episode, therefore we print the appropriate output. |
| solve("Teen Wolf",  48,  60) | You don't have enough time to watch Teen Wolf, you need 11 more minutes. | Time for lunch: 60 \* 1/8 = 7.5  Time for relaxation: 60 \* 1/4 = 15.0  Remaining time : 60 - 7.5 - 15 = 37.5  The remaining time is less than the duration of the episode, therefore we print the appropriate output. |