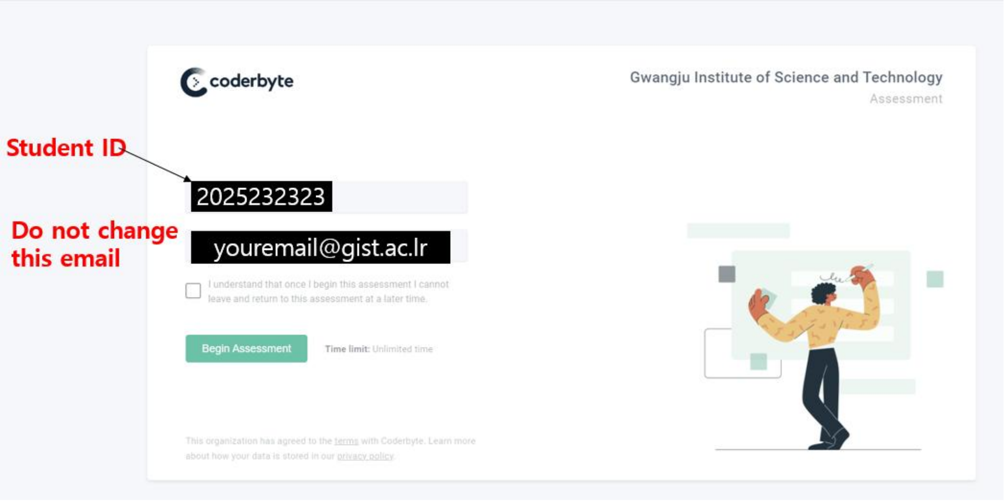
Algo Assignment 1: Basic Python Skills

General Instruction to submit the assignment

1. Open the link sent in email and enter your **Student ID** and **ZEUS email address**



Student ID

2025232323

Do not change this email

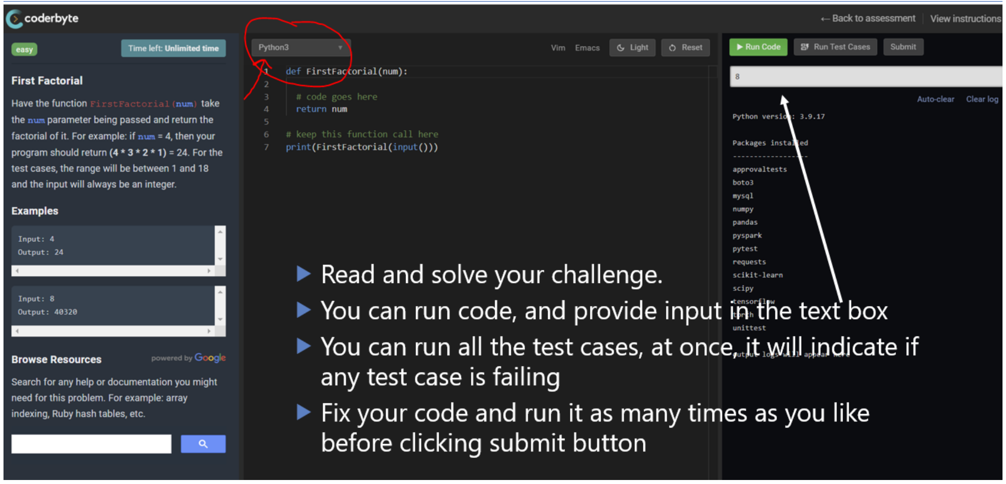
youremail@gist.ac.lr

□ I understand that once I begin this assessment I cannot leave and return to this assessment at a later time.

Time limit: Unlimited time

This organization has agreed to the terms with Coderbyte. Learn more about how your data is stored in our privacy policy.

2. You must change the programming language to Python3 before starting to code. There is a drop down button above the coding window.



First Factorial

Have the function `FirstFactorial(num)` take the `num` parameter being passed and return the factorial of it. For example: if `num` = 4, then your program should return (4 * 3 * 2 * 1) = 24. For the test cases, the range will be between 1 and 18 and the input will always be an integer.

Examples

```
Input: 4
Output: 24
```

```
Input: 8
Output: 40320
```

Browse Resources

Search for any help or documentation you might need for this problem. For example: array indexing, Ruby hash tables, etc.

```python
def FirstFactorial(num):

  # code goes here
  return num

# keep this function call here
print(FirstFactorial(input()))
```

```
8
```

```
Python version: 3.9.17

Packages installed
------------------
approvaltests
boto3
mysql
numpy
pandas
pyspark
pytest
requests
scikit-learn
scipy
tensorflow
torch
unittest

output logs will appear here
```

- Read and solve your challenge.
- You can run code, and provide input in the text box
- You can run all the test cases, at once, it will indicate if any test case is failing
- Fix your code and run it as many times as you like before clicking submit button

3. If you submit and try to resubmit etc., the system will not let you do that. (once submitted you can not update your code)
4. Hence solve this question before hand and then enter your code in Coderbyte system. This way you will save a lot of hassle. The questions will also be released in the pdf format. Make sure your code is giving right results for all the given test cases before you submit your code.
5. You should not use **system libraries** while solving questions for at least first 3 assignments. Starting from DP assignments you can use **min max** and **sort** functions of python3.
6. Remember total points from the assignments are only **10%** of your grades. Hence these are mainly for practicing for mid-term and final-term coding test.
7. The time complexity of your submissions will be tested only after 4th Assignment.

Q1. First Factorial

Have the function FirstFactorial(**num**) take the **num** parameter being passed and return the factorial of it. For example: if **num** = 4, then your program should return **(4 * 3 * 2 * 1)** = 24. For the test cases, the range will be between 1 and 18 and the input will always be an integer.

Code frame

```
function FirstFactorial(num) {

  // code goes here
  return num;

}

// keep this function call here
console.log(FirstFactorial(readline()));
```

Test cases

1. For input 3, the correct output is 6
2. For input 4, the correct output is 24
3. For input 5, the correct output is 120
4. For input 6, the correct output is 720
5. For input 7, the correct output is 5040
6. For input 8, the correct output is 40320
7. For input 9, the correct output is 362880
8. For input 10, the correct output is 3628800

Q2. Longest Word

Have the function `LongestWord(sen)` take the **`sen`** parameter being passed and return the longest word in the string. If there are two or more words that are the same length, return the first word from the string with that length. Ignore punctuation and assume **`sen`** will not be empty. Words may also contain numbers, for example "Hello world123 567"

Code frame

```
function LongestWord(sen) {

  // code goes here
  return sen;

}

// keep this function call here
console.log(LongestWord(readline()));
```

Test cases

1. For input `"hello world"`, the correct output is `hello`

2. For input `"this is some sort of sentence"`, the correct output is `sentence`

3. For input `"longest word!!"`, the correct output is `longest`

4. For input `"a beautiful sentence^&!"`, the correct output is `beautiful`

5. For input `"oxford press"`, the correct output is `oxford`

6. For input `"123456789 98765432"`, the correct output is `123456789`

7. For input `"letter after letter!!"`, the correct output is `letter`

8. For input `"a b c dee"`, the correct output is `dee`

9. For input `"a confusing /:sentence:/[ this is not!!!!!!!~"`, the correct output is `confusing`

Q3. First Reverse

Have the function `FirstReverse(str)` take the **`str`** parameter being passed and return the string in reversed order. For example: if the input string is "Hello World and Coders" then your program should return the string **sredoC dna dlroW olleH**.

Code frame

```
function FirstReverse(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(FirstReverse(readline()));
```

Test cases

1. For input `"Coderbyte"`, the correct output is `etybredoC`

2. For input `"I Love Coding"`, the correct output is `gnidoC evoL I`

3. For input `"h333llLo"`, the correct output is `oLll333h`

4. For input `"Yo0"`, the correct output is `0oY`

5. For input `"thisiscool"`, the correct output is `loocsisiht`

6. For input `"commacomma!"`, the correct output is `!ammocammoc`

7. For input `"123456789"`, the correct output is `987654321`

8. For input `"lettersz!23z"`, the correct output is `z32!zsrettel`

9. For input `"aq"`, the correct output is `qa`

Q4. Letter Changes

HIDE QUESTION

Have the function `LetterChanges(str)` take the **`str`** parameter being passed and modify it using the following algorithm. Replace every letter in the string with the letter following it in the alphabet (**ie.** c becomes d, z becomes a). Then capitalize every vowel in this new string (a, e, i, o, u) and finally return this modified string.

Code frame

```
function LetterChanges(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(LetterChanges(readline()));
```

Test cases

1. For input `"hello world"`, the correct output is `Ifmmp xpsmE`

2. For input `"sentence"`, the correct output is `tfOUfOdf`

3. For input `"replace!*"`, the correct output is `sfqmbdf!*`

4. For input `"coderbyte"`, the correct output is `dpEfsczUf`

5. For input `"beautiful^"`, the correct output is `cfbvUjgvm^`

6. For input `"oxford"`, the correct output is `pygpsE`

7. For input `"123456789ae"`, the correct output is `123456789bf`

8. For input `"this long cake@&"`, the correct output is `UIjt mpOh dblf@&`

9. For input `"a b c dee"`, the correct output is `b c d Eff`

10. For input `"a confusing /:sentence:/[ this is not!!!!!!!~"`, the correct output is `b dpOgvtjOh /:tfOUfOdf:/[ UIjt jt OpU!!!!!!!~`

Q5. Simple Adding

HIDE QUESTION

Have the function `SimpleAdding(num)` add up all the numbers from 1 to **num**. For example: if the input is 4 then your program should return 10 because 1 + 2 + 3 + 4 = 10. For the test cases, the parameter **num** will be any number from 1 to 1000.

Code frame

```
function SimpleAdding(num) {

  // code goes here
  return num;

}

// keep this function call here
console.log(SimpleAdding(readline()));
```

Test cases

1. For input `45`, the correct output is `1035`

2. For input `13`, the correct output is `91`

3. For input `2`, the correct output is `3`

4. For input `5`, the correct output is `15`

5. For input `156`, the correct output is `12246`

6. For input `999`, the correct output is `499500`

7. For input `67`, the correct output is `2278`

8. For input `123`, the correct output is `7626`

9. For input `9`, the correct output is `45`

10. For input `10`, the correct output is `55`

Q6. Letter Capitalize

Have the function `LetterCapitalize(str)` take the **`str`** parameter being passed and capitalize the first letter of each word. Words will be separated by only one space.

Code frame

```
function LetterCapitalize(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(LetterCapitalize(readline()));
```

Test cases

1. For input `"hello world"`, the correct output is `Hello World`

2. For input `"i love coding"`, the correct output is `I Love Coding`

3. For input `"h3llo yo people"`, the correct output is `H3llo Yo People`

4. For input `"yoooooooo hi"`, the correct output is `Yoooooooo Hi`

5. For input `"thisiscool"`, the correct output is `Thisiscool`

6. For input `"oxford comma"`, the correct output is `Oxford Comma`

7. For input `"letter by letter go"`, the correct output is `Letter By Letter Go`

8. For input `"a b c d e f"`, the correct output is `A B C D E F`

9. For input `"jelloupin here"`, the correct output is `Jelloupin Here`

Q7. Consonant Count

Have the function `ConsonantCount(str)` take the **`str`** string parameter being passed and return the number of consonants the string contains.

Code frame

```
function ConsonantCount(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(ConsonantCount(readline()));
```

Test cases

1. For input `"Hello World"`, the correct output is `7`

2. For input `"Alphabets"`, the correct output is `6`

3. For input `"Development"`, the correct output is `7`

4. For input `"Hewlett-Packard"`, the correct output is `10`

5. For input `"here"`, the correct output is `2`

6. For input `"www"`, the correct output is `3`

7. For input `"w"`, the correct output is `1`

8. For input `"hbhb"`, the correct output is `4`

9. For input `"aaaaaaaa"`, the correct output is `0`

10. For input `"zz*"`, the correct output is `2`

Q8. Number Reverse

Have the function `NumberReverse(str)` take the **`str`** parameter being passed which will be a string of numbers, and return a new string with the numbers in reverse order.

Code frame

```
function NumberReverse(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(NumberReverse(readline()));
```

Test cases

1. For input `"1 2 3"`, the correct output is `3 2 1`

2. For input `"10 20 50"`, the correct output is `50 20 10`

3. For input `"100 200 34"`, the correct output is `34 200 100`

4. For input `"2 3 1000000"`, the correct output is `1000000 3 2`

5. For input `"123123 2323423 23423412"`, the correct output is `23423412 2323423 123123`

6. For input `"23 23 23 566 76"`, the correct output is `76 566 23 23 23`

Q9. Happy Numbers

HIDE QUESTION

Have the function `HappyNumbers(num)` determine if a number is Happy, which is a number whose sum of the square of the digits eventually converges to 1. Return **true** if it's a Happy number, otherwise return **false**.

For example: the number 10 is Happy because 1^2 + 0^2 converges to 1.

Code frame

```
function HappyNumbers(num) {

  // code goes here
  return num;

}

// keep this function call here
console.log(HappyNumbers(readline()));
```

1. For input `1`, the correct output is `true`

2. For input `2`, the correct output is `false`

3. For input `10`, the correct output is `true`

4. For input `100`, the correct output is `true`

5. For input `101`, the correct output is `false`

6. For input `0`, the correct output is `false`

7. For input `5525`, the correct output is `true`

8. For input `5255`, the correct output is `true`

9. For input `2555`, the correct output is `true`

10. For input `5552`, the correct output is `true`

Q10. Codeland Username Validation

Have the function `CodelandUsernameValidation(str)` take the **`str`** parameter being passed and determine if the string is a valid username according to the following rules:

1. The username is between 4 and 25 characters.
2. It must start with a letter.
3. It can only contain letters, numbers, and the underscore character.
4. It cannot end with an underscore character.

If the username is valid then your program should return the string **true**, otherwise return the string **false**.

Code frame

```
function CodelandUsernameValidation(str) {

  // code goes here
  return str;

}

// keep this function call here
console.log(CodelandUsernameValidation(readline()));
```

Test cases

1. For input `"aaaaaaaaaa"`, the correct output is `true`

2. For input `"aa_"`, the correct output is `false`

3. For input `"u__hello_world123"`, the correct output is `true`

4. For input `"_"`, the correct output is `false`

5. For input `"__bbbbbbb"`, the correct output is `false`

6. For input `"b3333434_"`, the correct output is `false`

7. For input `"usernamehello123"`, the correct output is `true`

8. For input `"oooooooooooooooooo________a"`, the correct output is `false`

9. For input `"123abc444"`, the correct output is `false`

10. For input `"a______b_________555555555555aaaa"`, the correct output is `false`

If there are any issues with submitting assignment, please contact me.

김태성 Taesung Kim

E-MAIL: dampflok@gm.gist.ac.kr

TEL: 010-2777-1326