|  |  |
| --- | --- |
| P1 | initialize\_names.py |
| P2 | battle\_dictionary.py |
| P3 | vowel\_partition.py |
| P4 | factorialize.py |
| P5 | decimalize.py |
| P6 | diagonalize.py |
| P7 | color\_sort.py |

These are case sensitive as usual.

## Use if \_\_name\_\_ == '\_\_main\_\_' to test your code:

Use the if \_\_name\_\_ == '\_\_main\_\_' for any code you run to test your code. If you don't, then when we import your code to run it, whatever test code you have will run.

Remember, the code in the main block only runs when the file considers itself to be \_\_main\_\_ and if we import your code, then your file will not be the main file.

Your function, therefore should not be in the main block. If it is, then it won't be declared by python when the file is imported.

# 

# Problem 1 - Initialize the Names

Write a function get\_initials(the\_name) that takes a string of a name and returns the initials of the name, in upper case, separated by periods, with an additional period at the end.

Input-Output Samples

|  |  |
| --- | --- |
| John Q Public | J.Q.P. |
| Homer J Simpson | H.J.S. |
| Gloria kind of a big deal Tropalogos | G.K.O.A.B.D.T. |

If the argument to the function is the string on the left, then return the string on the right. You don't need to print, since that's the job of wherever the value is returned.

# 

# Problem 2 - Battle Dictionary

# Write a function fight(first\_pokemon, second\_pokemon) that takes the dictionaries for two pokemon and computes the winner of a fight. The winner of the fight must be the pokemon with the greatest sum of all stats. You must also write a stats(pokemon) function that takes a pokemon's dictionary and returns an integer representing the sum of that pokemon's stats.

|  |
| --- |
| {  "name": "bidoof",  "order": 506,  "species": {  "name": "bidoof",  "url": "https://pokeapi.co/api/v2/pokemon-species/399/"  },  "stats": [  {  "base\_stat": 31,  "effort": 0,  "stat": {  "name": "speed",  "url": "https://pokeapi.co/api/v2/stat/6/"  }  },  {  "base\_stat": 40,  "effort": 0,  "stat": {  "name": "special-defense",  "url": "https://pokeapi.co/api/v2/stat/5/"  }  },  {  "base\_stat": 35,  "effort": 0,  "stat": {  "name": "special-attack",  "url": "https://pokeapi.co/api/v2/stat/4/"  }  },  {  "base\_stat": 40,  "effort": 0,  "stat": {  "name": "defense",  "url": "https://pokeapi.co/api/v2/stat/3/"  }  },  {  "base\_stat": 45,  "effort": 0,  "stat": {  "name": "attack",  "url": "https://pokeapi.co/api/v2/stat/2/"  }  },  {  "base\_stat": 59,  "effort": 1,  "stat": {  "name": "hp",  "url": "https://pokeapi.co/api/v2/stat/1/"  }  }  ],  "types": [  {  "slot": 1,  "type": {  "name": "normal",  "url": "https://pokeapi.co/api/v2/type/1/"  }  }  ],  "weight": 200  } |

Dictionary for Ditto

|  |
| --- |
| {  "name": "ditto",  "order": 197,  "species": {  "name": "ditto",  "url": "https://pokeapi.co/api/v2/pokemon-species/132/"  },  "stats": [  {  "base\_stat": 48,  "effort": 0,  "stat": {  "name": "speed",  "url": "https://pokeapi.co/api/v2/stat/6/"  }  },  {  "base\_stat": 48,  "effort": 0,  "stat": {  "name": "special-defense",  "url": "https://pokeapi.co/api/v2/stat/5/"  }  },  {  "base\_stat": 48,  "effort": 0,  "stat": {  "name": "special-attack",  "url": "https://pokeapi.co/api/v2/stat/4/"  }  },  {  "base\_stat": 48,  "effort": 0,  "stat": {  "name": "defense",  "url": "https://pokeapi.co/api/v2/stat/3/"  }  },  {  "base\_stat": 48,  "effort": 0,  "stat": {  "name": "attack",  "url": "https://pokeapi.co/api/v2/stat/2/"  }  },  {  "base\_stat": 48,  "effort": 1,  "stat": {  "name": "hp",  "url": "https://pokeapi.co/api/v2/stat/1/"  }  }  ],  "types": [  {  "slot": 1,  "type": {  "name": "normal",  "url": "https://pokeapi.co/api/v2/type/1/"  }  }  ],  "weight": 40  } |

## Battle Dictionary - Sample Output

# 

# bidoof has 250 combined stats. ditto has 288 combined stats. ditto wins!

# Problem 3 - Vowel Partitions

Write a function count\_blocks(the\_string) in order to count the number of vowels blocks in a set of words. Return the number of vowel blocks from this function.

If separated by a space, you can start over, i.e. "**I** **a**m" has two blocks.

For instance in the blocks:

In "b**aa**bbb**aaa**bbbb**aaaaa**b**a**b" there are 4 vowel blocks.

**"e**nq**ueuei**ng **i**s **a** f**u**n th**i**ng t**o** d**o**." has 8 vowel blocks.

**"aeaeae**nmprt**ouou**." has 2 vowel blocks.

The only vowels which exist are aeiou. Do not consider y as a vowel. Any other letter or symbol is not a vowel.

# Problem 4 - Factorialize

Write a recursive function called dividing\_factorial(num, current=1) which determines the highest n! that divides the number.

Let's explain what we mean by this:

By divisible we mean without remainder (evenly divisible).

For instance:

6 is divisible by 3!, but not 4! = 24, so return 3

18 is divisible by 3! but not 4! again because 18, return 3.

240 is divisible by 5! = 120, return 5.

Considering 1 \* 2 \* 3 \* 4 \* 5 \* 7 = 840, this will be divisible by 5! = 1 \* 2 \* 3 \* 4 \* 5 but not 6! = 1 \* 2 \* 3 \* 4 \* 5 \* 6. You see that there isn't an additional 6 in the factorization. We will return 5 in this case.

So, return the highest n so that n! divides num. Remember that 0! = 1, and 1! = 1, so 1! divides any number.

Use a default parameter current=1 so that the user does not need to enter this parameter.

User input will be limited to positive integers.

# Problem 5 - Decimalize

Write a function **binary\_to\_decimal(binary\_number)** that takes a string for a binary number and output the decimal integer for that number.

Your solution MUST contain a for loop of this form (**power** is an integer variable you define earlier):

**for** i **in** range(power, -1, -1):

You may NOT use any other loops in your function.

The following main block:

**if** \_\_name\_\_ == **'\_\_main\_\_'**:

print(binary\_to\_decimal(**'11001111'**))

print(binary\_to\_decimal(**'00000011'**))

print(binary\_to\_decimal(**'100000000011'**))

print(binary\_to\_decimal(**'1'**))

Will print:

**207**

**3**

**2051**

**1**

# Problem 6 - Diagonalize

Write a function detect\_diagonal(the\_grid, row, col) which will determine if there is the same symbol, whatever it is, at (row, col) position in the\_grid (**which is a 2d list**) on some diagonal emanating out from that central point. Return True

if the symbol is on one of the diagonals (diagonal or anti-diagonal) and False otherwise.

Let's draw some examples:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 |  | x |  |  |  |  |
| 1 |  |  |  |  |  | p |
| 2 | a |  |  | x |  |  |
| 3 |  | \* |  |  |  | 2 |
| 4 |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |

This should return True, if you call detect\_diagonal(the\_grid, 2, 3) because there is an element at (0, 1) which matches.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 |  |  |  |  |  | b |
| 1 |  |  |  |  | a |  |
| 2 | \* | \* |  |  |  |  |
| 3 |  |  |  | \* | c |  |
| 4 |  |  |  |  |  | e |
| 5 |  |  |  |  |  | d |

This should return False, if you call detect\_diagonal(the\_grid, 3, 3) because even though there are other asterisk's at (2, 0) and (2, 1) there are none in any of the red squares, which are the diagonal.

# Problem 7 - ColorSort

Implement one of the sorting algorithms from class, either bubble, selection or quick sort with the following modification. Call the function color\_sort(the\_list)

Each element is now a list (or tuple) with two elements. The first element is the number value that you need to sort. The second is a "color" which is really a string.

In case of a tie, sort the highest priority color forward of the lower priority colors.

Color Hierarchy:

|  |
| --- |
| Gold - Highest |
| Red - Second |
| Blue - Third |
| Green - Fourth |

A sample list to be sorted will be for instance:

[(3, 'red'), (2, 'blue'), (2, 'gold'), (5, 'green'), (17, 'blue')]

Sorted it will be:

[(2, 'gold'), (2, 'blue'), (3, 'red'), (5, 'green'), (17, 'blue')]

If there are all the same color, then it sorts just as a usual list would:

[(27, 'red'), (10, 'red'), (21, 'red'), (46, 'red'), (13, 'red')]

Sorted it is:

[(10, 'red'), (13, 'red'), (21, 'red'), (27, 'red'), (46, 'red')]

Colors will only be in those four categories, and in the list, will always be in lower case.

Do **NOT** use .sort(), which will result in a zero for the problem.