Contents

1 Basic Test Results

```
Starting tests...
Tue Nov 10 21:28:36 IST 2015
ce4597df5838ec172c28767976c63062ea9f0099  -


Archive:  /tmp/bodek.2mL12Q/intro2cs/ex3/elinorperl/presubmission/submission
  inflating: src/README
  inflating: src/ex3.py

Testing README...
Done testing README...

Running presubmit tests...
result_code    ex3    8    1
Done running presubmit tests

Tests completed

Additional notes:

There will be additional tests which will not be published in advance.
```

2 README

```
elinorperl
329577464
Elinor Perl

I discussed the exercise with: labsupport, and Meital Zalcberg.

=============================
=  README for ex3:  =
=============================


usage: python3 Ex3  ex3.py

==================
=  Description:  =
==================
The exercised in ex3 were designed to test our abilities with using loops.
In the exercise, I defined functions that  created a list with user input,
combined seperated strings, calculated averages, created a Boolean function
that return true according to the  cyclic suitability, counted the amount of
elements within the function, returned all the prime numbers in the inputed
number, defined the cartesian product of two lists, and returned a list of
sums for a defined number from within a list.


=============================
=  List of submitted files: =
=============================

README        This file
ex3.py        Contains functions using different types of loops
...


======================
=  Special Comments  =
======================
```

3 ex3.py

```python
def create_list():
    """ Using the input of words from the user, I combined the input
    into one list.
     """
    my_list = input()
    lst = []
    while my_list:
        lst.append(my_list)
        my_list=input()
    return lst


def concat_list(lst_str):
    """Using the function, I combined the separated input
    into one string.
    """
    lst_str1 = ''
    for word in lst_str:
        lst_str1 += str(word)
    return lst_str1


def avr(num_list):
    """ A function that takes the numbers that were input, and
    finds the mean, first by calculating the sum and then, dividing
    the amount of numbers in the series.
    """
    num_sum = 0
    for num in num_list:
        num_sum += num
    average = num_sum/len(num_list)
    return average


def cyclic(lst1, lst2):

    """ Creating my own Boolean function, returning true only
     when both my lists have the same
    elements, disregarding their order.
    """
    if len(lst1) != len(lst2):
        return False
    if lst1 == lst2:
        return True
    """ I defined a constant that would spot the index of lst1 and find
    that number in lst2, I then created a temporary list combining the
    "beginning" of lst1 until the end, and from the start of lst2 until
    the "beginning" of lst1, afterwards comparing the lists to return
    true or false.
    """
    for element in lst1:
        beginning = lst2.index(element)
        temp_list = lst2[beginning:] + lst2[0:beginning]
        if temp_list == lst1:
            return True
        elif temp_list != lst1:
            return False


```

```python
def hist(n, list_num):
    """ I created an empty list. Using the loop and a the function "count"
    I counted the amount of times each number used and added it to the list.
    list_num contains numbers only from 0-(n-1) according to the instructions.
    """
    counting_elements = []
    for index in range(n):
        counting_elements.append(list_num.count(index))
    return counting_elements


def fact(n):
    """I defined a list for prime number collection, starting with the
        first prime number d=2. The function then breaks down the numbers
        by checking if they their divisible have remainders and adds them
        to the prime factor list if they prove suitable.
    """
    prime_factors = []
    d = 2
    while d*d <= n:
        while (n % d) == 0:
            prime_factors.append(d)
            n //= d
        d += 1
    if n > 1:
        prime_factors.append(n)
    return prime_factors


def cart(lst1, lst2):
    """I defined a cartesian product by making a nested loop (loop
    within a loop). The first one running until the length of
    the first list, and the second until the length of the second
    list, using this loops to add on to a new list.
    """
    if len(lst1) == 0 or len(lst2) == 0:
        return []
    cart_prod = []
    for num1 in lst1:
        for num2 in lst2:
            cart_prod.append([num1, num2])
    return cart_prod


def pair(n, num_list):
    """ I created an empty list, using a nested loop I tried adding
    each numbers to check that it would equal to the n that was input,
    returning the pairs that added up. If there were no sums,the function
    returns "None"
    *It was assumed that the the elements in num_list are different and
    all in fact numbers
    """
    added_sums = []
    for index in range(len(num_list)):
        for placement in range(index+1, len(num_list)):
            if num_list[index] + num_list[placement] == n:
                pair1 = [(num_list[index]), (num_list[placement])]
                added_sums.append(pair1)
    if not added_sums:
        return None
    return added_sums
```