İbrahim Talha ASAN
COE-64170019

Assignment 1 Report

Problem 1:

Definition: The goal in Problem 1 is that for k values less than 32, finding prime numbers among the numbers that satisfy the "x = y" equation and printing them on the screen.

Program Code:

```
def determine_prime(pn):#taking pn as parameter, defining function.
  if pn <= 1:#if pn is 0 or 1 and not pn is not prime number.
    return False
  for i in range(2, pn):
    if pn % i == 0:
      return False
  else :
    return True
for k in range(32):#generates k from 1 to 32.
  pn = 3**k - 2**k
  if determine_prime(pn):#calling the "determine_prime()" function.
    try:
      print(pn)
    except AssertionError as error:
      print(error)
```

Program Outputs:

```
5

19

211

129009091
```

Discussions: I used the "try-except" construct to find the reason for printing the remaining elements. But I could not reach any result. I think the problem is that the capacity of the int data type is exceeded and the remaining values cannot be calculated. (I waited for 5 -6 minutes.)

İbrahim Talha ASAN
COE-64170019

Problem 2:

Definition: The goal in Problem 2 is, using nested loops to obtain the Pattern A and Pattern B given in the question.

Program Code:

```
#Pattern A
number_of_rows=5
for i in range(1,number_of_rows+1):#generates the number of rows the pattern A has.
  number=2
  for j in range(1,number_of_rows+1-i):#print the  1 in the mid-side.
    print('',end = '  ')#set the space
  for k in range(i,0,-1):#print the left-side triangle.
    print(pow(3,k-1),end=' ')#set the space
  for x in range(2,i+1):#print the right-side triangle.
    print(number,end=' ')#set the space
    number=number*2
  print()
print("\n")
#Pattern B
number_of_rows = 6
for i in range(number_of_rows, 1, -1):#generates the number of rows the pattern A has.

  for j in range(1, i - 1):#print the left-side upper triangle.
   print(j, end=" ")

  for k in range(i - 1, 0, -1):#print the right-side upper triangle.
    print(k, end=" ")

  print()
```

Program Outputs:

Pattern A:

```
      1
    3 1 2
  9 3 1 2 4
 27 9 3 1 2 4 8
81 27 9 3 1 2 4 8 16
```

Pattern B:

```
1 2 3 4 5 4 3 2 1
1 2 3 4 3 2 1
1 2 3 2 1
1 2 1
1
```

İbrahim Talha ASAN
COE-64170019

Discussions: The (1) s, which should be in the middle of Pattern A, should have been placed correctly. However, I could not do the positioning of the (1) correctly. The reason for this problem is probably due to the number of gaps given and although I tried too many times, I could not find a solution to the problem. There is no problem with Pattern B.

Problem 3:

Definition: The goal in Problem 3 is to create a function called "find_consecutives ()" that takes a tuple of int values as parameters. The task of this function is to find the consecutive elements in the tuple given to it.

Program Code:

```
def find_consecutives(tup): #taking tuple as parameter, defining function.
   elements = iter(tup)#"iter ()" function returns an iterator in tuple(with this iterator ,can access the
   #elements in the tuple sequentially.)and assigned elements.
   x,y=None,next(elements,None)#declaring x and y  with None in elements.
   consecutives=[y]#consecutive ones are put in "consecutives".
   while y is not None:
      x,y=y,next(elements,None)
      if y is not None and x + 1==y: #It is checked whether the numbers are consecutive or not.
         consecutives.append(y)#If the numbers are consecutive, they are added to "consecutives".
      else:
         if len(consecutives) > 1:
            yield list(consecutives)
         consecutives=[y]
print(list(find_consecutives((2,8,4,6,1,2,8,4,7,9,5,6,7))))
```

Program Outputs:

```
[[1, 2], [5, 6, 7]]
```

Discussions: There is no problem with Problem 3.