wrdks82y8

December 20, 2024

[ ]:
```

```

[ ]:
```
#if-elif-else
'''
if condition:
  do something
elif condition2:
  do this
else:
  do this
```

[ ]:
```
n = int(input())
print(n)
```

```
3
3
```

[ ]:
```
if n%2 ==0:
  print('even')
else:
  print('odd')
```

```
odd
```

[ ]:
```
'''
given a number ,
print 'foo' if it is a multiple of 3
print 'bar; if it is amultiple of 5
print 'foobar;if it s multiple of both 3 and 5.
'''
n= int(input())
```

```
15
```

[ ]:
```
if (n%3 == 0) and (n%5 == 0):
  print('foobar')
elif(n%3 ==0):
  print('foo')
```

```
elif(n%5 == 0):
  print('bar')
else:
  print('nothing')
```

```
foobar
```

[ ]:
```
#loops
#while loop ->print all the numbers from 1 to 10 (both inlcusive)
#initilaiation

#while confition:
 #code
 #updation
```

[ ]:
```
i=1
while i <10:
  print(i, end ="#")
  i += 1
```

```
1#2#3#4#5#6#7#8#9#
```

[ ]:
```
i=1
while i <10:
  print(i, end =" ")
  i += 1
```

```
1 2 3 4 5 6 7 8 9
```

[ ]:
```
#for loop
#exclusing last value.
print(list(range(10)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[ ]:
```
print(list(range(3,10)))
```

```
[3, 4, 5, 6, 7, 8, 9]
```

[ ]:
```
print(list(range(3,10,2)))
```

```
[3, 5, 7, 9]
```

[ ]:
```
for i in range(1,11):
  print(i, end = " " )
```

```
1 2 3 4 5 6 7 8 9 10
```

[ ]:
```python
for i in range(10,0,-1):
  print(i, end = " " )
```

```
10 9 8 7 6 5 4 3 2 1
```

[ ]:
```python
for i in range(1,10,3):
  print(i, end = " " )
```

```
1 4 7
```

”’ list ”’

[ ]:
```python
#create a list
a= [3,45,6,1,10,2]
type(a)
```

[ ]:
```
list
```

[ ]:
```python
print(a[0])
```

```
3
```

[ ]:
```python
print(a[3])
```

```
1
```

[ ]:
```python
print(a[-1])
```

```
2
```

[ ]:
```python
print(a[-3])
```

```
1
```

[ ]:
```python
#slicing
a[1:3]
```

[ ]:
```
[45, 6]
```

[ ]:
```python
a[:3]
```

[ ]:
```
[3, 45, 6]
```

[ ]:
```python
a[1:]
```

[ ]:
```
[45, 6, 1, 10, 2]
```

[ ]:
```python
a[:]
```

```
[ ]: [3, 45, 6, 1, 10, 2]
```

```
[ ]: a[1:4:2]
```

```
[ ]: [45, 1]
```

```
[ ]: a[-1:-4:-1]
```

```
[ ]: [2, 10, 1]
```

```
[ ]: a[::-1]
```

```
[ ]: [2, 10, 1, 6, 45, 3]
```

```
[ ]: a =[45,67,89,12,34,56,100]
     #create list of all the number  of even indexes
     ans = []
     n =len(a)
     print(n)
```

```
7
```

```
[ ]: ans =[]
     for i in range(0,n):
       if i%2 ==0:
         ans.append(a[i])

     #print(ans)
```

```
[ ]: print(ans)
```

```
[45, 89, 34, 100]
```

```
[ ]: #list comprehension

     a = [3,6,1,4]
     ans = []
     n =len(a)
     for i in range(n):
       ans.append(a[i]**2)
```

```
[ ]: print(ans)
```

```
[9, 36, 1, 16]
```

```
[ ]: for elem in a:
       print(elem, end =" ")
```

```
3 6 1 4
```

[ ]:
```
n = len(a)
ans =[a[i]**2 for i in range(n)]
```

[ ]:
```
print(ans)
```

```
[9, 36, 1, 16]
```

[ ]:
```
#squatre the elelments present at even index and dont do anything for the␣
  ↪element
n = len(a)
ans =[a[i]**2 for i in range(n) if i%2 == 0]
print(ans)
```

```
[9, 1]
```

[ ]:
```
#squatre the elelments present at even index and dont do anything for the␣
  ↪element
n = len(a)
ans =[a[i]**2 if i%2 == 0 else a[i] for i in range(n)]
print(ans)
```

```
[9, 6, 1, 4]
```

[ ]:
```
#squatre the elelments present at even index and dont do anything for the␣
  ↪element
a =[3,6,2,5]
n = len(a)
ans =[a[i]**2 if i%2 == 0 else a[i] for i in range(n)]
print(ans)
```

```
[9, 6, 4, 5]
```

[ ]:
```
#list iof lists
a = [[1,2,3],[5,6,7],[2,3,4]]
print(a)
```

```
[[1, 2, 3], [5, 6, 7], [2, 3, 4]]
```

[ ]:
```
print(a[1][2])
```

```
7
```

[ ]:
```
n = len(a)
for i in range(n):
  m =len(a[i])
  for j in range(m):
    print(a[i][j], end = " ")
  print()
```

```
1 2 3
5 6 7
2 3 4
```

[ ]:
```
for row in a:
  for elem in row:
    print(elem, end = " ")
  print()
```

```
1 2 3
5 6 7
2 3 4
```

[ ]:
```
ans = [elem for row in a for elem in row]
print(ans)
```

```
[1, 2, 3, 5, 6, 7, 2, 3, 4]
```

[ ]:
```
#tuple an d listy only one difference ->tuple are immutable

a =(4,1,2,3)
print(a)
```

```
(4, 1, 2, 3)
```

[ ]:
```
type(a)
```

[ ]:
```
tuple
```

[ ]:
```
#similar to list also indexes from 0

b = [1,5,7,3]
b[1] =10
print(b)
```

```
[1, 10, 7, 3]
```

[ ]:
```
#packing un packing

tup = 1,2,4,3
print(tup)
```

```
(1, 2, 4, 3)
```

[ ]:
```
#un packing
t = (5,7,2)
a,b,c =t
print(a)
```

```
5
```

[ ]:
```
a = [('a',1),('b',2),('c',3)]
type(a)
```

[ ]:
```
list
```

[ ]:
```
type(a[1])
```

[ ]:
```
tuple
```

[ ]:
```
a[1]
```

[ ]:
```
('b', 2)
```

[ ]:
```
for elem in a:
  print(elem)
```

```
('a', 1)
('b', 2)
('c', 3)
```

[ ]:
```
for elem in a:
  i, j =elem
  print(i,j)
```

```
a 1
b 2
c 3
```

[ ]:
```
#intitialize in single value
a =(2)
type(a)
```

[ ]:
```
int
```

[ ]:
```
a =(2,)
type(a)
```

[ ]:
```
tuple
```

[ ]:
```
#set
a =set()
print(a)
```

```
set()
```

[ ]:
```

```

[ ]:
```
a = {1,2,5,2,3,3,4,1,6}
print(a)
```

```
{1, 2, 3, 4, 5, 6}
```

[ ]:
```
type(a)
```

[ ]:
```
set
```

[ ]:
```
a1 = {2,3,4,5}
a2 ={3,4,1,6,7,9}
type(a1)
```

[ ]:
```
set
```

[ ]:
```
print(a1.intersection(a2))
```

```
{3, 4}
```

[ ]:
```
print(a1.union(a2))
```

```
{1, 2, 3, 4, 5, 6, 7, 9}
```

[ ]:
```
print(a1 |a2)
```

```
{1, 2, 3, 4, 5, 6, 7, 9}
```

[ ]:
```
print(a1-a2)# ellemtsn present in a1 but not a2
```

```
{2, 5}
```

[ ]:
```
print(a2-a1)
```

```
{1, 9, 6, 7}
```

[ ]:
```
print(a1 ^ a2) #union of above two
```

```
{1, 2, 5, 6, 7, 9}
```

[ ]:
```
#empty dictionary
d = {}
type(d)
```

[ ]:
```
dict
```

[ ]:
```
student = {"name":"aditya","city":"nagpur","state":"maharastra"}
type(student)
```

[ ]:
```
dict
```

[ ]:
```
student["name"]
```

[ ]:
```
'aditya'
```

[ ]:
```
student["city"]
```

[ ]:
```
'nagpur'
```

[ ]:
```
print(student)
```

```
{'name': 'aditya', 'city': 'nagpur', 'state': 'maharastra'}
```

”’ 1.03 ”’

[ ]:
```
#keys
#values
#items - (key,value)
```

[ ]:
```
for k in student.keys():
  print(k)
```

```
name
city
state
```

[ ]:
```
for v in student.values():
  print(v)
```

```
aditya
nagpur
maharastra
```

[ ]:
```
for i in student.items():
  print(i)
```

```
('name', 'aditya')
('city', 'nagpur')
('state', 'maharastra')
```

```
[ ]: for elem in student.items():
       #unpacking
       k,v = elem
       print(f"key is {k} and the value is {v}")
```

```
key is name and the value is aditya
key is city and the value is nagpur
key is state and the value is maharastra
```

```
[ ]: freq = {
         "a" :14,
         "b":30,
         "z":31,
         "k":10,
         "j":4
     }
```

```
[ ]: max_freq = -1
     max_alpha = None
     for elem in freq:
       #unpacking
       alpha,freq = elem
       if freq > max_freq:
         max_freq = freq
         max_alpha =alpha

     print(f"the character {max_alpha} is having the highest frequency of␣
      ↪{max_freq}")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-10545fe797c3> in <cell line: 3>()
      1 max_freq = -1
      2 max_alpha = None
----> 3 for elem in freq:
      4   #unpacking
      5   alpha,freq = elem

NameError: name 'freq' is not defined
```

```
[ ]: #function

     def check_even_odd(n):
       if n%2 == 0:
         return "even"
       else:
```

```
    return "odd"
check_even_odd(2)
```

[ ]: 'even'

[ ]:
```
ans = check_even_odd(7)
print(ans)
```

```
odd
```

[ ]:
```
def solve(a,b,c,d):
  print(a,b,c,d)
solve(3,2,5,1)
```

```
3 2 5 1
```

[ ]:
```
solve(3,2,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-f6e040f2909e> in <cell line: 1>()
----> 1 solve(3,2,5)

TypeError: solve() missing 1 required positional argument: 'd'
```

[ ]:
```
solve(a=1,b=3,c=2,d=10)
```

```
1 3 2 10
```

[ ]:
```
solve(b=3,a=1,c=2,d=10)
```

```
1 3 2 10
```

[ ]:
```
def solve(a,b,c=2):
  print(a,b,c)
```

[ ]:
```
solve(1,3)
```

```
1 3 2
```

[ ]:
```
solve(1,3,5)
```

```
1 3 5
```

[ ]:
```
def solve(a,b,c,e):
  print(a,b,c,e)
solve(1,2,e=4,c=3)
```

```
1 2 3 4
```

[ ]:
```
solve(1,2,e=4,c=3)
```

```
1 2 3 4
```

Python 1 scaler

#flow 1->specific goal,accoutability,

[ ]:
```
flow 2-> task should be right difficulty
```

#flow 3 -> feedback

flow 4 -> deep focus => switch off your phone

#complete the python 1 all codes in jupyter notebook

[ ]:
```
#q1
def main():
    # YOUR CODE GOES HERE
    # Please take input and print output to standard input/output (stdin/stdout)
    # E.g. 'input()/raw_input()' for input & 'print' for output

     # Take input from the user
    name = input().strip()#rempove whote spaces

    # Check if the length of the name is within the specified constraints
    if 1 <= len(name) <= 15 :
        # Print the desired output
        print("Hello", name)
    else:
        # If constraints are not met, print an error message
        print("Invalid input. Please enter a lowercase name with length between␣
 ↪1 and 15 characters.")


    return 0

if __name__ == '__main__':
    main()
```

```
 heloo rgv jkdcj
Hello heloo rgv jkdcj
```

[ ]:
```
#Q2
# write your code here
# Perform the operations
addition_result = 6 + 3
subtraction_result = 6 - 3
```

```
multiplication_result = 6 * 3
division_result = 6 / 3

# Print the results in separate lines
print(addition_result)
print(subtraction_result)
print(multiplication_result)
print(division_result)
```

```
9
3
18
2.0
```

[ ]:
```
#Q3

print("A")
print("B")
print("C")
print("D")
print("E")
```

```
A
B
C
D
E
```

[ ]:
```
#Q4
total_savings = int(input())
amount_spent = int(input())

remaining_savings = total_savings - amount_spent


print(remaining_savings)
```

```
116
12
104
```

[ ]:
```
#Q5
A = input().strip()
B = input().strip()


print(f"{A} says Hi to {B}")
```

```
RAM
SYAM
RAM says Hi to SYAM
```

#OPERATORS

```
[ ]: #Q1
     result = (3 + 4) // 2 + 6
     print(result)
```

```
9
```

```
[ ]: #Q2
     a = -1
     b = 0
     c = 1

     # DO NOT CHANGE THIS
     x = a < b + c

     print(x) # this should be True
```

```
True
```

```
[ ]: #Q3
     import math

     #integer
     total_budget = float(input().strip())

     #integer
     single_bill_value = int(input().strip())

     #floor down to integer
     number_of_bills = math.floor(total_budget / single_bill_value)


     print(number_of_bills)
```

```
126.3
5
25
```

```
[ ]: #q4
     from builtins import input

     varun_mobile_number = 1234880990
```

```
def is_spammer(input_mobile_number):
    return input_mobile_number < varun_mobile_number

# Accept mobile number as input
input_mobile_number = int(input(""))

# Check if the mobile number belongs to spammers
result = is_spammer(input_mobile_number)

# Print the result
print(result)
```

```
123488099
True
```

[ ]:
```
#q5
print("54" + "23")
```

```
5423
```

[ ]:
```
#Control statements
```

[ ]:
```
#Q1
# Function to check if the triangle is valid
def is_triangle_valid(A, B, C):
    # Sum of angles should be 180 for a valid triangle
    if A + B + C == 180:
        return 1  # Valid triangle
    else:
        return 0  # Invalid triangle

# Input angles from the user
A = int(input())
B = int(input())
C = int(input())

# Check and print the result
result = is_triangle_valid(A, B, C)
print(result)
```

```
60
60
60
1
```

[ ]:
```
#Q2
# Function to determine profit or loss and calculate total profit or loss
```

```python
def calculate_profit_loss(cp, sp):
    if cp < sp:
        # Profit case
        return 1, sp - cp
    else:
        # Loss case
        return -1, cp - sp

# Input
cp = int(input())
sp = int(input())

# Calculate profit or loss and total profit or loss
result, total_amount = calculate_profit_loss(cp, sp)

# Output
print(result)
print(total_amount)
```

```
2
4
1
2
```

[ ]:
```python
#Q3
# Input the two numbers
A = int(input())
B = int(input())

# Find and print the maximum element among A and B
if A > B:
    print(A)
else:
    print(B)
```

```
10
100
100
```

#Q4 a. if a>=2: print("TRUE"): This statement is valid. It checks if the value of a is greater than or equal to 2, and if so, it prints "TRUE".

b. if (a=>2): print("TRUE"): This statement is invalid. In Python, there's no operator =>. To check if a is greater than or equal to 2, you should use >=. So, the correct syntax would be if a >= 2:.

c. if (a%2!=0): print("TRUE"): This statement is valid. It checks if the remainder of a divided by 2 is not equal to 0, which means it's checking if a is odd. If so, it prints "TRUE".

d. if a//3=1: print("TRUE"): This statement is invalid. The = operator is used for assignment, not for comparison. To check if a divided by 3 is equal to 1, you should use == for comparison. So, the correct syntax would be if a//3 == 1:.

[ ]:
```
#Q5
# Get the age as input from the user
age = int(input(""))

# Check if the user is old enough to ride the roller coaster
if age >= 13:
    print("You can ride the roller coaster!")
else:
    print("You can't ride the roller coaster.")
```

```
12
You can't ride the roller coaster.
```

#Loops-While and For

[ ]:
```
#q1
n = 5  # Change n to any desired value

i = 2  # Initialize i with the first even number

while (i <= 2 * n):  # Loop until we find 2*n even numbers
    if i % 2 == 0:  # Check if i is even
        print(i)  # Print the even number
    i += 1  # Move to the next number
```

```
2
4
6
8
10
```

[ ]:
```
#q2
def main():
    # Obtain user input for N
    N = int(input(""))

    # Initialize i as 1
    i = 1

    # Loop through numbers from 1 to N
    while i <= N:
        # Check if the number is even
        if i % 2 == 0:
            # Print the even number without a newline
            print(i, end=" ")
        # Increment i by 1
        i += 1

    # Add a newline character at the end
    #print()

if __name__ == '__main__':
    main()
```

```
10
2 4 6 8 10
```

#print() doudt

[ ]:
```
#q3
def main():
    # Obtain user input for N
    N = int(input(""))

    # Initialize i as 1
    i = 1

    # Loop through numbers from 1 to N
    while i <= N:
        # Check if the number is even
        if i*i <= N:
            # Print the even number without a newline
            print(i*i, end=" ")

        # Increment i by 1
        i += 1

    # Add a newline character at the end
    print()

if __name__ == '__main__':
    main()
```

```
20
1 4 9 16
```

[ ]:
```
#q4
for i in range(-6, -10, -1):
     print(i, end =" ")
```

```
-6 -7 -8 -9
```

[ ]:
```
#q5
def main():
    # Read input from the user
    N = int(input(""))

    # Initialize sum variable
    ans = 0

    # Calculate the sum of natural numbers from 1 to N
    for i in range(1, N+1):
        ans += i

    # Print the result
    print(ans)

# Call the main function
main()
```

```
5
15
```

#nested loops

[ ]:
```
#Q1 reverse of number
def main():
    # Take input for the number of test cases
    tc = int(input())

    # Process each test case
    while tc > 0:
        # Decrease the count of remaining test cases
        tc -= 1

        # Take input for the number to be reversed
        number = int(input())

        # Initialize variable to store reversed number
        revs_number = 0

        # Reverse the digits of the number
        #memeorize.
        while number > 0:
            revs_number = (revs_number * 10) + number % 10
            number = number // 10

        # Print the reversed number
        print(revs_number)
```

```
if __name__ == '__main__':
    main()
```

```
3
101
101
501
105
522
225
```

[ ]:
```
#Q4
def calculate_hcf(a, b):
    while b:
        a, b = b, a % b
    return a

def main():
    t = int(input())

    for _ in range(t):
        a = int(input())
        b = int(input())
        hcf = calculate_hcf(a, b)
        print(hcf)

if __name__ == '__main__':
    main()
```

```
1
15
105
15
```

[ ]:
```
#Q5
def main():
    t = int(input())  # Number of test cases

    for _ in range(t):
        A = int(input())  # Input first number
        B = int(input())  # Input second number

        G = 1  # Initialize GCD as 1

        # Find the GCD (Greatest Common Divisor)
        #memorize this part.
        for i in range(1, min(A, B) + 1):
            if A % i == 0 and B % i == 0:
                G = i

        # Calculate LCM (Least Common Multiple)
        L = (A * B) // G

        print(L)  # Print LCM for the current test case

if __name__ == '__main__':
    main()
```

```
2
6
8
24
10
12
60
```

[ ]:
```
#functions
#q1
def celsius_farhen(Celsius) :
    Fahrenheit = ((9/5)*Celsius) + 32
    ans = round(Fahrenheit ,2)
    return ans

celsius_farhen(25)
```

[ ]:
```
77.0
```

[ ]:
```
#2
def road_tax(price):
    tax = None
    # use if else to check price range
    if price > 100000:
        tax = 20/100*price
    elif price >75000 and price <=100000:
        tax = 15/100*price
    elif price >50000 and price <=75000:
        tax = 10/100*price
    else:
        tax = 5/100*price
    tax=round(tax,1)
    return tax

road_tax(85000)
```

```
[ ]: 12750.0
```

```
[ ]: #q3
     def sum_squares(n):
         ans = n * (n + 1) * (2 * n + 1) // 6
         return ans

     sum_squares(3)
```

```
[ ]: 14
```

```
[ ]: #q4

     def Interest(p,c,t=2,r=0.09):
       return p*t*r

     The correct options are: Interest(p=1000,c=5) and Interest(c=4,r=0.12,p=5000)

     Explanation:

     For a functions valid call, all the parameters that are not defined by default␣
       ↪should have a value.
     In Interest(p=1000,c=5), all parameters have a value.
     In Interest(r=0.05,5000,3), we assign the value of a keyword parameter before␣
       ↪positional arguments, hence not allowed.
     In Interest(500,t=2,r=0.05), the variable ‘c’ is not assigned any value, hence␣
       ↪not allowed.
     In Interest(c=4,r=0.12,p=5000), all arguments have an assigned value, hence␣
       ↪valid.
```

```
[ ]: x = "global"
     def foo():
         global x
         print(x, end=",")
         x = "local"
         print(x, end=",")

     foo()
     print(x, end="")

     The correct answer is: global,local,local

     Explanation

     The first call to the print function that gets executed is the first one inside␣
       ↪of the foo function.
```

```
This call prints "global," because at that execution point in the program, x␣
  ↪has only been defined as "global" in the global scope.
However, after this first print statement, we assign x to the new value of␣
  ↪"local", meaning that when x is printed again right after, we print "local".
We do not create a new x in the local scope, instead we change the value of x␣
  ↪in the global scope.
Finally, the last call to print takes in the x defined in the global scope,␣
  ↪which was modified in the foo function because of the presence of the global␣
  ↪keyword.
Therefore, this call prints "local".
```

[ ]:
```
#List 1
def even_odd(A):
    even_sum = 0
    odd_sum = 0
    for x in A:
        # % operator will give remainder, use that to check number is even or␣
  ↪not
        if x % 2 == 0:
            even_sum = even_sum + x
        else:
            odd_sum = odd_sum + x
    return even_sum - odd_sum
```

[ ]:
```
#Q5,q3 in lists
append,extend,sort
```

[ ]:
```

```