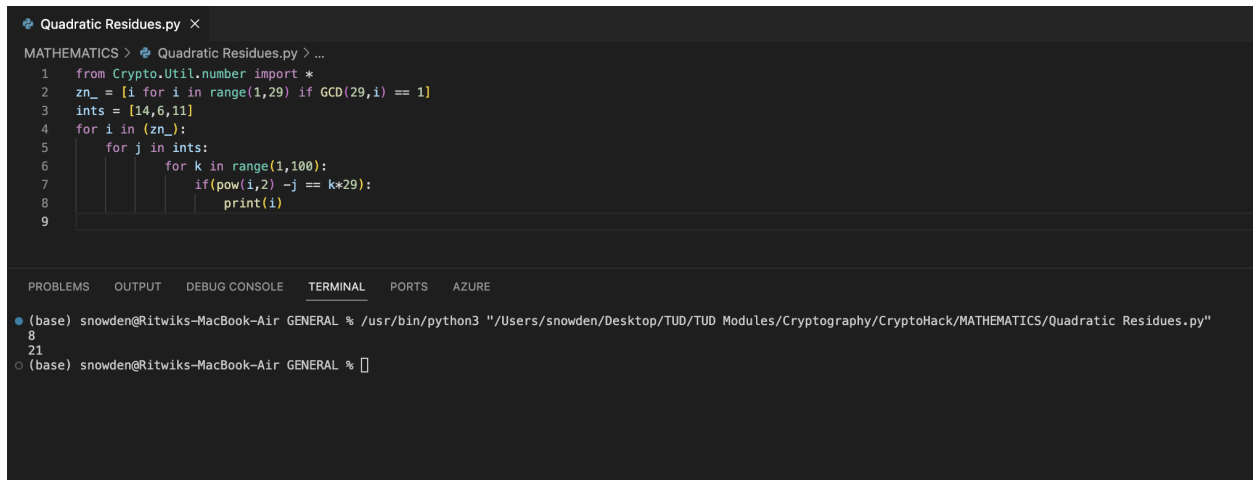


# MATHEMATICS

## Quadratic Residues



```
Quadratic Residues.py X
MATHEMATICS > Quadratic Residues.py > ...
1 from Crypto.Util.number import *
2 zn_ = [i for i in range(1,29) if GCD(29,i) == 1]
3 ints = [14,6,11]
4 for i in (zn_):
5     for j in ints:
6         for k in range(1,100):
7             if (pow(i,2) -j) == k*29:
8                 print(i)
9
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
● (base) snowden@Ritwicks-MacBook-Air GENERAL % /usr/bin/python3 "/Users/snowden/Desktop/TUD/TUD Modules/Cryptography/CryptoHack/MATHEMATICS/Quadratic Residues.py"
8
21
○ (base) snowden@Ritwicks-MacBook-Air GENERAL %
```

## Legendre Symbol

```
Quadratic Residues.py  Legenre_symbol.py 1 X
MATHEMATICS > Legenre_symbol.py > modular_sqrt
1 import math
2
3 def modular_sqrt(a, p):
4     if legendre_symbol(a, p) != 1:
5         return 0
6     elif a == 0:
7         return 0
8     elif p == 2:
9         return 0
10    elif p % 4 == 3:
11        return pow(a, (p + 1) // 4, p)
12
13    s = p - 1
14    e = 0
15
16    while s % 2 == 0:
17        s /= 2
18        e += 1
19    n = 2
20
21    while legendre_symbol(n, p) != -1:
22        n += 1
23
24    x = pow(a, (s + 1) / 2, p)
25    b = pow(a, s, p)
26    g = pow(n, s, p)
27    r = e
28
29    while True:
30        t = b
31        m = 0
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
(base) snowden@Ritwicks-MacBook-Air GENERAL % /usr/bin/python3 "/Users/snowden/Desktop/TUD/TUD Modules/Cryptography/CryptoHack/MATHEMATICS/Legenre_symbol.py"
1
93291799125366706806545638475797430512104976066103610269938025709952247020061090804870186195285998727680200979853848718589126765742550855954805290253592144209552123062
161458584575060939481368210688629862036958857604707468372384278049741369153506182660264876115428251983455344219194133033177700490981696141526
```

## Chinese Remainder Theorem

```
Chinese Remainder Theorem.py X
MATHEMATICS > Chinese Remainder Theorem.py > ...
1 def chinese_remainder_theorem(moduli, remainders):
2     def modinv(a, m):
3         m0, x0, x1 = m, 0, 1
4         while a > 1:
5             q = a // m
6             m, a = a % m, m
7             x0, x1 = x1 - q * x0, x0
8         return x1 + m0 if x1 < 0 else x1
9
10    product = 1
11    for m in moduli:
12        product *= m
13
14    result = 0
15    for mi, ai in zip(moduli, remainders):
16        bi = product // mi
17        result += ai * modinv(bi, mi) * bi
18
19    return result % product
20
21    # Example usage
22    moduli = [5, 11, 17]
23    remainders = [2, 3, 5]
24
25    result = chinese_remainder_theorem(moduli, remainders)
26    print("Flag: ", result)
27

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE
● (base) snowden@Ritwicks-MacBook-Air GENERAL % /usr/bin/python3 "/Users/snowden/Desktop/TUD/TUD Modules/Cryptography/CryptoHack/MATH
Flag: 872
○ (base) snowden@Ritwicks-MacBook-Air GENERAL %
```

# LATTICES

## Gram Schmidt

```
Gram Schmidt.py X
MATHEMATICS > Gram Schmidt.py > ...
1  from math import sqrt
2
3  def dot_product(v1, v2):
4      return sum(a*b for a, b in zip(v1, v2))
5
6  def vector_norm(v):
7      return sqrt(dot_product(v, v))
8
9  vectors = [[4, 1, 3, -1], [2, 1, -3, 4], [1, 0, -2, 7], [6, 2, 9, -5]]
10 # Gram smith algorithm implementation
11 def gram_smith(vectors):
12     u = []
13     for i in range(len(vectors)):
14         ui = vectors[i]
15         for j in range(i):
16             muj = dot_product(vectors[i], u[j]) / vector_norm(u[j])**2
17             ui = [ui[k] - muj * u[j][k] for k in range(len(ui))]
18         u.append(ui)
19     return u
20
21 flag = round(gram_smith(vectors)[3][1],5)
22 print('FLAG: ',flag)
23
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

```
● (base) snowden@Ritwicks-MacBook-Air GENERAL % /usr/bin/python3 "/Users/snowden/Desktop/TUD/TUD Modules/Cryptography/CryptoHack/MATHEMATICS/Gram Schmidt.py"
FLAG: 0.91611
○ (base) snowden@Ritwicks-MacBook-Air GENERAL %
```