Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> def a_pow_n(a,n):
        #return a^n, where a:base,n:power
        y,x=1,a
        while n>0:
                if n%2 ==1:
                        y=y*x
                x=x*x
                n=n//2
        return y

>>> a_pow_n(1013,419)
22405653013340406314151345781876843560122551559500779953590249754235318376624469
03105694108693673163702643664991815403766897275327722317342407719078641524809289
79353868361759455888851516001166575493240297955745240659499754995331761683740706
29673220949559109020675154620486237964644007536842958954443581757883492731492132
54715824741381734737735014037730861652768113183372636644505565439147530182945832
60603863466885922903189232958848729458689482297985676410157391249495880375006104
69921990289565552072941679472396360550124009274671784556459084489081663109245640
66151056165135695034970394183617048406461682607078078673058816509548798431682899
97986443419537742204499583104903972358007395431622523562190501252959332129182097
42527958652709544931784310887341902389872492121623113812721823882638503558919426
10376360649454951147740983253494772614794597893059113792752868986732418911264842
69817249257278352891415904074345576878139248871230490110609506286174253787368019
03200766011873541103553144398165556809904973436867322358259481442379286606216432
34520708683411314469492042932224994628853858142100998251274010661313608316059670
61349537534315680600507650500539428371617728268388951576542561894061835493660626
62661168915970888112500105300054922374128028267049395471367 7
>>>
```

1번-방법1

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> def pow(a, n):
    if n < 0:
        return pow(1 / a, -n)
    elif n == 0:
        return 1
    elif n == 1:
        return a
    elif n % 2 == 0:
        return pow(a * a, n / 2)
    else:
        return a * pow(a * a, (n - 1) / 2)

>>> pow(1013,419)
22405653013340406314151345781876843560122551559500779953590249754235318376624469
03105694108693673163702643664991815403766897275327722317342407719078641524809289
79353868361759455888851516001166575493240297955745240659499754995331761683740706
29673220949559109020675154620486237964644007536842958954443581757883492731492132
54715824741381734737735014037730861652768113183372636644505565439147530182945832
60603863466885922903189232958848729458689482297985676410157391249495880375006104
69921990289565552072941679472396360550124009274671784556459084489081663109245640
66151056165135695034970394183617048406461682607078078673058816509548798431682899
97986443419537742204499583104903972358007395431622523562190501252959332129182097
42527958652709544931784310887341902389872492121623113812721823882638503558919426
10376360649454951147740983253494772614794597893059113792752868986732418911264842
69817249257278352891415904074345576878139248871230490110609506286174253787368019
03200766011873541103553144398165556809904973436867322358259481442379286606216432
34520708683411314469492042932224994628853858142100998251274010661313608316059670
61349537534315680600507650500539428371617728268388951576542561894061835493660626
62661168915970888112500105300054922374128028267049395471367 7
```

1번-방법2

>>>

Ln: 17 Col: 4

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> def return_factorizer(n):
        for i in range (1,n+1):
                if n%i==0:
                        print(i)


>>> return_factorizer(231358069909)
1
480787
481207
```

2번-방법1

이 알고리즘은 변수를 증가시키며 나누어 떨어지는 수만을 print 해주는 방법입니다.

따라서, 이처럼 큰 소수 두개의 곱으로 이루어진 숫자는 소인수를 찾아낼 수 있어도,소인수분해가 아닌 약수찾기에 더 가깝습니다.

그래서 다른 방법도 생각해보았습니다.

Ln: 11 Col: 0

Python 3.8.2 Shell - C:/Users/smk38/Desktop/fjkdfjk.py (3.8.2)

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> def primeSieve(sieveSize):
        # creating Sieve (0~n까지의 slot)
        sieve = [True] * (sieveSize+1)
        # 0과 1은 소수가 아니므로 제외
        sieve[0] = False
        sieve[1] = False
        # 2부터 (루트 n) + 1까지의 숫자를 탐색
        for i in range(2,int(math.sqrt(sieveSize))+1):
                # i가 소수가 아니면 pass
                if sieve[i] == False:
                        continue
                # i가 소수라면 i*i~n까지 숫자 가운데 i의 배수를
                # 소수에서 제외
                for pointer in range(i**2, sieveSize+1, i):
                        sieve[pointer] = False
        primes = []
        # sieve 리스트에서 True인 것이 소수이므로
        # True인 값의 인덱스를 결과로 저장
        for i in range(sieveSize+1):
                if sieve[i] == True:
                        primes.append(i)
        return primes

>>> def get_prime_factors(n):
        # n 범위 내의 소수를 구한다
        primelist = primeSieve(n)
        # 이 소수들 중 n으로 나누어 떨어지는
        # 소수를 구하고, 몇 번 나눌 수 있는지 계산
        # 예 : n = 8, factors = [(2, 3)]
        # 예 : n = 100, fcount = [(2: 2), (5: 2)]
        factors = []
        for p in primelist:
                count = 0
                while n % p == 0:
                        n /= p
                        count += 1
                if count > 0:
                        factors.append((p, count))
        return factors
```

2번-방법2

먼저, 에라토스테네스의 체를 이용해 구하고자 하는 숫자보다 작은 모든 소수를 구해줍니다.

그 다음, 이 소수들 중 나누어 떨어지는 소수를 구한뒤 몇번 나눌 수 있는지 계산하고, 이를 리스트로 나타내면 됩니다.

Python 3.8.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> def primeSieve(sieveSize):
        # creating Sieve (0~n까지의 slot)
        sieve = [True] * (sieveSize+1)
        # 0과 1은 소수가 아니므로 제외
        sieve[0] = False
        sieve[1] = False
        # 2부터 (루트 n) + 1까지의 숫자를 탐색
        for i in range(2,float(math.sqrt(sieveSize))+1):
                # i가 소수가 아니면 pass
                if sieve[i] == False:
                        continue
                # i가 소수라면 i*i~n까지 숫자 가운데 i의 배수를
                # 소수에서 제외
                for pointer in range(i**2, sieveSize+1, i):
                        sieve[pointer] = False
        primes = []
        # sieve 리스트에서 True인 것이 소수이므로
        # True인 값의 인덱스를 결과로 저장
        for i in range(sieveSize+1):
                if sieve[i] == True:
                        primes.append(i)
        return primes

>>> def get_prime_factors(n):
        # n 범위 내의 소수를 구한다
        primelist = primeSieve(n)
        # 이 소수들 중 n으로 나누어 떨어지는
        # 소수를 구하고, 몇 번 나눌 수 있는지 계산
        # 예 : n = 8, factors = [(2, 3)]
        # 예 : n = 100, fcount = [(2: 2), (5: 2)]
        factors = []
        for p in primelist:
                count = 0
                while n % p == 0:
                        n /= p
                        count += 1
                if count > 0:
                        factors.append((p, count))
        return factors

>>> get_prime_factors(231358069909)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    get_prime_factors(231358069909)
  File "<pyshell#4>", line 3, in get_prime_factors
    primelist = primeSieve(n)
  File "<pyshell#2>", line 3, in primeSieve
    sieve = [True] * (sieveSize+1)
OverflowError: cannot fit 'int' into an index-sized integer
>>>
```

하지만 문제에 제시된 숫자가 int형의 범위를 넘어선 큰 수여서 에러가 발생하게 됩니다.

이후 int형를 float형으로 변환하려고 시도했지만 계속된 오류가 발생했습니다.

Ln: 5 Col: 31

Python 3.8.2 Shell - C:/Users/smk38/Desktop/fjkdfjk.py (3.8.2)

File Edit Shell Debug Options Window Help

```
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:45:29) [MSC v.1916 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import math
>>> def primeSieve(sieveSize):
        # creating Sieve (0~n까지의 slot)
        sieve = [True] * (sieveSize+1)
        # 0과 1은 소수가 아니므로 제외
        sieve[0] = False
        sieve[1] = False
        # 2부터 (루트 n) + 1까지의 숫자를 탐색
        for i in range(2,int(math.sqrt(sieveSize))+1):
                # i가 소수가 아니면 pass
                if sieve[i] == False:
                        continue
                # i가 소수라면 i*i~n까지 숫자 가운데 i의 배수를
                # 소수에서 제외
                for pointer in range(i**2, sieveSize+1, i):
                        sieve[pointer] = False
        primes = []
        # sieve 리스트에서 True인 것이 소수이므로
        # True인 값의 인덱스를 결과로 저장
        for i in range(sieveSize+1):
                if sieve[i] == True:
                        primes.append(i)
        return primes

>>> def get_prime_factors(n):
        # n 범위 내의 소수를 구한다
        primelist = primeSieve(n)
        # 이 소수들 중 n으로 나누어 떨어지는
        # 소수를 구하고, 몇 번 나눌 수 있는지 계산
        # 예 : n = 8, factors = [(2, 3)]
        # 예 : n = 100, fcount = [(2: 2), (5: 2)]
        factors = []
        for p in primelist:
                count = 0
                while n % p == 0:
                        n /= p
                        count += 1
                if count > 0:
                        factors.append((p, count))
        return factors

>>> get_prime_factors(36)
[(2, 2), (3, 2)]
>>>
```

Int 범위 내의 숫자인 36을 대입하면 다음과 같이 잘 실행되는 것을 볼 수 있습니다.

Ln: 20 Col: 26