pb37

March 4, 2020

[370]:
```
from itertools import islice
```

[367]:
```
def is_prime(n):
    if n < 2: return False
    for i in range(2,int(n**0.5)+1):
        if n%i == 0: return False
    return True
```

[396]:
```
list(filter(is_prime, range(10)))
```

[396]:
```
[2, 3, 5, 7]
```

[391]:
```
def is_left_truncatable_prime(n):
#     print(n)
    if len(n) > 1:
        return is_prime(int(n)) and is_left_truncatable_prime(n[1:])
    return is_prime(int(n))

def is_right_truncatable_prime(n):
#     print(n)
    if len(n) > 1:
        return is_prime(int(n)) and is_right_truncatable_prime(n[:-1])
    return is_prime(int(n))

def is_truncatable_prime(n):
    return is_left_truncatable_prime(n) and is_right_truncatable_prime(n)
```

[397]:
```
def truncatables():
    i = 9
    while True:
        if is_truncatable_prime(str(i)):
            yield i
        i += 1

sum(islice(truncatables(), 11)) # [23, 37, 53, 73, 313, 317, 373, 797, 3137,␣
↪3797, 739397]
```

[397]: 748317

[385]:
```
def primes_naive():
    i = 2
    while True:
        if is_prime(i):
            yield i
        i += 1

primes_optimized = filter(is_prime, range(2, 100))

print(list(islice(primes_naive(), 5))) # isslice permet de stoper la boucle en␣
↪cours
print(list(islice(primes_optimized, 5)))
```

```
[2, 3, 5, 7, 11]
[2, 3, 5, 7, 11]
```

[413]:
```
is_multiple = lambda x : x%3 == 0 or x%5 == 0

def multiples():
    i = 0
    while True:
        if is_multiple(i):
            yield i
        i += 1

# methode 1
s1 = sum(filter(is_multiple, range(1000))) # itere de i = 0 à 1000 et renvoi␣
↪les elements verifiants la condition

# methode 2
s2 = sum(islice(multiples(), 467)) # renvoi les 467 elements verifiants la␣
↪condition 'is_multiple'

print(s1, s2)
```

```
233168 233168
```

[ ]: