pb41

March 7, 2020

[105]:
```
from itertools import permutations, count
```

[263]:
```
# methode 1 de generation
ndigits_pandigital = lambda n : ''.join(('%s' %i for i in range(n,0,-1)))
```

[264]:
```
ndigits_pandigital(9)
```

[264]:
```
'987654321'
```

[ ]:
```
# methode 2 de generation
def generate_basic_ndigits_pandigital(n): # a n-pandigital number contains all␣
↪digits from 1 to n
    nb = n*10**(n-1)
    for i in range(n-1):
        nb += (i+1)*10**(i)
    return nb
```

[276]:
```
list((generate_basic_ndigits_pandigital(i) for i in range(9,0,-1))) # base of␣
↪9-digitals to 1 digital numbers
```

[276]:
```
[987654321, 87654321, 7654321, 654321, 54321, 4321, 321, 21, 1]
```

[ ]:
```
# methode 3 de generation
def generate_basic_pandigitals():
    yield from (n*10**(n-1) + sum(((i+1)*10**i for i in range(n-1))) for n in␣
↪range(9,0,-1))
```

[277]:
```
list((i for i in generate_basic_pandigitals()))
```

[277]:
```
[987654321, 87654321, 7654321, 654321, 54321, 4321, 321, 21, 1]
```

[ ]:
```
def fact(n):
    if n < 2:
        return 1
    return n * fact(n-1)
```

[278]:
```
sum((fact(i) for i in range(9,0,-1))) # iterations totales en pire cas sur le calcul des permutations
```

[278]:
```
409113
```

[54]:
```
def is_prime(n):
    for i in range(2, int(n**0.5)+1):
        if n%i == 0:
            return False
    return True
```

[265]:
```
list(filter(is_prime, range(2,30)))
```

[265]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

1 Méthode 1

[270]:
```
def pandigitals_old_fashioned():
    n = 9 # nth of digits
    while n > 1:
        nb = ndigits_pandigital(n)
        for i in permutations(nb):
            i_int = int(''.join(i))
            if is_prime(i_int):
                return i_int
        n -= 1
```

[271]:
```
pandigitals_old_fashioned()
```

[271]:
```
7652413
```

2 Méthode 2

[275]:
```
max((int(''.join(i)) for nb in range(2, 9) for i in permutations(ndigits_pandigital(nb)) if is_prime(int(''.join(i)))))
```

[275]:
```
7652413
```

It's time consumming to call "int(".join(i))" each time, so we will use nested generators to just call the fonction once

[326]:
```
max(res for res in (int(''.join(i)) for nb in range(2, 9) for i in permutations(ndigits_pandigital(nb))) if is_prime(res))
```

[326]: 7652413

3 Méthode 3

[255]:
```
def find_largest_pandigital_prime():
    for p in generate_basic_pandigitals():
        p_string = str(p)
        for p_str in permutations(p_string):
            p_int = int(''.join(p_str))
            if is_prime(p_int):
                return p_int
    return
```

[256]:
```
find_largest_pandigital_prime()
```

[256]: 7652413

4 Performance's evaluations

[327]:
```
%timeit max(res for res in (int(''.join(i)) for nb in range(2, 9) for i in␣
↪permutations(ndigits_pandigital(nb))) if is_prime(res))
```

239 ms ± 1.99 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[328]:
```
%timeit find_largest_pandigital_prime()
```

846 ms ± 55.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[329]:
```
%timeit max((int(''.join(i)) for nb in range(2, 9) for i in␣
↪permutations(ndigits_pandigital(nb)) if is_prime(int(''.join(i)))))
```

247 ms ± 21.3 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

[330]:
```
%timeit pandigitals_old_fashioned()
```

792 ms ± 7.38 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)