# 문제 1

num\_list1 = list(range(1, 1001)) # 1부터 1000까지의 수를 저장하는 리스트

for i in num\_list1:

if (i % 2 == 0) : # 대상이 되는 수가 2의 배수라면 그 수를 리스트에서 제거한다

num\_list1.remove(i)

for i in num\_list1:

if i % 3 == 0:

num\_list1.remove(i) # 대상이 되는 수가 3의 배수라면 그 수를 리스트에서 제거한다

print(num\_list1)

[1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47, 49, 53, 55, 59, 61, 65, 67, 71, 73, 77, 79, 83, 85, 89, 91, 95, 97, 101, 103, 107, 109, 113, 115, 119, 121, 125, 127, 131, 133, 137, 139, 143, 145, 149, 151, 155, 157, 161, 163, 167, 169, 173, 175, 179, 181, 185, 187, 191, 193, 197, 199, 203, 205, 209, 211, 215, 217, 221, 223, 227, 229, 233, 235, 239, 241, 245, 247, 251, 253, 257, 259, 263, 265, 269, 271, 275, 277, 281, 283, 287, 289, 293, 295, 299, 301, 305, 307, 311, 313, 317, 319, 323, 325, 329, 331, 335, 337, 341, 343, 347, 349, 353, 355, 359, 361, 365, 367, 371, 373, 377, 379, 383, 385, 389, 391, 395, 397, 401, 403, 407, 409, 413, 415, 419, 421, 425, 427, 431, 433, 437, 439, 443, 445, 449, 451, 455, 457, 461, 463, 467, 469, 473, 475, 479, 481, 485, 487, 491, 493, 497, 499, 503, 505, 509, 511, 515, 517, 521, 523, 527, 529, 533, 535, 539, 541, 545, 547, 551, 553, 557, 559, 563, 565, 569, 571, 575, 577, 581, 583, 587, 589, 593, 595, 599, 601, 605, 607, 611, 613, 617, 619, 623, 625, 629, 631, 635, 637, 641, 643, 647, 649, 653, 655, 659, 661, 665, 667, 671, 673, 677, 679, 683, 685, 689, 691, 695, 697, 701, 703, 707, 709, 713, 715, 719, 721, 725, 727, 731, 733, 737, 739, 743, 745, 749, 751, 755, 757, 761, 763, 767, 769, 773, 775, 779, 781, 785, 787, 791, 793, 797, 799, 803, 805, 809, 811, 815, 817, 821, 823, 827, 829, 833, 835, 839, 841, 845, 847, 851, 853, 857, 859, 863, 865, 869, 871, 875, 877, 881, 883, 887, 889, 893, 895, 899, 901, 905, 907, 911, 913, 917, 919, 923, 925, 929, 931, 935, 937, 941, 943, 947, 949, 953, 955, 959, 961, 965, 967, 971, 973, 977, 979, 983, 985, 989, 991, 995, 997]

**문제 1의 첫번째 풀이**

* 수를 저장하는 리스트, 그 리스트에서 2의 배수 제거와 3의 배수 제거가 관건이었다.
* range()를 이용해서 1부터 1000까지의 수를 생성하고, list()로 그것을 리스트로 만들어준다
* 리스트는 iterable 객체이기 때문에 for in 반복문을 사용할 수 있다.
* 리스트의 remove() 메소드는 리스트 객체에서 remove의 parameter를 제거할 수 있다

# 문제 1

num\_list2 = list(range(1, 1001))

for i in num\_list2:

if (i % 2 == 0) or (i % 3 ==0) : # 2의 배수이거나 3의 배수이면

num\_list2.remove(i) # 그 수를 리스트에서 제거한다

print(num\_list2) # 3의 배수가 remove되지 않는데 그 이유는...

# or 연산에서 앞부분만 참이면 전체가 참이 되서 그런듯..이라고 추정

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 389, 391, 393, 395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 425, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455, 457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 481, 483, 485, 487, 489, 491, 493, 495, 497, 499, 501, 503, 505, 507, 509, 511, 513, 515, 517, 519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 549, 551, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577, 579, 581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 609, 611, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 711, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, 735, 737, 739, 741, 743, 745, 747, 749, 751, 753, 755, 757, 759, 761, 763, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789, 791, 793, 795, 797, 799, 801, 803, 805, 807, 809, 811, 813, 815, 817, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 851, 853, 855, 857, 859, 861, 863, 865, 867, 869, 871, 873, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895, 897, 899, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 921, 923, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 949, 951, 953, 955, 957, 959, 961, 963, 965, 967, 969, 971, 973, 975, 977, 979, 981, 983, 985, 987, 989, 991, 993, 995, 997, 999]

# 문제 1

num\_list3 = list(range(1, 1001))

for i in num\_list3:

if (i % 3 == 0) or (i % 2 == 0): # 2의 배수이거나 3의 배수이면

num\_list3.remove(i) # 그 수를 리스트에서 제거한다

print(num\_list3) # 3의 배수가 remove되지 않는데 그 이유는...

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259, 261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287, 289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343, 345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365, 367, 369, 371, 373, 375, 377, 379, 381, 383, 385, 387, 389, 391, 393, 395, 397, 399, 401, 403, 405, 407, 409, 411, 413, 415, 417, 419, 421, 423, 425, 427, 429, 431, 433, 435, 437, 439, 441, 443, 445, 447, 449, 451, 453, 455, 457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 479, 481, 483, 485, 487, 489, 491, 493, 495, 497, 499, 501, 503, 505, 507, 509, 511, 513, 515, 517, 519, 521, 523, 525, 527, 529, 531, 533, 535, 537, 539, 541, 543, 545, 547, 549, 551, 553, 555, 557, 559, 561, 563, 565, 567, 569, 571, 573, 575, 577, 579, 581, 583, 585, 587, 589, 591, 593, 595, 597, 599, 601, 603, 605, 607, 609, 611, 613, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 671, 673, 675, 677, 679, 681, 683, 685, 687, 689, 691, 693, 695, 697, 699, 701, 703, 705, 707, 709, 711, 713, 715, 717, 719, 721, 723, 725, 727, 729, 731, 733, 735, 737, 739, 741, 743, 745, 747, 749, 751, 753, 755, 757, 759, 761, 763, 765, 767, 769, 771, 773, 775, 777, 779, 781, 783, 785, 787, 789, 791, 793, 795, 797, 799, 801, 803, 805, 807, 809, 811, 813, 815, 817, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 851, 853, 855, 857, 859, 861, 863, 865, 867, 869, 871, 873, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895, 897, 899, 901, 903, 905, 907, 909, 911, 913, 915, 917, 919, 921, 923, 925, 927, 929, 931, 933, 935, 937, 939, 941, 943, 945, 947, 949, 951, 953, 955, 957, 959, 961, 963, 965, 967, 969, 971, 973, 975, 977, 979, 981, 983, 985, 987, 989, 991, 993, 995, 997, 999]

**문제 1의 두번째 풀이**

* or 연산자를 이용해서 2의 배수와 3의 배수를 동시에 제거하려 했다
* 근데 3의 배수가 제거되지 않는다
* 처음에는 (i % 2 == 0)이 or의 앞에 와서 or 연산의 앞부분이 참이면 전체가 참이기 때문에 (i % 3 ==0) 부분은 실행되지 않은건가 했는데
* (i % 3 == 0)을 앞에 놓아도 3의 배수가 제거되지 않더라

# 문제 1 numpy의 ndarray로 계산

import numpy as np

np\_list = np.arange(1, 1001) # 1부터 1000까지의 수, num\_list3 = list(range(1, 1001))와 같다

# 2의 배수가 아니고, 3의 배수가 아닌 것들만으로 ndarray(리스트)를 초기화

np\_list = np\_list[((np\_list % 2 !=0 ) & (np\_list % 3 != 0))]

print(np\_list)

[ 1 5 7 11 13 17 19 23 25 29 31 35 37 41 43 47 49 53

55 59 61 65 67 71 73 77 79 83 85 89 91 95 97 101 103 107

109 113 115 119 121 125 127 131 133 137 139 143 145 149 151 155 157 161

163 167 169 173 175 179 181 185 187 191 193 197 199 203 205 209 211 215

217 221 223 227 229 233 235 239 241 245 247 251 253 257 259 263 265 269

271 275 277 281 283 287 289 293 295 299 301 305 307 311 313 317 319 323

325 329 331 335 337 341 343 347 349 353 355 359 361 365 367 371 373 377

379 383 385 389 391 395 397 401 403 407 409 413 415 419 421 425 427 431

433 437 439 443 445 449 451 455 457 461 463 467 469 473 475 479 481 485

487 491 493 497 499 503 505 509 511 515 517 521 523 527 529 533 535 539

541 545 547 551 553 557 559 563 565 569 571 575 577 581 583 587 589 593

595 599 601 605 607 611 613 617 619 623 625 629 631 635 637 641 643 647

649 653 655 659 661 665 667 671 673 677 679 683 685 689 691 695 697 701

703 707 709 713 715 719 721 725 727 731 733 737 739 743 745 749 751 755

757 761 763 767 769 773 775 779 781 785 787 791 793 797 799 803 805 809

811 815 817 821 823 827 829 833 835 839 841 845 847 851 853 857 859 863

865 869 871 875 877 881 883 887 889 893 895 899 901 905 907 911 913 917

919 923 925 929 931 935 937 941 943 947 949 953 955 959 961 965 967 971

973 977 979 983 985 989 991 995 997]

num\_list4 = list(range(1, 1001))

num\_list4[((num\_list4 % 2 !=0 ) and (num\_list4 % 3 != 0))]

---------------------------------------------------------------------------

TypeError Traceback (most recent call last)

<ipython-input-4-919d2594f220> in <module>

1 num\_list4 = list(range(1, 1001))

2

----> 3 num\_list4[((num\_list4 % 2 !=0 ) and (num\_list4 % 3 != 0))]

TypeError: unsupported operand type(s) for %: 'list' and 'int'

**문제 1의 세번째 풀이**

* numpy 모듈(라이브러리?)를 이용했다.
* numpy에서는 boolean indexing을 이용해서 for,if를 사용하지 않아도 반복적인 조건판별이 가능하더라
* numpy 안쓰고도 boolean indexing이 가능한지 시도해봣으나 안되었음

import numpy as np

target = np.random.randint(100) # numpy를 사용하지 않을거면 target = random.randrange(1,100)

# 1부터 100 중에서의 임의의 수

print(target) # 테스팅할 때 편하라고 일부러 target을 출력

count = 0 # 반복횟수

while(True): # 얼마나 반복해야 될지 모르므로 for 대신 while 반복

trial = int(input('숫자를 맞춰보세요: ')) # 입력시도

if target > trial: # target이 trial보다 크면 큼! 이라고 출력

count += 1

print('큼!')

elif target < trial: # 작으면 작음! 이라고 출력

count += 1

print('작음!')

else:

count+=1

print('정답!')

print(count,'번 만에 정답을 맞추셨습니다')

quit = input('Quit(Y/N): ') # 종료여부를 입력받음

if quit == 'Y':

print('게임 종료')

break

elif quit == 'N': # 종료하지 않을 경우('N') 전체 초기화

print('계속합니다')

target = np.random.randint(100) # target을 재설정

count = 0 # 반복횟수도 재설정

continue # 처음으로 돌아간다

else:

print('잘못된 입력입니다')

print('프로그램을 종료합니다')

59

숫자를 맞춰보세요: 59

정답!

1 번 만에 정답을 맞추셨습니다

Quit(Y/N): N

계속합니다

숫자를 맞춰보세요: 60

큼!

숫자를 맞춰보세요: 80

큼!

숫자를 맞춰보세요: 99

작음!

숫자를 맞춰보세요: 95

정답!

4 번 만에 정답을 맞추셨습니다

Quit(Y/N): Y

게임 종료

**문제 2**

* random.randrange()를 사용해도 좋지만 numpy의 np.arange()를 사용했다
* 임의의 수를 먼저 생성해야하고, 입력받는 수(trial)와 맞추려고하는 수(target)의 비교가 필요
* 맞출 때까지 반복되야하므로 for보단 while이 적합하다
* 종료여부도 입력받고 판단해야한다
* 입력받는 부분은 input()를 사용했음
* 과제에는 명시되어 있지 않지만 종료하지 않고 계속 진행했을 경우,
* 목적이 되는 수를 리셋해야하고 반복횟수도 재설정해줘야한다(아예 새로운 게임이 시작되는것)
* 종료여부에서 잘못 입력됬을 경우 프로그램 종료(일종의 예외처리)

temperature = input('온도(높음/낮음): ')

wind = input('바람(강/약): ')

weather = input('하늘(비/흐림/맑음): ')

# 하늘이 '비'일땐 항상 TV, '맑음'일 땐 항상 축구여서

# 기준으로 하늘을 먼저 세우는 것이 편했다

if weather == '비': # 하늘이 '비'이면 항상 TV

print('TV')

elif weather == '맑음': # 하늘이 '맑음'이면 항상 축구

print('축구')

elif weather == '흐림': # 흐림일 때 TV,축구 둘다 있어서 더 자세한 조건이 필요

# 흐림일때 바람이 '강'일 때도 축구, '약'일때도 축구여서 바람조건보다는 온도 조건이 더 탁월

if temperature == '낮음':

print('TV') # 즉,

else :

print('축구') # 즉,

온도(높음/낮음): 낮음

바람(강/약): 강

하늘(비/흐림/맑음): 맑음

축구

temperature = input('온도(높음/낮음): ')

wind = input('바람(강/약): ')

weather = input('하늘(비/흐림/맑음): ')

# 하늘이 '비'일땐 항상 TV, '맑음'일 땐 항상 축구여서

# 기준으로 하늘을 먼저 세우는 것이 편했다

if weather == '비': # 하늘이 '비'이면 항상 TV

print('TV')

elif weather == '맑음': # 하늘이 '맑음'이면 항상 축구

print('축구')

elif weather == '흐림':

if wind == '강':

print('축구')

elif wind == '약':

if temperature == '높음':

print('축구')

elif temperature == '낮음':

print('TV')

온도(높음/낮음): 낮음

바람(강/약): 약

하늘(비/흐림/맑음): 흐림

TV

**문제 3**

* 무엇을 기준으로 제일 처음 나눌것인가?가 첫번째 관건
* 하늘을 기준으로 세운 이유는, 온도는 '높음/낮음'일 때 축구,TV 둘다 있었고, 바람또한 '강/약' 관계없이 축구,TV가 있었다.
* 하늘은 그런 애매함 없이 '비'이면 항상 TV, '맑음'이면 축구(비록 케이스가 하나이지만)로 단순했다
* 하늘이 '흐림'일 때 축구/TV로 나뉘기 때문에 더 세세한 조건이 필요
* 주어진 데이터론 바람이 '강'일때도 축구, '약'일때도 축구라서 바람조건보다는 온도조건으로 기준을 세우는 것이 더 효과적
* 실제로 바람조건으로 기준을 세웠을 때는 조건문이 늘어난다