

UNIT 20

FizzBuzz 문제

20 FizzBuzz 문제

» FizzBuzz 문제

- FizzBuzz는 매우 간단한 프로그래밍 문제이며 규칙은 다음과 같음

1. 1에서 100까지 출력
2. 3의 배수는 Fizz 출력
3. 5의 배수는 Buzz 출력
4. 3과 5의 공배수는 FizzBuzz 출력

- 1부터 100까지 숫자를 출력하면서 3의 배수는 숫자 대신 'Fizz', 5의 배수는 숫자 대신 'Buzz', 3과 5의 공배수는 숫자 대신 'FizzBuzz'를 출력하면 됨
- FizzBuzz 문제는 프로그래밍 면접에 자주 등장하는 문제임
- 문제 조건을 꼼꼼히 따지지 않으면 경력자도 실수하기 쉽고, 면접에서 탈락하기도 함
- 그만큼 기초실력을 가늠하는 문제이기 때문에 잘 알아 두면 좋음

20.1 1부터 100까지 숫자 출력하기

» 1부터 100까지 숫자 출력하기

- FizzBuzz 문제는 반복문, 조건문, 나머지 연산자, 비교 연산자를 모두 동원해야 풀 수 있음

print_1_to_100.py

```
for i in range(1, 101):    # 1부터 100까지 100번 반복
    print(i)
```

실행 결과

```
1
2
3
... (생략)
98
99
100
```

- for와 range로 1부터 100까지 100번 반복하면서 print로 변수의 값을 출력하면 됨
- range에서 반복되는 마지막 숫자는 끝나는 숫자보다 1이 더 작으므로 101을 지정해야 됨

20.2 3의 배수일 때와 5의 배수일 때 처리하기

» 3의 배수일 때와 5의 배수일 때 처리하기

multiple_of_3_5.py

```
for i in range(1, 101):    # 1부터 100까지 100번 반복
    if i % 3 == 0:         # 3의 배수일 때
        print('Fizz')     # Fizz 출력
    elif i % 5 == 0:      # 5의 배수일 때
        print('Buzz')     # Buzz 출력
    else:
        print(i)          # 아무것도 해당되지 않을 때 숫자 출력
```

```
1
2
Fizz
... (생략)
97
98
Fizz
Buzz
```

20.3 3과 5의 공배수 처리하기

» 3과 5의 공배수 처리하기

- 공배수는 다음과 같이 논리 연산자 and를 사용하면 됨

fizzbuzz.py

```
for i in range(1, 101):
    if i % 3 == 0 and i % 5 == 0:
        print('FizzBuzz')
    elif i % 3 == 0:
        print('Fizz')
    elif i % 5 == 0:
        print('Buzz')
    else:
        print(i)
```

1부터 100까지 100번 반복
3과 5의 공배수일 때
FizzBuzz 출력
3의 배수일 때
Fizz 출력
5의 배수일 때
Buzz 출력
아무것도 해당되지 않을 때 숫자 출력

실행 결과

```
1
2
Fizz
... (생략)
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

20.3 3과 5의 공배수 처리하기

» 3과 5의 공배수 처리하기

- $i \% 3 == 0$ and $i \% 5 == 0$ 처럼 i 를 3으로 나눴을 때 나머지가 0이면서 5로 나눴을 때도 나머지가 0이면 3과 5의 공배수임
- 이때는 print로 'FizzBuzz'를 출력해줍
- i 가 30인데 if에서 3의 배수를 먼저 검사하면 3과 5의 공배수는 검사를 하지 않고 그냥 넘어가버리므로 주의하자

```
if i % 3 == 0:                # i가 30이면
    print('Fizz')            # Fizz를 출력하고 그냥 넘어가버림
elif i % 5 == 0:
    print('Buzz')
elif i % 3 == 0 and i % 5 == 0: # 3과 5의 공배수는 검사하지 못함
    print('FizzBuzz')
else:
    print(i)
```

20.4 논리 연산자를 사용하지 않고 3과 5의 공배수 처리하기

» 논리 연산자를 사용하지 않고 3과 5의 공배수 처리하기

- 3 * 5 = 15는 3과 5의 최소공배수이므로 15로 나눴을 때 나머지가 0인 값들은 3과 5의 공배수임

fizzbuzz_without_logical_operator.py

```
for i in range(1, 101):    # 1부터 100까지 100번 반복
    if i % 15 == 0:        # 15의 배수(3과 5의 공배수)일 때
        print('FizzBuzz') # FizzBuzz 출력
    elif i % 3 == 0:       # 3의 배수일 때
        print('Fizz')     # Fizz 출력
    elif i % 5 == 0:       # 5의 배수일 때
        print('Buzz')     # Buzz 출력
    else:                  # 아무것도 해당되지 않을 때 숫자 출력
        print(i)
```

실행 결과

```
1
2
Fizz
... (생략)
FizzBuzz
91
92
Fizz
94
Buzz
Fizz
97
98
Fizz
Buzz
```

20.5 코드 단축하기

» 코드 단축하기

fizzbuzz_code_golf.py

```
for i in range(1, 101):  
    print('Fizz' * (i % 3 == 0) + 'Buzz' * (i % 5 == 0) or i)  
    # 문자열 곱셈과 덧셈을 이용하여 print 안에서 처리
```

실행 결과

```
1  
2  
Fizz  
... (생략)  
FizzBuzz  
91  
92  
Fizz  
94  
Buzz  
Fizz  
97  
98  
Fizz  
Buzz
```


20.5 코드 단축하기

» 코드 단축하기

- 코드 골프란 골프에서 따온 말인데, 실제 골프 경기는 더 적은 타수로 승부를 거름
- 코드 골프도 소스 코드의 문자 수를 최대한 줄여서 작성하는 놀이(코드의 문자 수를 얼마나 줄일 수 있는지 겨루는 놀이일 뿐 실무에서 이런 방식으로 작성하면 나중에 작성자 본인을 포함해서 여러 사람이 고생하게 됨)
- 문자열을 곱하면 문자열이 반복되고, 문자열을 더하면 두 문자열이 연결됨
- 문자열에 True를 곱하면 문자열이 그대로 나오고, False를 곱하면 문자열이 출력되지 않음(True는 1, False는 0으로 연산)

```
>>> 'Fizz' + 'Buzz'
'FizzBuzz'
>>> 'Fizz' * True
'Fizz'
>>> 'Fizz' * False
''
```

20.5 코드 단축하기

» 코드 단축하기

- 문자열 곱셈을 이용하여 3의 배수일 때 'Fizz'를 출력
- i 가 3의 배수이면 $i \% 3 == 0$ 은 True이므로 'Fizz'가 출력되고, 3의 배수가 아니면 False이므로 'Fizz'가 출력되지 않음

```
'Fizz' * (i % 3 == 0)
```

- 'Buzz'도 문자열 곱셈을 이용하여 5의 배수일 때 출력

```
'Buzz' * (i % 5 == 0)
```

20.5 코드 단축하기

» 코드 단축하기

- 3과 5의 공배수일 때는 'FizzBuzz'를 출력해야 하는데 이때는 문자열 덧셈을 이용함
- 3과 5의 공배수이면 'Fizz' * True + 'Buzz' * True가 되므로 'Fizz' + 'Buzz'로 'FizzBuzz'를 출력
- 만약 한 쪽이 만족하지 않으면 덧셈할 문자열이 없으므로 'Fizz'나 'Buzz'만 출력

```
'Fizz' * (i % 3 == 0) + 'Buzz' * (i % 5 == 0)
```

- 3 또는 5의 배수가 아닐 때는 'Fizz' * False + 'Buzz' * False가 되고 결과는 빈 문자열 ""이 되는데, 이때는 or 연산자를 사용함
- 빈 문자열은 False로 취급하고, i는 항상 1 이상의 숫자이므로 or로 연산하면 i만 남게 되어 숫자가 그대로 출력

```
print('Fizz' * (i % 3 == 0) + 'Buzz' * (i % 5 == 0) or i)
```

20.5 코드 단축하기

» 코드 단축하기

- 파이썬에서 연산자의 특성을 활용하면 코드의 길이를 짧게 줄일 수 있음
- 코드가 읽기 어려워 진다면 이해하기 쉽도록 풀어서 작성하는 것이 좋음
- 프로그래밍 실력보다 중요한 것이 요구 사항에 대한 이해력이라는 점 잊지말자