





- 1. 함수의 정의 및 호출
- 2. 매개변수
- 3. return
- 4. 지역변수와 전역변수
- 5. 함수를 담기
- 6. 함수를 다른 함수의 매개변수로 사용하기
- 7. 함수에서 함수를 return 하기
- 8. 함수 안의 함수
- 9. 재귀 함수
- 10. 함수의 시간 측정
- 11. lambda를 통한 함수 생성
- 12. 파일 생성
- 13. 파일 쓰기, 읽기
- 14. 파일에 새로운 내용 추가하기
- 15. with문과 함께 사용하기

# 함수

```
def 함수이름(매개변수1, 매개변수2):  
    # 코드 블록  
    return 결과값
```

```
함수이름(매개변수1, 매개변수2)
```

## 함수의 정의, 호출

```
def hello1():  
    print("Hello")  
  
def hello2(name):  
    print("Hello~", name)  
  
def hello3(name, count):  
    for x in range(count):  
        print("Hello!", name)
```

```
hello1()  
hello2("Kim")  
hello3("Lee", 3)
```

## 함수의 매개변수

```
def hello(name = '', count = 1):  
    for x in range(count):  
        print("Hello", name)
```

```
hello()  
hello('손흥민')  
hello('이강인', 3)  
  
hello(count = 4)
```

# 함수의 매개변수

```
>>> help(print)
```

```
Help on built-in function print in module builtins:
```

```
print(*args, sep=' ', end='\n', file=None, flush=False)
```

```
    Prints the values to a stream, or to sys.stdout by default.
```

```
    sep
```

```
        string inserted between values, default a space.
```

```
    end
```

```
        string appended after the last value, default a newline.
```

```
    file
```

```
        a file-like object (stream); defaults to the current sys.stdout.
```

```
    flush
```

```
        whether to forcibly flush the stream.
```

## 함수의 매개변수

```
def hello_names(*names):  
    for name in names:  
        print("Hello", name)
```

```
hello_names('손흥민', '이강인', '황희찬')
```

## 함수의 매개변수

```
def hello_player(**players):  
    for key in players:  
        print("Hello", key, players[key])
```

```
hello_player(손흥민 = 'FW', 이강인 = 'MF', 황희찬 = 'MF')
```



## 함수의 매개변수

```
def hello_names(count = 1, *names):  
    for name in names:  
        print("Hello" * count, name)  
  
# hello_names('손흥민', '이강인', '황희찬', 2)  
hello_names(2, '손흥민', '이강인', '황희찬')  
# hello_names(count = 2, '손흥민', '이강인', '황희찬')  
# hello_names('손흥민', '이강인', '황희찬', count = 2)  
# hello_names('손흥민', '이강인', '황희찬')
```

## 함수의 매개변수

```
def hello_names(*names, count = 1):  
    for name in names:  
        print("Hello" * count, name)  
  
hello_names('손흥민', '이강인', '황희찬', 2)  
hello_names(2, '손흥민', '이강인', '황희찬')  
# hello_names(count = 2, '손흥민', '이강인', '황희찬')  
hello_names('손흥민', '이강인', '황희찬', count = 2)  
hello_names('손흥민', '이강인', '황희찬')
```



# 함수의 return

1. return문에 결과데이터 전달하기 → 함수는 종료되고, 데이터가 전달됩니다.
2. return문 뒤에 결과데이터를 생략하기 → 함수가 종료됩니다. (None을 반환)
3. return문 자체를 생략하기 → 함수의 모든 코드가 실행되어야 종료됩니다. (None을 반환)

```
def one_return():  
    print("1을 반환합니다")  
    return 1  
    print("1을 반환했습니다")
```

```
def empty_return():  
    print("반환값이 없습니다")  
    return  
    print("return이 실행되었습니다")
```

```
def no_return():  
    print("return이 없습니다")
```

## 지역변수, 전역변수

```
score = 100

def score_change(score):
    score -= 5

score_change(score)

print(score)
```

## 지역변수, 전역변수

```
score = 100

def score_change():
    global score
    score -= 5

score_change()
print(score)
```

```
score = 100

def score_change(score):
    return score - 5

score = score_change(score)
print(score)
```

# 함수를 담기

```
def print_something(text):  
    print(text)  
  
ps = print_something  
ps("Hello World!")
```

```
def plus(a, b):  
    return a + b  
  
def minus(a, b):  
    return a - b  
  
calc_list = [plus, minus]  
  
print(calc_list[0](1, 2))  
print(calc_list[1](1, 2))
```

```
def plus(a, b):  
    return a + b  
def minus(a, b):  
    return a - b  
  
calc_dict = {'plus': plus, 'minus': minus}  
  
print(calc_dict['plus'](1, 2))  
print(calc_dict['minus'](1, 2))
```

# 함수를 다른 함수의 매개변수로 사용하기

```
def robot_calc(calc_function):  
    print("로봇 계산기입니다.")  
    num1 = int(input("첫 번째 숫자를 입력하세요: "))  
    num2 = int(input("두 번째 숫자를 입력하세요: "))  
    result = calc_function(num1, num2)  
    print("계산 결과는 {}입니다.".format(result))  
  
def plus(num1, num2):  
    return num1 + num2  
  
def minus(num1, num2):  
    return num1 - num2  
  
robot_calc(plus)  
robot_calc(minus)
```

# 함수에서 함수를 return 하기

```
def say_hello():  
    nation = input("국적을 입력하세요: ")  
    if nation.lower() == "korea":  
        return hello_Korean()  
    else:  
        return hello_English()  
def hello_Korean():  
    print("안녕하세요")  
  
def hello_english():  
    print("Hello")  
  
say_hello()
```



## 함수 안의 함수

```
def average(*scores):  
    def length():  
        return len(scores)  
    def sum():  
        total = 0  
        for score in scores:  
            total += score  
        return total  
    return sum() / length()  
  
print(average(10, 20, 30, 40, 50))
```



# 재귀 함수

```
def 함수이름(매개변수):  
    if 매개변수가종료조건에부합  
        return 결과값
```

...

```
    함수이름(변경된입력값)  
    return 결과값
```

함수이름(입력값)

## 재귀 함수

```
def recursion_sum(num):  
    if num == 1:  
        return 1  
    result = recursion_sum(num - 1)  
    return num + result  
  
print(recursion_sum(5))
```

## 재귀 함수

`recursion_sum(5) = 5 + recursion_sum(4) = 5 + 4 + 3 + 2 + 1`

`recursion_sum(4) = 4 + recursion_sum(3) = 4 + 3 + 2 + 1`

`recursion_sum(3) = 3 + recursion_sum(2) = 3 + 2 + 1`

`recursion_sum(2) = 2 + recursion_sum(1) = 2 + 1`

`recursion_sum(1) = 1`

# 팩토리얼

<https://www.acmicpc.net/problem/27433>

0보다 크거나 같은 정수  $N$ 이 주어진다. 이때,  $N!$ 을 출력하는 프로그램을 작성하시오.

입력	출력
10	3628800
입력	출력
5	120
입력	출력
0	1

# 하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>

세 개의 장대가 있고 첫 번째 장대에는 반경이 서로 다른  $n$ 개의 원판이 쌓여 있다.  
각 원판은 반경이 큰 순서대로 쌓여 있다.  
다음 규칙에 따라 첫 번째 장대에서 세 번째 장대로 옮기려 한다.

- 한 번에 한 개의 원판만을 다른 탑으로 옮길 수 있다.
- 쌓아 놓은 원판은 항상 위의 것이 아래의 것보다 작아야 한다.

이 작업을 수행하는데 필요한 이동 순서를 출력하는 프로그램을 작성하라.  
단, 이동 횟수는 최소가 되어야 한다.

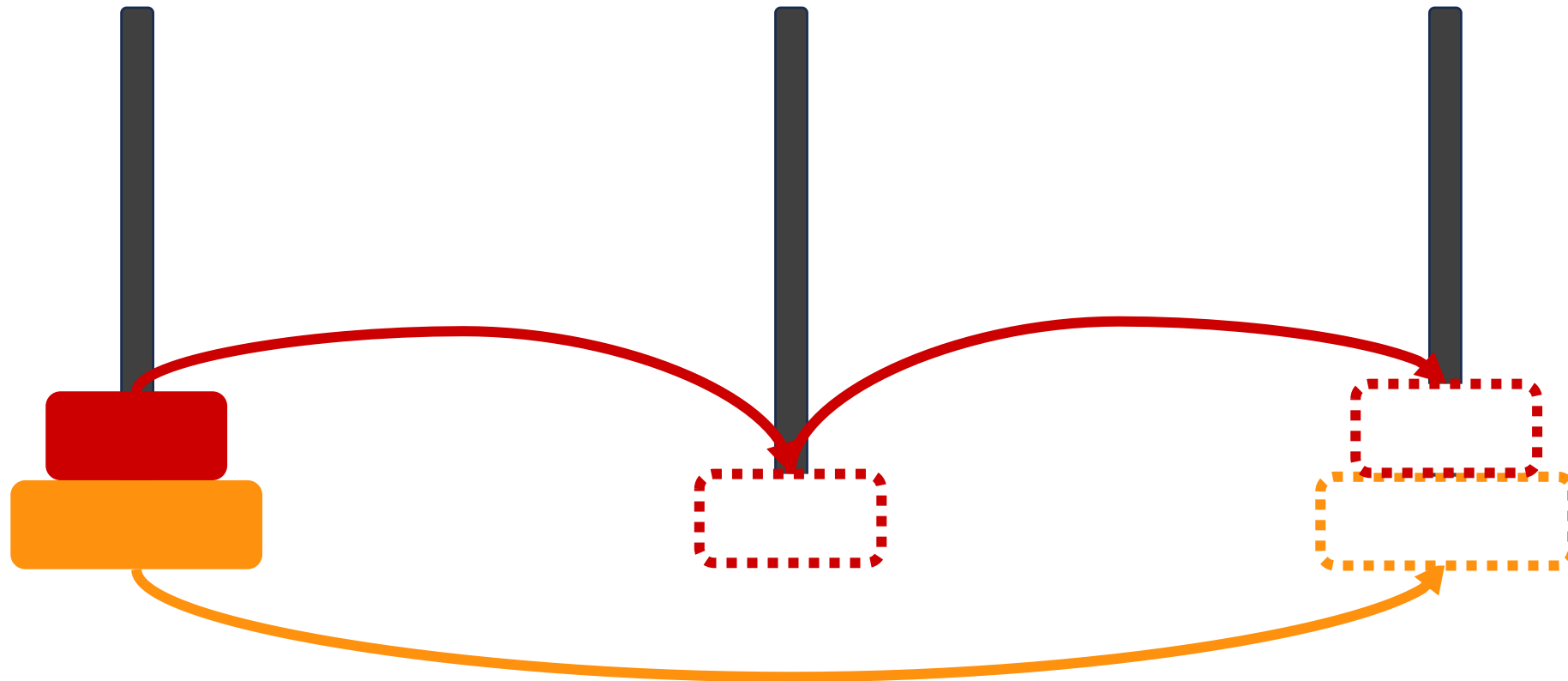
# 하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>



# 하노이 탑 이동 순서

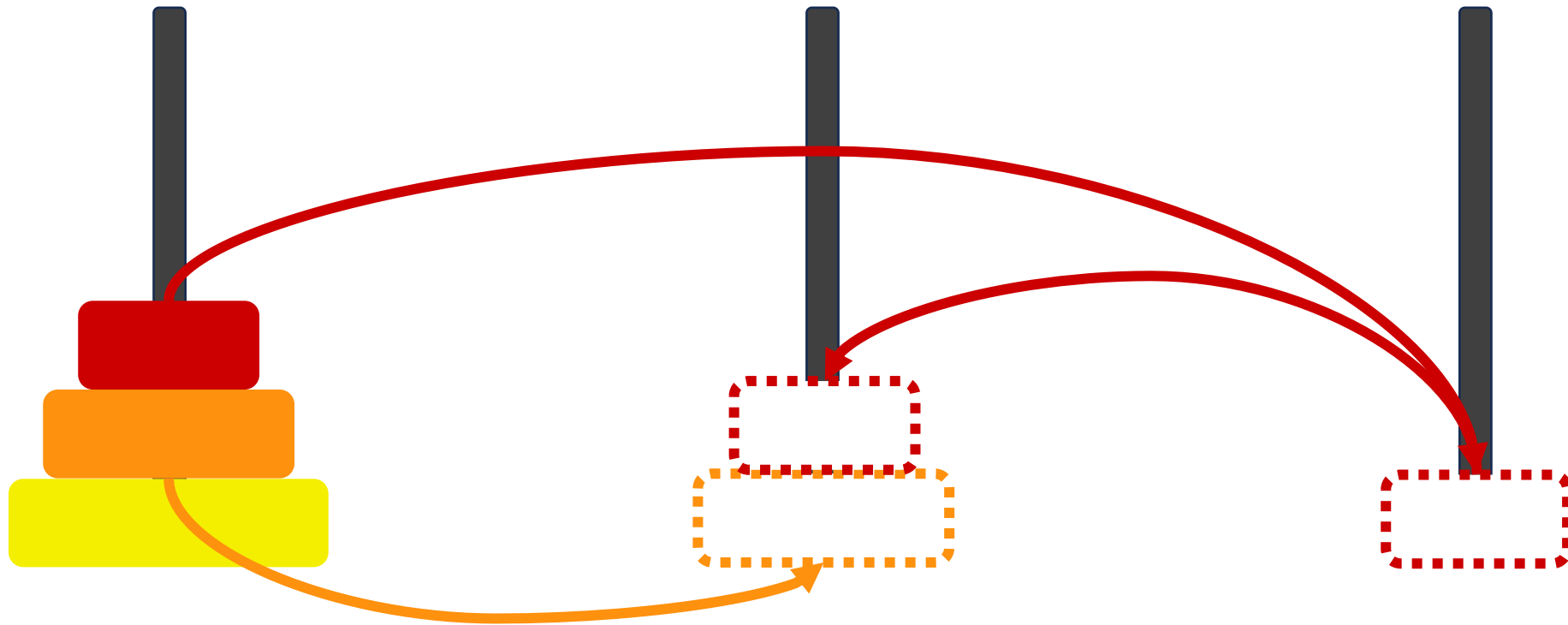
<https://www.acmicpc.net/problem/11729>





# 하노이 탑 이동 순서

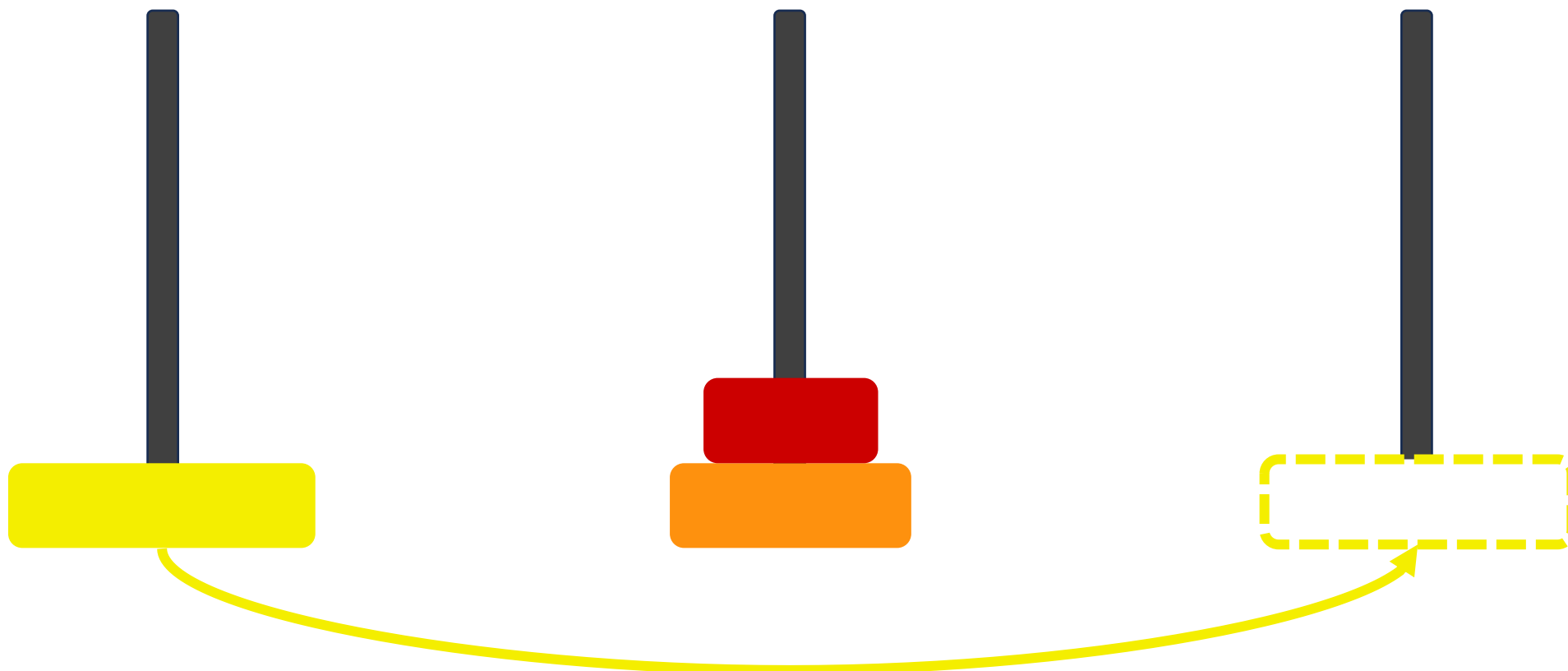
<https://www.acmicpc.net/problem/11729>





# 하노이 탑 이동 순서

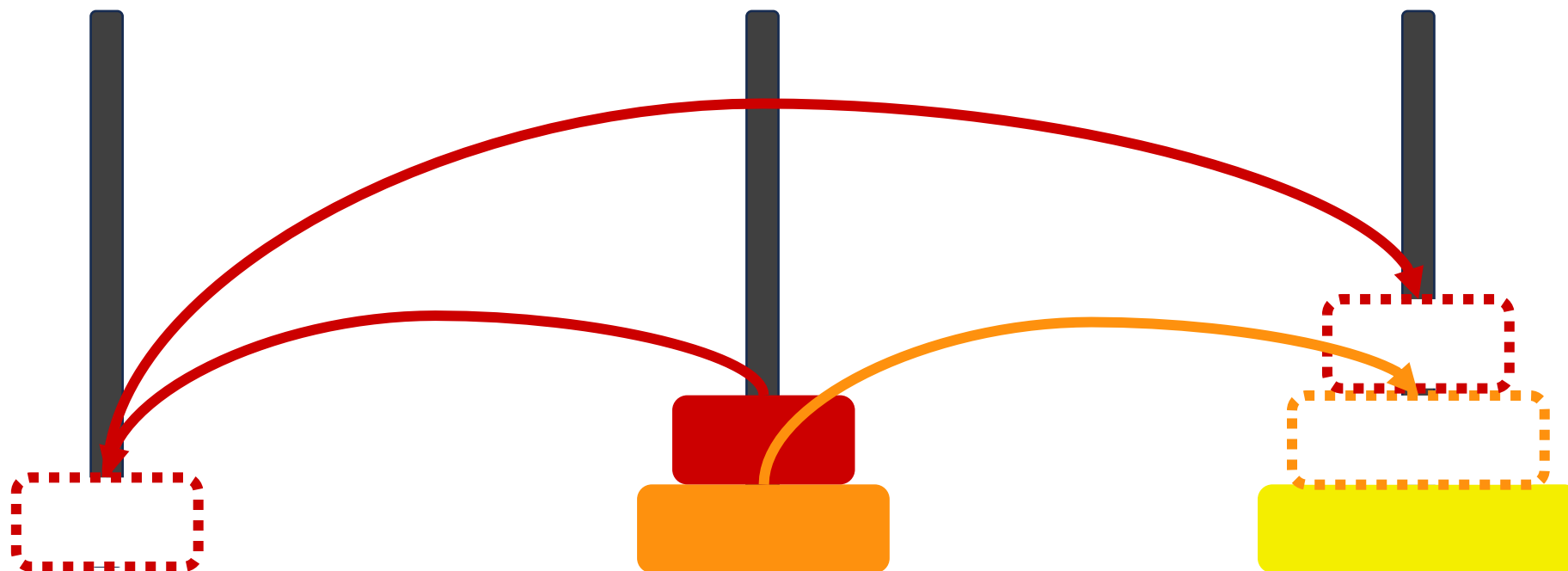
<https://www.acmicpc.net/problem/11729>





# 하노이 탑 이동 순서

<https://www.acmicpc.net/problem/11729>



# 함수의 시간 측정

```
import time

start = time.time()
# 함수의 연산 속도를 측정하고 싶은 부분
end = time.time()
print("{0:0.20} sec.".format(end - start))
```

# lambda를 통한 함수 생성

```
sum = lambda x, y: x + y
```

```
def sum(x, y):  
    return x + y
```



파일 열기



파일 읽기 / 쓰기



파일 닫기

# 파일 생성

```
f = open("note.txt", 'w')  
f.close()
```

파일 모드	설명
r	읽기 모드 - 파일을 읽기만 할 때 사용 (기본값)
w	쓰기 모드 - 파일에 내용을 쓸 때 사용 (파일이 존재하면 파일 내용을 비움)
x	쓰기 모드 - 파일에 내용을 쓸 때 사용 (파일이 존재하면 에러 발생)
a	추가 모드 - 파일의 마지막에 새로운 내용을 추가할 때 사용 (파일이 존재하면 이어써짐)

## 파일 쓰기 & 읽기

```
f = open("hello10.txt", 'w')
for i in range(10):
    f.write("Hello, world{}!\n".format(i+1))
f.close()
```

```
f = open("hello10.txt", 'r')
line = f.readline()
print(line)
f.close()
```



## 파일 읽기 readline()

```
f = open("hello10.txt", 'r')
while True:
    line = f.readline()
    if not line:
        break
    print(line.strip())
f.close()
```

## 파일 읽기 for line in file

```
f = open("hello10.txt", 'r')
for line in f:
    print(line.strip())
f.close()
```

## 파일 읽기 read()

```
f = open("hello10.txt", 'r')  
data = f.read()  
print(data)
```

## 파일에 새로운 내용 추가하기

```
f = open("hello10.txt", 'a', encoding='utf-8')
for i in range(11, 20):
    data = "안녕, 파이썬 %d~!\n" % i
    f.write(data)
f.close()
```

## with문과 함께 사용하기

```
with open("with.txt", 'a', encoding='utf-8') as f:  
    f.write("with 블록을 벗어나는 순간 파일 객체 f가 자동으로 close됨")
```

# 랜덤 숫자 위치 맞추기

1~30까지 10개의 무작위 숫자가 입력되어 있는 rand\_num10.txt라는 파일이 있다.

숫자를 입력했을 때, 해당 숫자가 10개의 무작위 숫자 중에 존재한다면 몇 번째 줄에 있는지 반환하고, 없으면 -1을 반환한다.

rand_num10.txt  24 4 24 11 11 23 11 8 30 8	입력  11	출력  3
	입력  7	출력  -1