LET’S MAKE A DEAL

Play a Game

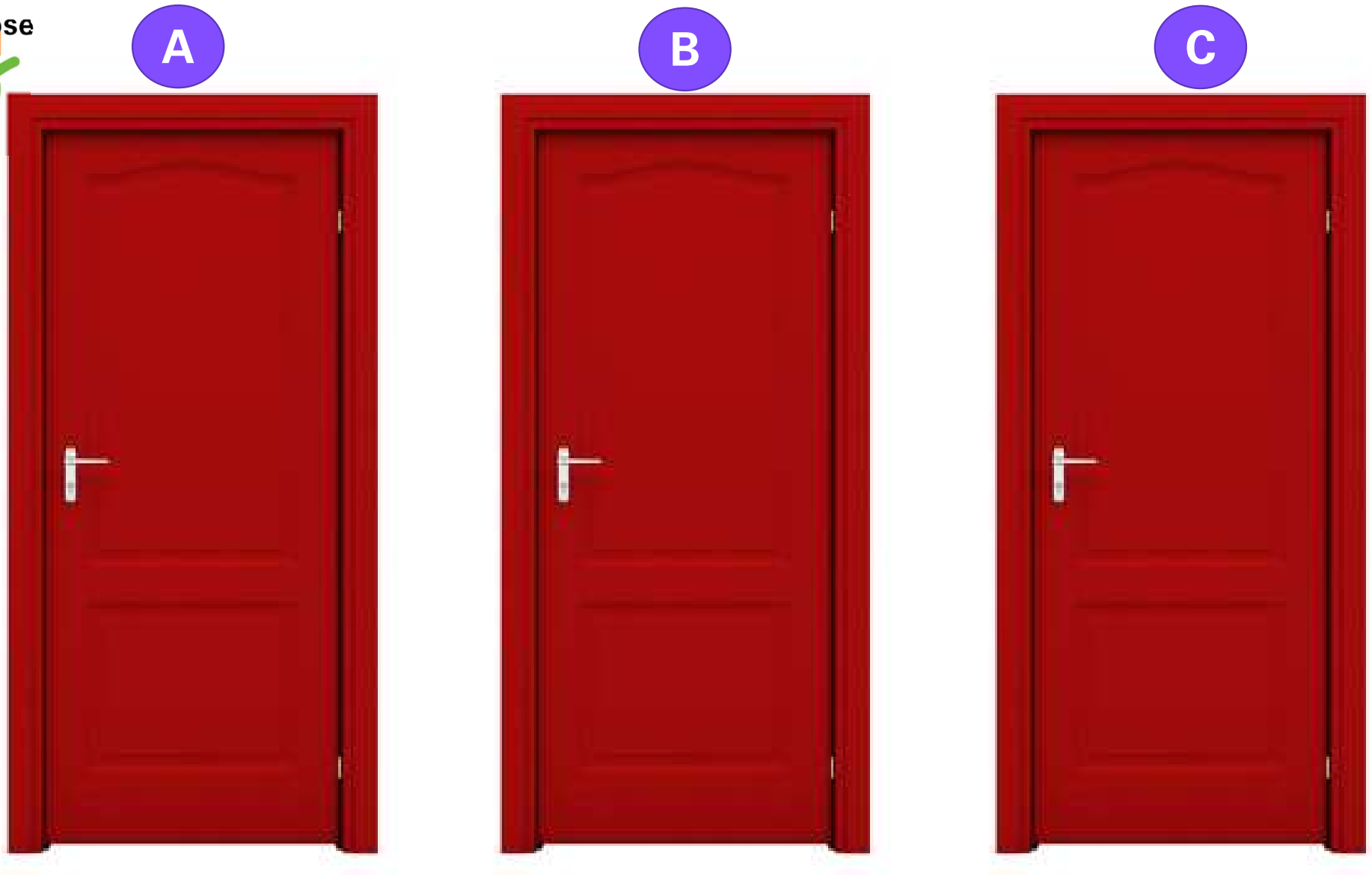
PLAY 1

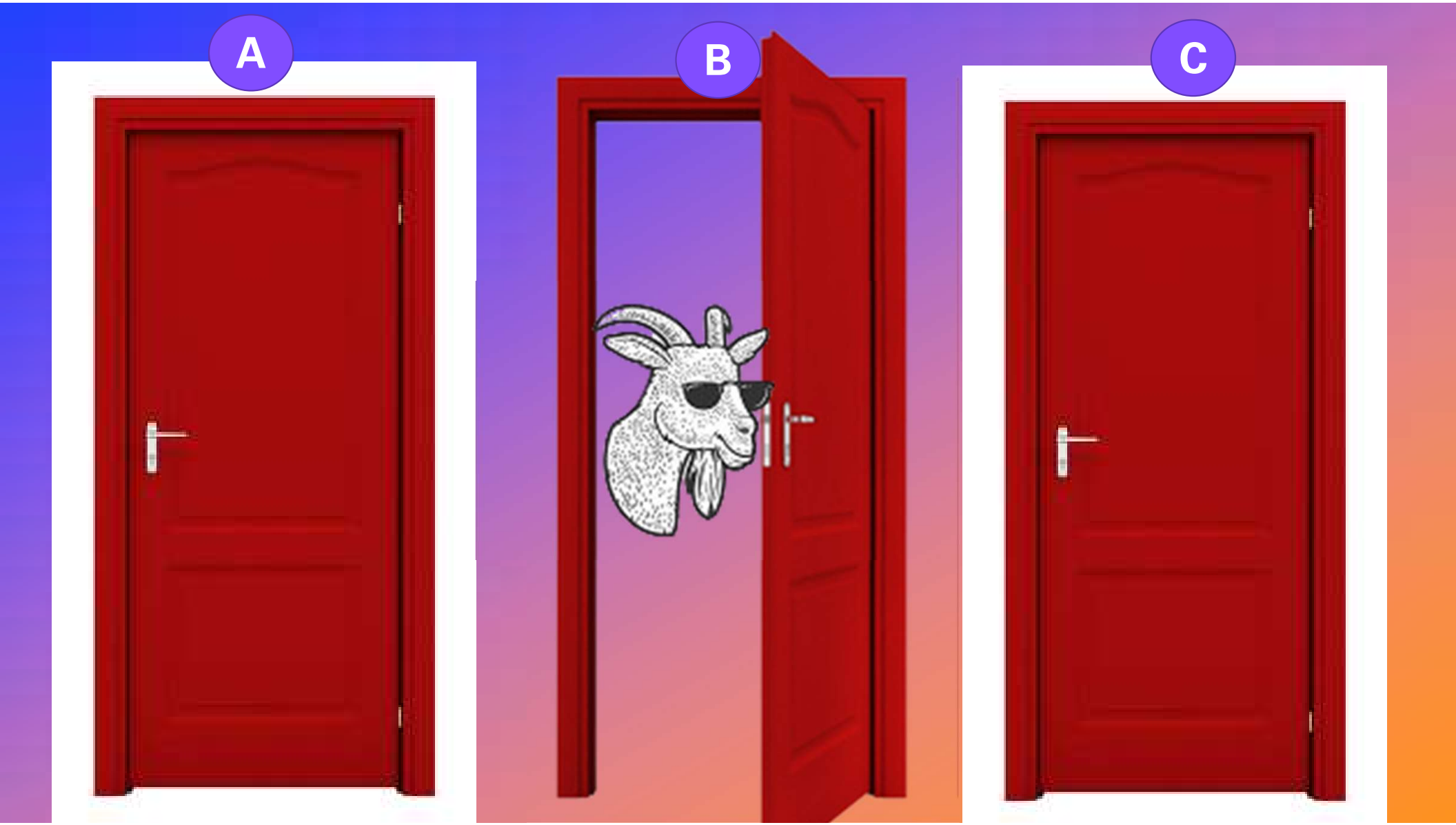
A

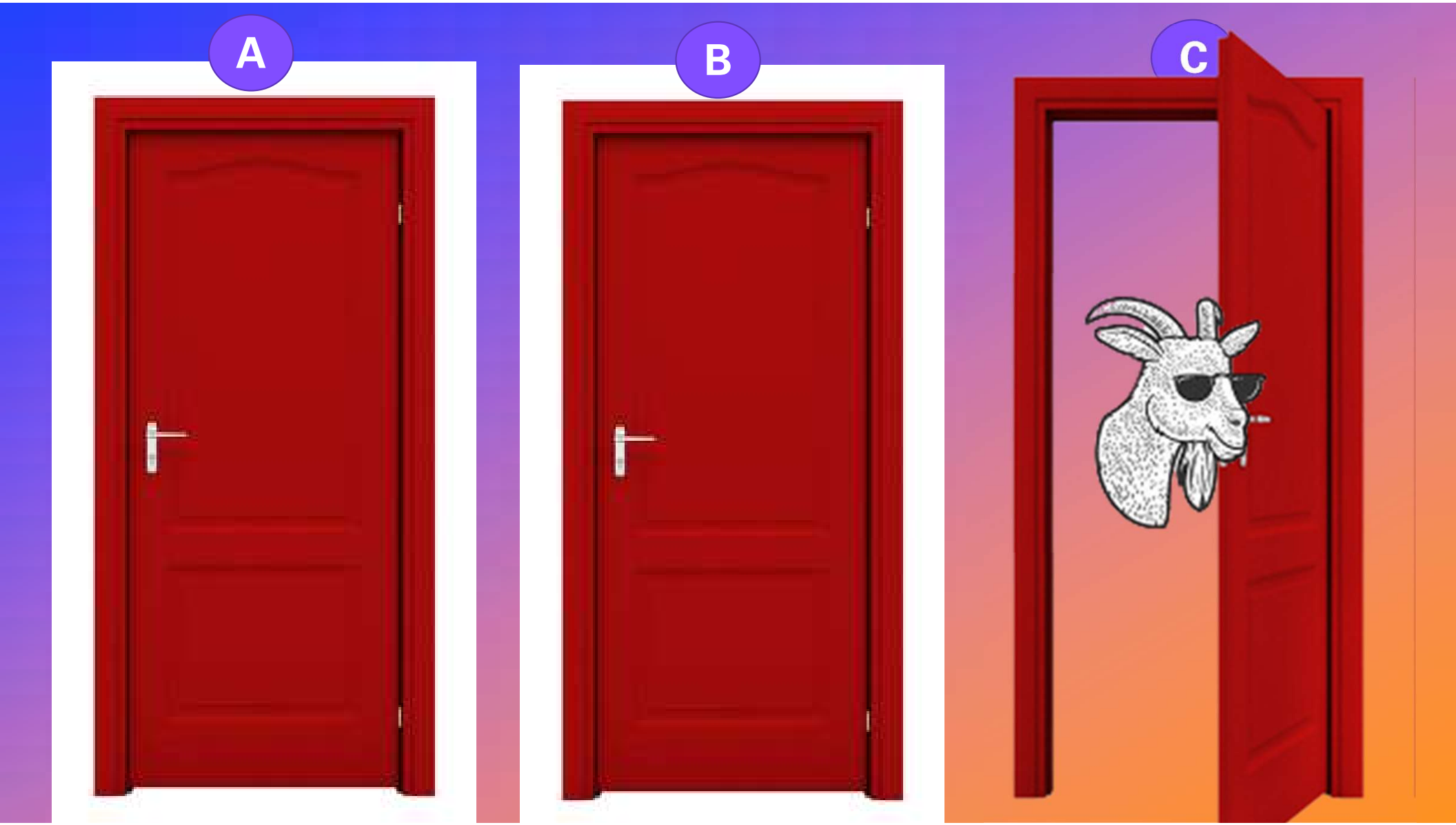
B

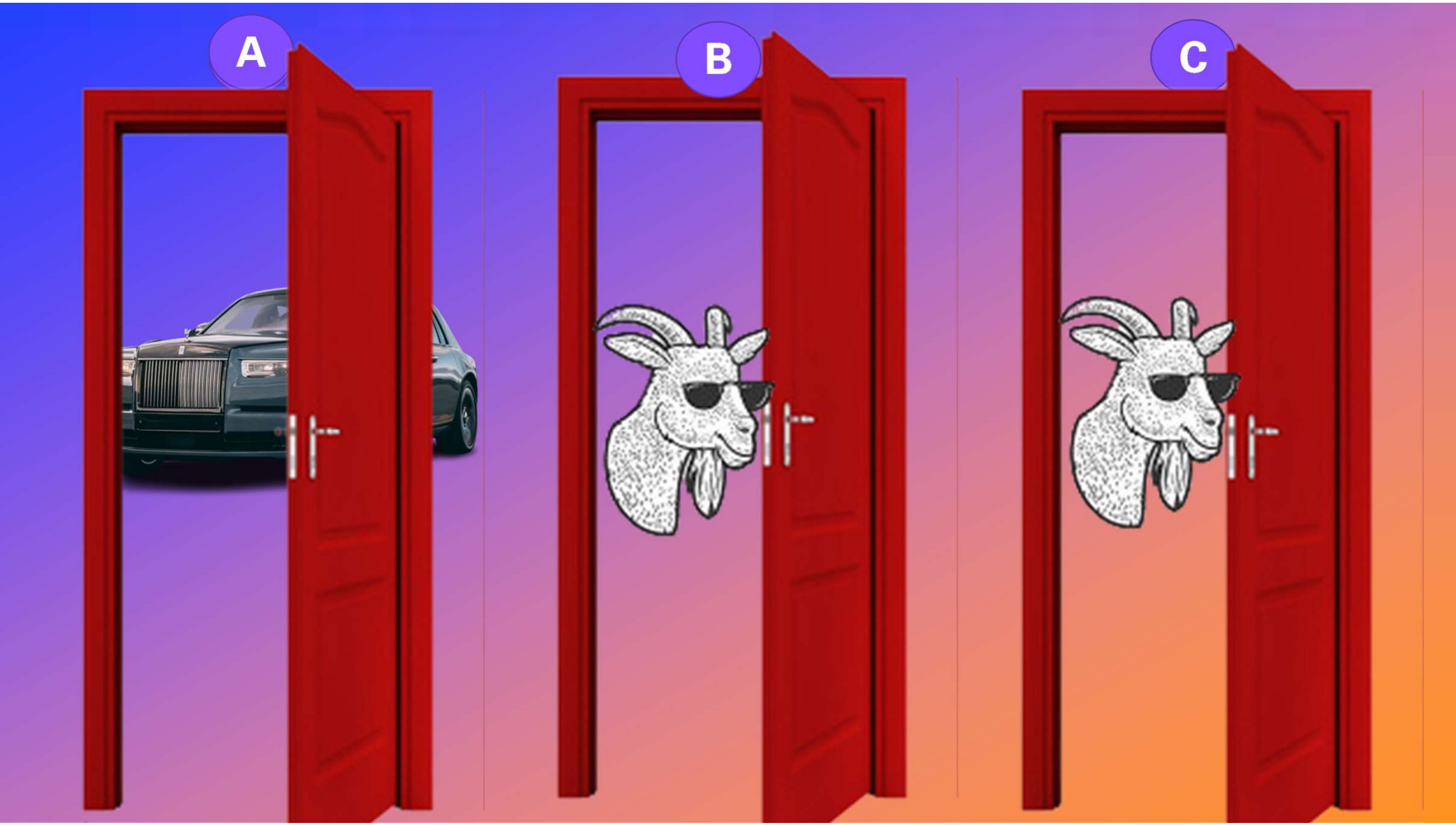
C

Choose

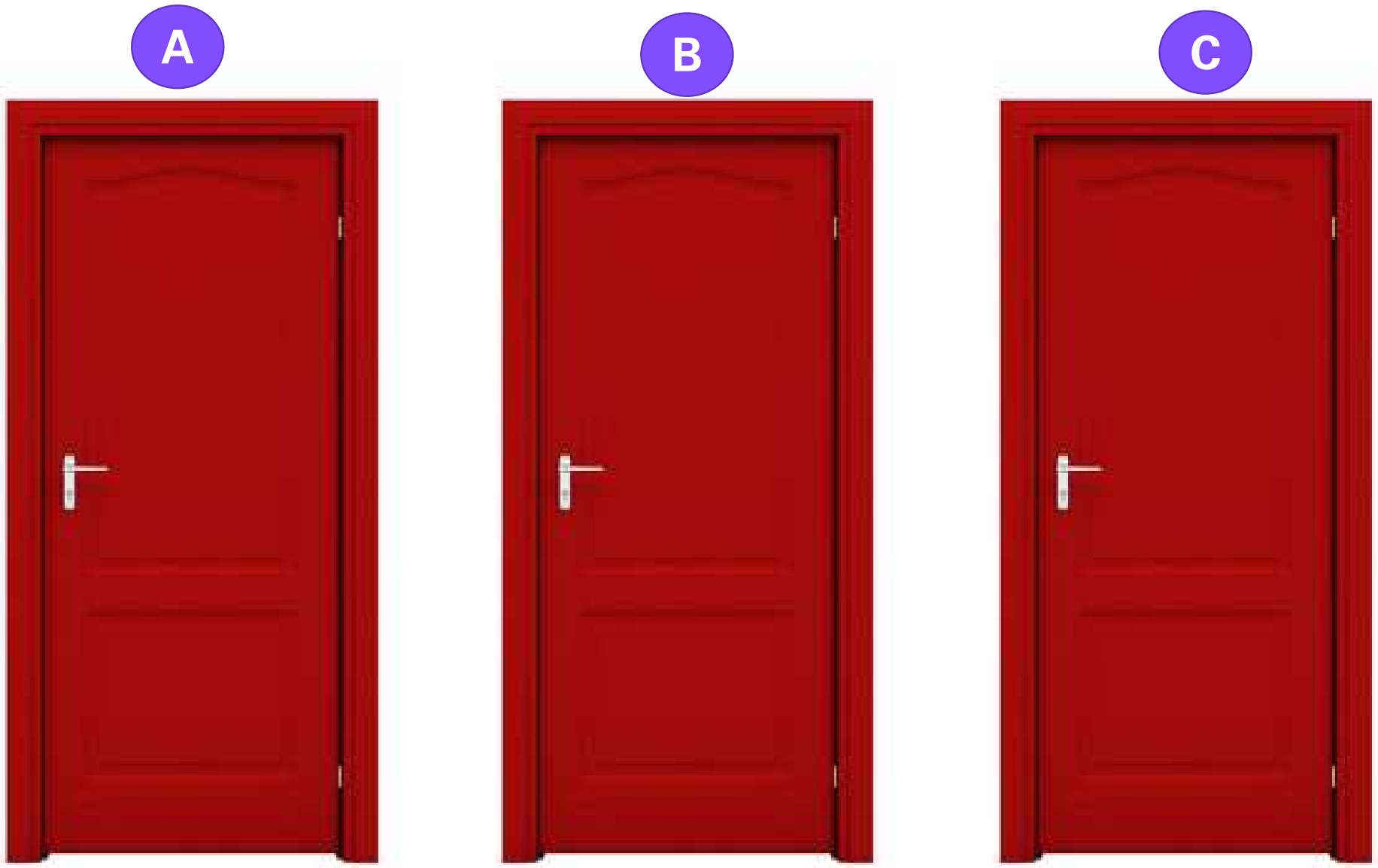



A
B
C

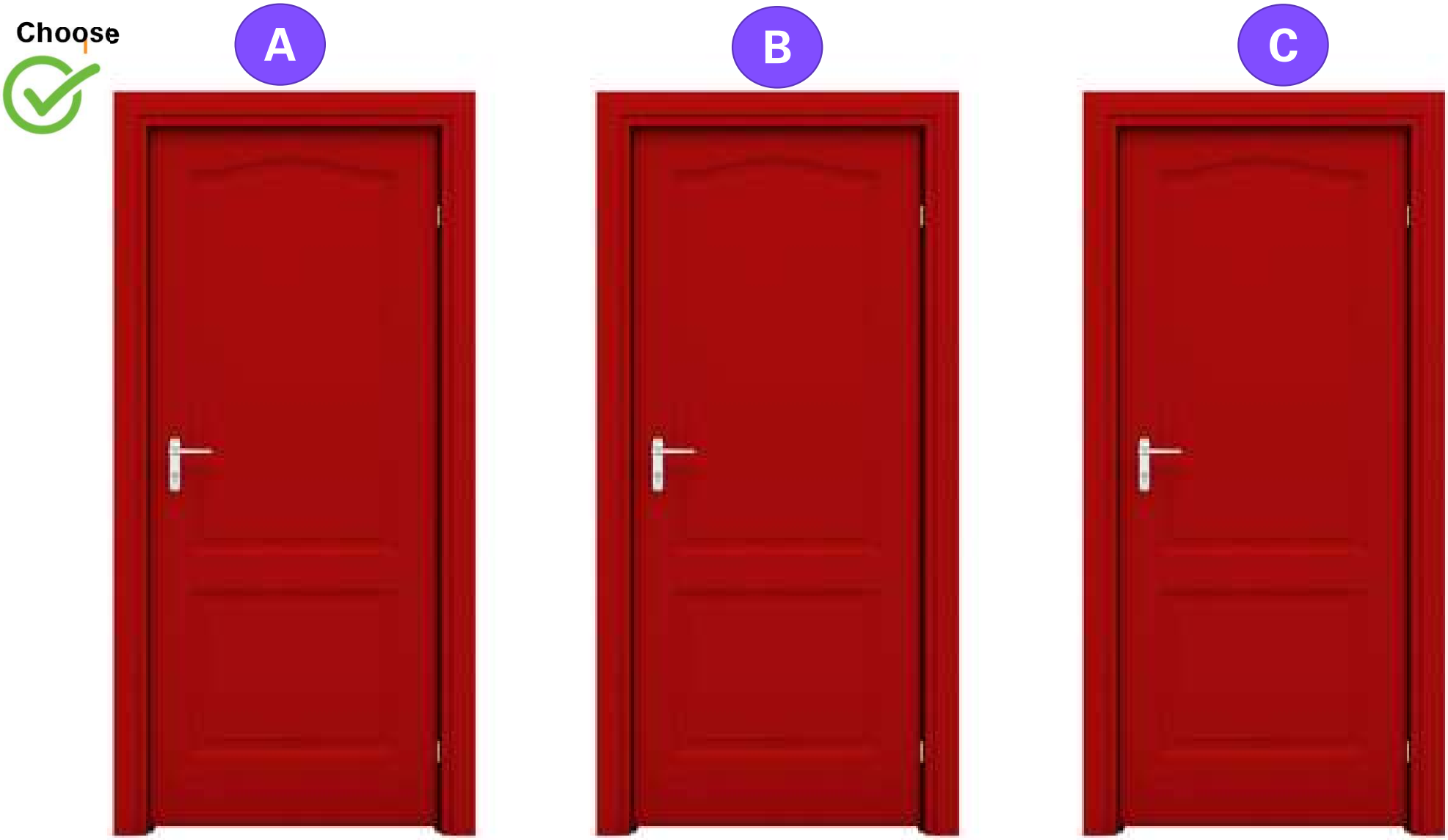

A
B
C

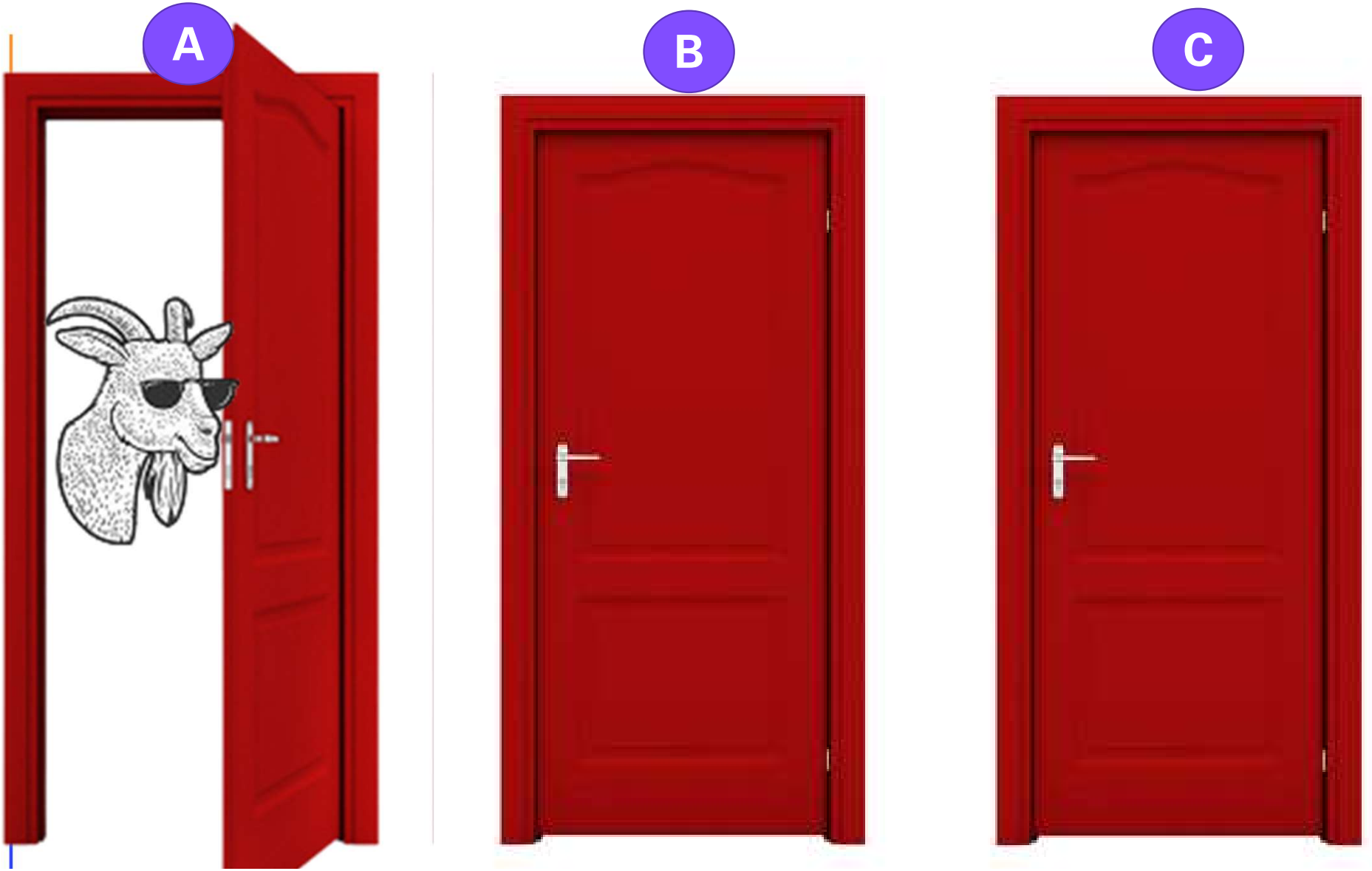

A
B
C

PLAY 2


A
B
C

Choose

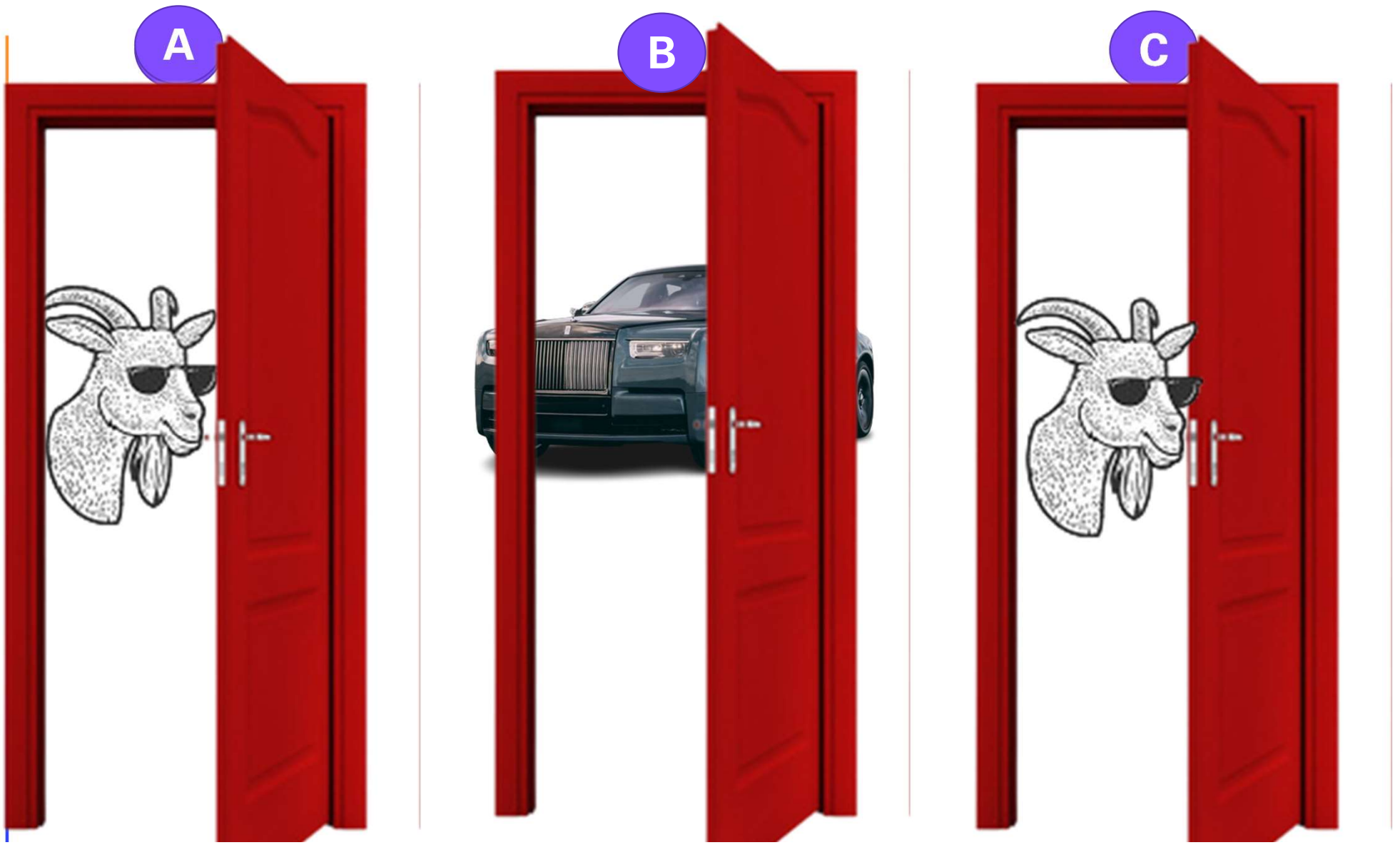



A
B
C

A

B

C

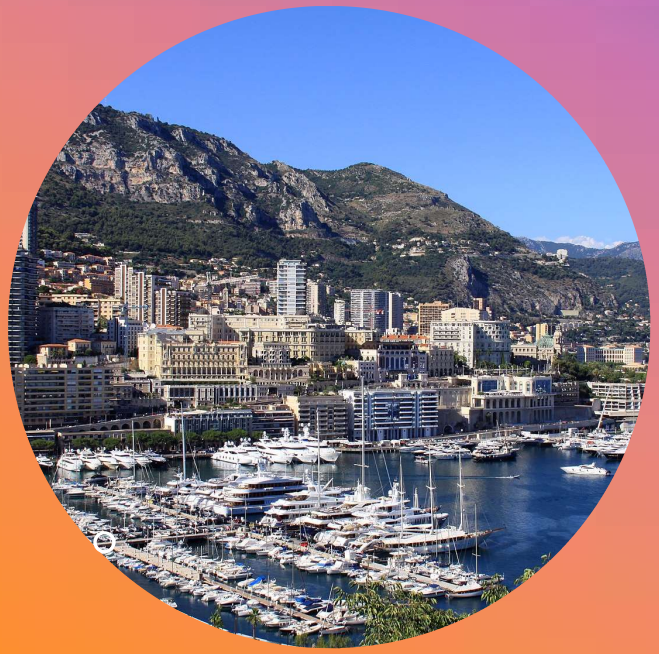

A
B
C

MONTY HALL PROBLEM

"brain teaser, in the form of a probability puzzle

loosely based on the American television game show *Let's Make a Deal* and named after its original host, Monty Hall.

MONTE CARLO



"random sampling to estimate probabilities and make predictions about uncertain outcomes."

SIMPLE CODE

Code

```
from random import shuffle, choice
def monty_hall():
  # Initialize the door with random allocation of the car i.e 1
  door = [0, 1, 0]
  shuffle(door)

  # Randomy Choose among three Doors
  door_choose = choice([0, 1, 2])

  # Open the door after user chooses the for the 1st time that does not
consist the car
  goat_door = list()
  for is_user_choice, is_goat_door in enumerate(door):
      if is_goat_door == 0 and is_user_choice != door_choose:
        goat_door.append(is_user_choice)
  host_door_opened = choice(goat_door)
```

Code

```
# if a user does not switch the door and Gets CAR --> Success
if door[door_choose] == 1:
  no_switch_success =  True
else:
  no_switch_success = False

# if a user switches the door and Gets CAR --> Success
left_door = set([0,1,2]).difference([door_choose, host_door_opened])
left_door =  left_door.pop()
if door[left_door] == 1:
  switch_success =  True
else:
  switch_success = False

return no_switch_success, switch_success
```

Code

```
def monte_carlo(n):
  no_switch_success_num = 0
  switch_success_num = 0

  for simulations in range(n):
      ns, ss = monty_hall()
      no_switch_success_num += ns
      switch_success_num += ss

  print("The number of simulations: {}".format(n))
  print("The number of success on switch: {}%".format((switch_success_num/n)*100))
  print("Number of success on non-switch: {}%".format((no_switch_success_num/n)*100))

number_of_trials = int(input("Enter the number of simulation (Large number up to
Thousands -> 000)"))
monte_carlo(number_of_trials)
```

OOP CODE

Code

```
import random

class MontyHallSimulation:
    def __init__(self):
        self.doors = [0, 0, 1]  # 0 represents a goat and 1 represents a car
        random.shuffle(self.doors)  # shuffle the doors randomly
        self.choice = None
        self.goat_door = None
        self.switch_choice = None

    def choose_door(self):
        self.choice = random.randint(0, 2)  # randomly choose a door
        self.goat_door = random.choice([i for i in range(3) if i != self.choice and
self.doors[i] == 0])  # host opens a goat door
```

Code

```
    def switch(self):
        self.switch_choice = next(i for i in range(3) if i != self.choice and i != 
self.goat_door)  # switch to the other unopened door

    def win_no_switch(self):
        return self.doors[self.choice] == 1  # return True if the contestant wins without 
switching

    def win_switch(self):
        return self.doors[self.switch_choice] == 1  # return True if the contestant wins 
by switching
```

Code

```
class MontyHall:
    def __init__(self, num_simulations):
        self.num_simulations = num_simulations
        self.wins_no_switch = 0
        self.wins_switch = 0

    def run_simulations(self):
        for i in range(self.num_simulations):
            simulation = MontyHallSimulation()
            simulation.choose_door()
            if simulation.win_no_switch():
                self.wins_no_switch += 1
            simulation.switch()
            if simulation.win_switch():
                self.wins_switch += 1
```

Code

```
    def print_results(self):
        print(f"Results after {self.num_simulations} simulations:")
        print(f"Win percentage without switching: {100 * self.wins_no_switch /
self.num_simulations:.2f}%")
        print(f"Win percentage with switching: {100 * self.wins_switch /
self.num_simulations:.2f}%")

monty_hall = MontyHall(1000)
monty_hall.run_simulations()
monty_hall.print_results()
```

Chart: Result of Simulation

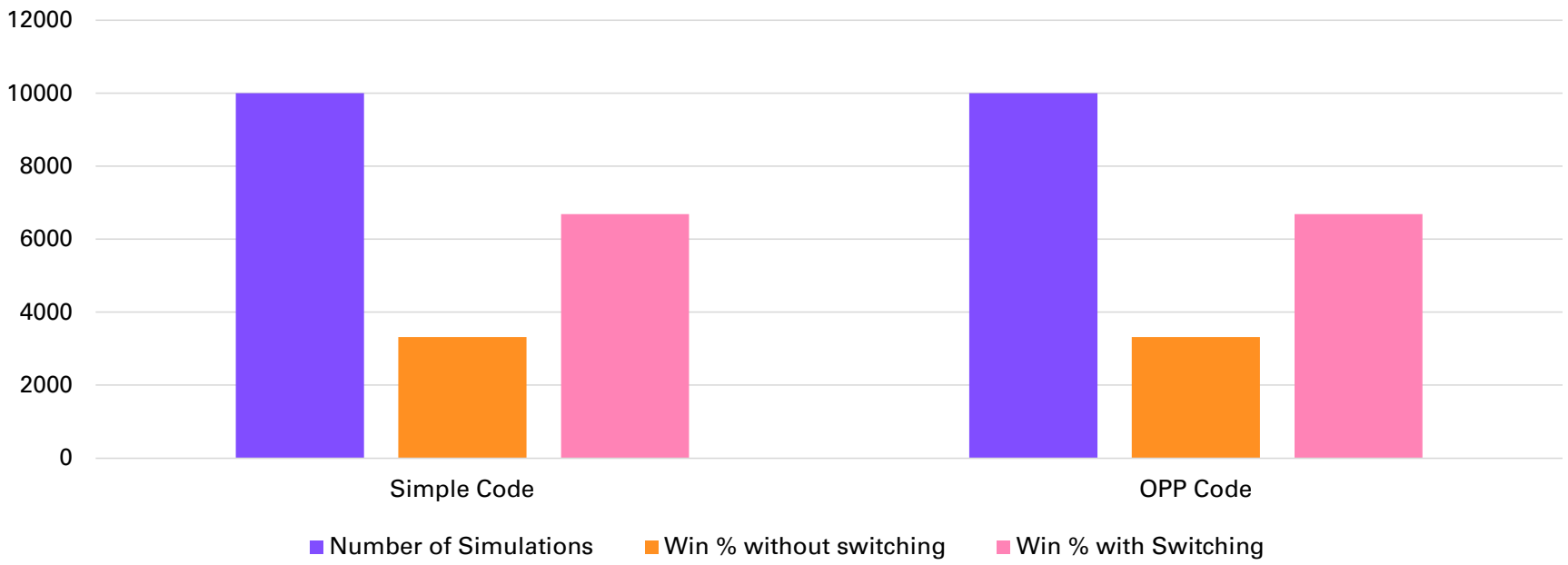


Table: Probability

Behind door 1	Behind door 2	Behind door 3	Result if staying at door #1	Result if switching to the door offered
Goat	Goat	**Car**	Wins goat	**Wins car**
Goat	**Car**	Goat	Wins goat	**Wins car**
Car	Goat	Goat	**Wins car**	Wins goat

TEAM

THANK YOU

Manoj Nath Yogi
manoj.nathyogi@bison.howard.edu

Aryan Bagale
aryan.bagale@bison.howard.edu

Bijay Pandey
bijay.pandey@bison.howard.edu