
FIAP

# Fórmulas

- Permutações

- $P_k^n = \frac{n!}{(n-k)!}$, sem repetição
- $P_k^n = n^k$, com repetição

- Combinações

- $C_k^n = \frac{n!}{k!(n-k)!}$, sem repetição
- $C_k^n = \frac{(n+k-1)!}{k!(n-1)!}$, com repetição

# Fórmulas no Python

- Permutações

- $P_k^n = \frac{n!}{(n-k)!}$, sem repetição
- $P_k^n = n^k$, com repetição

- Pacotes itertools: https://docs.python.org/3/library/itertools.html

- Sem repetição

```
cores = ['preta', 'vermelha', 'amarela']

permutacoes = permutations(cores, 3)

for i in list(permutacoes):
  print(i)
```

$k = 3$

- Com repetição

```
cores = ['preta', 'vermelha', 'amarela']

perm_repeticao = product(cores, repeat=3)

for i in list(perm_repeticao):
  print(i)
```

$k = 3$

# Fórmulas no Python

- Permutações

- $P_k^n = n^k$, com repetição
- $P_k^n = \frac{n!}{(n-k)!}$, sem repetição

- O módulo “math” auxilia no calculo do fatorial: https://docs.python.org/3/library/math.html
- Fórmula na raça:

```
def formula_permutacao(n, r, repeticao = False):
  if repeticao is True:
    return n**r
  else:
    return factorial(n) / factorial(n-r)
```

- Utilizando o módulo math: python 3.8+: perm(n, k)

# Fórmulas no Python

- Combinações

- $C_k^n = \frac{n!}{k!(n-k)!}$, sem repetição
- $C_k^n = \frac{(n+k-1)!}{k!(n-1)!}$, com repetição

- Mostra as combinações

```
cores = ['verde', 'vermelha', 'roxa', 'azul']

combinacao = combinations(cores, 2)

for i in list(combinacao):
  print(i)
```

```
cores = ['verde', 'vermelha', 'roxa', 'azul']

combinacao_com_rep = combinations_with_replacement(cores, 2)

for i in list(combinacao_com_rep):
  print(i)
```

# Fórmulas no Python

- Combinações

- $C_k^n = \frac{n!}{k!(n-k)!}$, sem repetição
- $C_k^n = \frac{(n+k-1)!}{k!(n-1)!}$, com repetição

- Formula na raça:

```
def formula_combinacao(n, k, repeticao = False):
  if repeticao is False:
    return factorial(n) / (factorial(k) * factorial(n-k))
  else:
    return factorial(n+k-1) / (factorial(k) * factorial(n-1))
```

- Utilizando o módulo math: python 3.8+: comb(n, k)

# Permutação ou Combinação?

- Precisamos escolher uma senha contendo **6** caracteres, sendo que estes caracteres somente podem ser **números** e **não podem ser repetidos**.

  - Números podem escolhidos entre 0 – 9. Temos 10 maneiras para arranjar.
  - Temos que escolher 6 números
  - A ordem dos números importa? Pense comigo, uma senha 123456 é a mesma que 654321?

  - Este é um problema de permutação sem repetição.
  - $P_6^{10} = \frac{10!}{(10-6)!} = 151.200$ possíveis permutações diferentes de senhas

# Permutação ou Combinação?

- Resultado no Python

```
formula_permutacao(n = 10, # 10 maneiras de arranjar
                   r = 6, # escolher 6 numeros
                   repeticao = False # nao pode repetir o numero
                   )
```

```
151200.0
```

# Permutação ou Combinação?

- Na Megasena precisamos escolher **6** números dentro de **60** possibilidades, quando um número é sorteado, ela **não pode aparecer novamente**.

  - Números podem escolhidos entre 1 – 60.
  - Temos que escolher 6 números
  - A ordem do jogo importa? Pense comigo, uma aposta (50, 20, 31, 37, 10, 45) é a mesma que (20, 31, 50, 10, 45, 37)?

  - Este é um problema de combinação sem repetição.
  - $C_6^{60} = \frac{60!}{6!(60-6)!}$ possíveis combinações.
    Aproximadamente 50 milhões de combinações.

# Permutação ou Combinação?

- Resultado no Python.

```
formula_combinacao(n = 60, # 60 maneiras de arranjar
                   k = 6, # escolher 6 numeros
                   repeticao = False # nao pode repetir o numero
                   )
```

```
50063860.0
```

# Resolução 1

- Desafio implícito: quantas regras teremos que criar para organizar esses bancos e termos uma classificação única (Opt-In, Opt-Out e Drop-Out) para cada cliente?

| Gerenciamento de entrada 1 | Gerenciamento de entrada 2 | Gerenciamento de saída 1 | Gerenciamento de saída 2 |
|---|---|---|---|
| Opt-In | Opt-In | Unsubscribe | Unsubscribe |
| Opt-Out | Opt-Out | - | - |

- Neste exemplo temos um problema de combinação, uma vez que a ordem do gerenciamento de entrada não importa. Além disto, não há como um cliente dar unsubscribe de um email não recebido.
- Temos n = 8 elementos a serem arranjados em k = 4 posições.
- Deste modo, teríamos que criar $C_4^8 = \frac{8!}{4!(8-4)!} = 70$ regras para classificar todos os 40k clientes em Opt-In, Opt-Out e Drop-Out.

# Resolução 1

- Resultado no Python

```
formula_combinacao(n = 8,
                   k = 4,
                   repeticao = False
                   )
```

```
70.0
```


THE WAY WE ARE