Exemplos - aula 2

Site: Moodle Unicamp
Curso: T_TT007B_2020S2 - Tópicos Especiais em Telecomunicações III
Livro: Exemplos - aula 2

Impresso por: Arthur Briganti Gini 213253
Data: sexta, 9 Out 2020, 11:22

Sumário

1. Exemplo 1

```
Um número primo é um inteiro maior que 1 que só é divisível por um e por ele mesmo.
Escreva uma função que determine se seu parâmetro é primo ou não, retornando
True se for, e False caso contrário. Escreva um programa principal que leia um inteiro
do usuário e exibe uma mensagem indicando se é primo ou não. Garantir
que o programa principal não será executado se o arquivo que contém sua solução for importado
por outro programa.
```

2. Solução 1

```
##
# Determine if a number entered by the user is prime.
#

## Determine whether or not a number is prime
# @param n the integer to test
# @return True if the number is prime, False otherwise
def isPrime(n):
  if n <= 1:
    return False

  # Check each number from 2 up to but not including n to see if it divides evenly into n
  for i in range(2, n):
    if n % i == 0:
      return False
  return True

# Determine if a number entered by the user is prime
def main():
  value = int(input("Enter an integer: "))
  if isPrime(value):
    print(value,"is prime.")
  else:
    print(value,"is not prime.")

# Call the main function if the file has not been imported
if __name__ == "__main__":
  main()
```

If `n % i == 0` then `n` is evenly divisible by `i`, indicating that `n` is not prime.

3. Exemplo 2

Neste exercício, você escreverá uma função que determina se uma senha é boa ou não. Definiremos uma boa senha com pelo menos 8 caracteres e contém pelo menos uma letra maiúscula, uma letra minúscula e um número. Sua função deve retornar verdadeiro se a senha passada para ela como seu único parâmetro é bom. Caso contrário, ele deve retornar falso. Inclui um programa principal que lê uma senha do usuário e informa se é ou não boa.

4. Solução 2

```
##
# Check whether or not a password is good.
#

## Check whether or not a password is good.  A good password is at least 8 characters
# long and contains an uppercase letter, a lowercase letter and a number.
# @param password the password to check
# @return True if the password is good, False otherwise
def checkPassword(password):
  has_upper = False
  has_lower = False
  has_num = False

  # Check each character in the password and see which requirement it meets
  for ch in password:
    if ch >= "A" and ch <= "Z":
      has_upper = True
    elif ch >= "a" and ch <= "z":
      has_lower = True
    elif ch >= "0" and ch <= "9":
      has_num = True

  # If the password has all 4 properties
  if len(password) >= 8 and has_upper and has_lower and has_num:
    return True

  # The password is missing at least on property
  return False

# Demonstrate the password checking function
def main():
  p = input("Enter a password: ")
  if checkPassword(p):
    print("That's a good password.")
  else:
    print("That isn't a good password.")

# Call the main function only if the file has not been imported into another program
if __name__ == "__main__":
  main()
```

5. Exemplo 3

A data mágica é uma data onde o dia multiplicado pelo mês é igual as dois últimos dígitos do ano. Por exemplo, 10 de junho de 1960 é uma data mágica porque junho é o sexto mês, e
6 vezes 10 é 60, que é igual ao ano com dois dígitos. Escreva uma função que determina se uma data é ou não uma data mágica. Use sua função para criar um programa principal
que encontra e exibe todas as datas mágicas do século XX.

6. Solução 3

```
##
# Determine all of the magic dates in the 1900s
#
from days_in_month import daysInMonth

## Determine whether or not a date is ''magic''
# @param day the day portion of the date
# @param month the month portion of the date
# @param year the year portion of the date
# @return True if the date is magic, False otherwise
def isMagicDate(day, month, year):
  if day * month == year % 100:
    return True

  return False

# Find and display all of the magic dates in the 1900s
def main():
  for year in range(1900, 2000):
    for month in range(1, 13):
      for day in range(1, daysInMonth(month, year) + 1):
        if isMagicDate(day, month, year):
          print("%02d/%02d/%04d is a magic date." % (day, month, year))

# Call the main function
main()
```

The expression `year % 100` evaluates to the two digit year.