Módulo 1: Variáveis, Operações, Condições

Logística:

- Pressione **Shift + Enter** para executar cada bloco de código no ambiente do Google Colab ou copie e cole os códigos/comentários em um arquivo python.
- As apresentações e explicações conceituais são fornecidas em blocos de texto. As explicações relacionadas aos códigos são incluídas como comentários acima dos códigos.
- Exercícios/problemas práticos são indicados em cada módulo. Para exercícios de codificação, você pode baixar uma IDE externa de Python (por exemplo, Anaconda) para programar e testar sua implementação. Uma possível implementação do exercício é fornecida sob o problema.

Objetivos de aprendizado:

1. Definir e modificar variáveis de vários tipos de dados. Converter entre tipos de dados.
2. Compreender as características e usos de cada operação e a saída (output) correspondente.
3. Compreender e corrigir estas declarações para verificar as condições.

1.1: Variáveis

1.1.1: Atribuição de Variáveis

As variáveis são as mesmas que as variáveis em matemática, exceto que variáveis matemáticas são muitas vezes letras, mas variáveis de programação podem ser palavras.

Variáveis: um conjunto que contém algumas informações.

Nota sobre a declaração de variáveis:

- *Case-sensitive* (ou seja, caracteres em caixa alta ou baixa são tratados como diferentes)
- DEVE começar com uma letra ou um sublinhado; NÃO PODE começar com números.
- NÃO PODE ser o mesmo nome que as palavras-chave Python (por exemplo, `class` , `finally` , etc.)
- NÃO especificar o tipo de informação armazenada na variável. *(Consulte os seguintes códigos para um exemplo.)*

In [ ]:
```
# Exemplos de declarações de variáveis
width = 10

# Note que o "H" está em letra maiúscula
Height = 5

area = 0
```

In [ ]:
```
width
```

Out[ ]:
```
10
```

In [ ]:
```
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-d56756f3e5e3> in <module>()
      2 # ERROR CODE: "height" is not defined.
      3
----> 4 height

NameError: name 'height' is not defined
```

In [ ]:
```
Height
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-e6498a6da099> in <module>()
----> 1 Height

NameError: name 'Height' is not defined
```

In [ ]:
```
# Usando uma palavra-chave python como um nome variável
# ERROR CODE: sintaxe inválida

global = 1

global
```

In [ ]:
```
# Mais declarações para diferentes tipos de variáveis

# armazenando uma string
helloMessage = "Hello World!"
first_name = "John"

# armazenando um caractere
character_example = 'a'

# armazenando um float
_newFloat = 1.0

# armazenando um valor booleano
bool_Condition = True
```

In [ ]:
```
helloMessage
```

In [ ]:
```
character_example
```

In [ ]:
```
_newFloat
```

In [ ]:
```
bool_Condition
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-999c5ab5a2cb> in <module>()
----> 1 bool_Condition

NameError: name 'bool_Condition' is not defined
```

1.1.2: Conversor de Tipo

A partir do tópico 1.1.1, aprendemos como declarar corretamente um nome variável para diferentes tipos de dados. Neste tópico, vamos explorar como "computar" ou converter o tipo dos dados entre si.

Uma função útil: `type()` define o tipo dos dados

In [ ]:
```
# A partir da declaração acima, width = 10 e 10 é int, então esperamos que a função r
etorne int
type(width)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-1b5d9269b43f> in <module>()
      1 # A partir da declaração acima, width = 10 e 10 é int, então esperamos que a
 função retorne int
----> 2 type(width)

NameError: name 'width' is not defined
```

In [ ]:
```
type(helloMessage)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-2cbfd6484f50> in <module>()
----> 1 type(helloMessage)

NameError: name 'helloMessage' is not defined
```

In [ ]:
```
type(bool_Condition)
```

In [ ]:
```
# Vamos computar um float em um int e vice-versa
# Computaremos o tipo e depois o armazenaremos em uma nova variável
width_float = float(width)

type(width_float)
```

In [ ]:
```
# Computar de float para int
width_int = int(width_float)

type(width_int)
```

In [ ]:
```
# Computar entre string e int
# Lembrar que width armazena um int

# Converter width para string
width_string = str(width)
type(width_string)
```

In [ ]:
```
# Converter width_string de volta a um int
type(int(width_string))
```

1.2: Operações

1.1.1 Operadores Aritméticos

In [ ]:
```
# Operações matemáticas básicas com Números

# Adição
print(5+23)

# Subtração
print(100-25)

# Multiplicação
print(5*10)

# Potência/Exponente
# o operador ** é equivalente ao expoente
print(5**2)

# 5*5 = 5^2 = 5**2
print(5*5)


# Divisão (float)
# Retornar o valor decimal real da divisão
print(36/4)
print(10/3)

# Divisão (int)
# Retornar um int. Se o quociente real for um valor decimal, apenas um número inteiro irá retornar
print(10//3)
print(19//6)

# Divisão Modular: retornar o restante da divisão
print(10%3)
```

```
28
75
50
25
25
9.0
3.3333333333333335
3
3
1
```

In [ ]:
```
# Operações com Strings e Caracteres
print("foo" * 5)
print('x'*3)
```

```
foofoofoofoofoo
xxx
```

In [ ]:
```
# ERRO: o compilador trata x como uma variável, não como um caractere
print(x*3)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-47a2cb16f654> in <module>()
      1 # ERROR: compiler treats x as a variable, not a character
----> 2 print(x*3)

NameError: name 'x' is not defined
```

In [ ]:
```
# ERRO: não pode concatenar um int com uma string --> necessidade de computar int com uma string
print("hello" + 5)
```

In [ ]:
```
# Fix
print("hello " + str(5))
```

In [ ]:
```
# Adição de String = concatenação
print("hello " + "world")
```

1.1.2: Outros Operadores

In [ ]:
```
# Comparadores: retornar valor booleano

# Igualdade ==
# Nota: DEVE ser sinais de igual Duplos, um sinal de igual único é atribuição
print(5 == 5.0)

# Maior do que >
print(7 > 1)

# Menor do que <
print(1.5 < 90)

# Maior ou igual a >=
print(5.0 >= 5)
print(5.0 >= 4)
print(5 >= 13)

# Menor ou igual a <=
print(10 <= 10.0)
print(10 <= 20)
print(8 <= 3)
```

In [ ]:
```
# Comparadores em Strings

print("hello" < "world")
print("hello" == "world")
print("hello" > "world")

print("hello" == "hello")

print("cat" < "dog")
```

1.3: Declarações Condicionais

1.3.1: Estrutura Condicional `If`

Avisos importantes:

- A ordem das condições importa!
  - Se mais de uma condição estiver satisfeita, então as ações associadas à primeira condição satisfeita serão executadas e saltarão as demais condições e códigos.
- " `elif` " = " `else if` "
  - " `elif` " expressa o mesmo significado que " `else if` "
- Pelo menos uma condição DEVE ser prevista para ambas as cláusulas `if` e `elif` , senão **ERROR**!
- Os parênteses para `if` e `elif` são opcionais. Seu código funcionará com ou sem o ().

In [ ]:
```
x = 7
y = 14


if (2*x == y):
  print("y is double of x")
elif (x**2 == y):
  print("y is the squared of x")
else:
  print("y is NOT double of x")
```

```
y is double of x
```

In [ ]:
```
x = 7
y = 49


if (2*x == y):
  print("y is double of x")
elif (x**2 == y):
  print("y is the squared of x")
else:
  print("y is NOT related to x")
```

```
y is the squared of x
```

In [ ]:
```
x = 7
y = 50


if (2*x == y):
  print("y is double of x")
elif (x**2 == y):
  print("y is the squared of x")
else:
  print("y is NOT double nor squared of x")
```

```
y is NOT double nor squared of x
```

1.3.2: Switch Cases

O Python NÃO tem uma implementação para os `switch cases` , mas uma maneira de implementá-lo é com o dicionário, uma estrutura de dados que armazena o par de valores chave (Módulo 3).

- As condições `switch` são armazenadas como chaves no dicionário, e as ações armazenadas como valor.
  - Se houver uma série de ações para cada caso, então considere escrever uma função para cada caso e use as chamadas de função como o valor.
- A condição padrão é listada manualmente como um valor chave no `get()` .

In [ ]:
```
def switcher(number):

  # Use dicionário (do Módulo 3) para armazenar switch cases
  # Se não for encontrado, o get() será o valor padrão
  return {
    '0':"Entered 0",
    '1':"Entered 1",
    '2':"Entered 2",
    '3':"Entered 3",
    '4':"Entered 4",
    '5':"Entered 5",
    '6':"Entered 6",
    '7':"Entered 7",
    '8':"Entered 8",
    '9':"Entered 9",
  }.get(number,"Invalid number!")


# input() lê uma entrado do usuário de stdin
number = input("Dial a number")
switcher(number)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _input_request(sel
f, prompt, ident, parent, password)
    728             try:
--> 729                 ident, reply = self.session.recv(self.stdin_socket, 0)
    730             except Exception:

/usr/local/lib/python3.7/dist-packages/jupyter_client/session.py in recv(self, socke
t, mode, content, copy)
    802         try:
--> 803             msg_list = socket.recv_multipart(mode, copy=copy)
    804         except zmq.ZMQError as e:

/usr/local/lib/python3.7/dist-packages/zmq/sugar/socket.py in recv_multipart(self, fl
ags, copy, track)
    582         """
--> 583         parts = [self.recv(flags, copy=copy, track=track)]
    584         # have first part already, only loop while more to receive

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.recv()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._recv_copy()

/usr/local/lib/python3.7/dist-packages/zmq/backend/cython/checkrc.pxd in zmq.backend.
cython.checkrc._check_rc()

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-17-8531db5f3174> in <module>()
     18
     19 # input() reads in an user input from stdin
---> 20 number = input("Dial a number")
     21 switcher(number)

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in raw_input(self, pro
mpt)
    702             self._parent_ident,
    703             self._parent_header,
--> 704             password=False,
    705         )
    706

/usr/local/lib/python3.7/dist-packages/ipykernel/kernelbase.py in _input_request(sel
f, prompt, ident, parent, password)
    732             except KeyboardInterrupt:
    733                 # re-raise KeyboardInterrupt, to truncate traceback
--> 734                 raise KeyboardInterrupt
    735             else:
    736                 break

KeyboardInterrupt:
```

In [ ]:

```
"""
EXERCÍCIO: implemente o exemplo de switch case acima usando as condições "if/else"

Prompt: para cada dígito entre 0-9, o programa imprimirá uma confirmação
para o valor inserido ou irá imprimir "invalid inputs" para todos os outros números.
"""
```

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

NÃO VEJA A SOLUÇÃO ABAIXO ANTES DE TENTAR FAZER O EXERCÍCIO!

In [ ]:
```
number = input("Dial a number")

if number == '0':
    print("Entered 0")
elif number == '1':
    print("Entered 1")
elif number == '2':
    print("Entered 2")
elif number == '3':
    print("Entered 3")
elif number == '4':
    print("Entered 4")
elif number == '5':
    print("Entered 5")
elif number == '6':
    print("Entered 6")
elif number == '7':
    print("Entered 7")
elif number == '8':
    print("Entered 8")
elif number == '9':
    print("Entered 9")
else:
    print("Invalid number!")
```