Python 101

Introducción superficial a Python

M. Torre Castro[1]

[1]Python Coruña

Charlas Python Coruña, 2023

Índice

Outline

Que é unha linguaxe de programación

Como comunicarse cunha máquina...

- Unha linguaxe de programación dicta unha maneira de transmitirlle instruccións a unha máquina
- Polo xeral un programa é un conxunto de liñas, onde cada liña é unha instrucción pra a máquina
- Os programas manexan a información como valores

Que é unha linguaxe de programación

Como comunicarse cunha máquina...

- Unha linguaxe de programación dicta unha maneira de transmitirlle instruccións a unha máquina
- Polo xeral un programa é un conxunto de liñas, onde cada liña é unha instrucción pra a máquina
- Os programas manexan a información como valores

Que é unha linguaxe de programación

Como comunicarse cunha máquina...

- Unha linguaxe de programación dicta unha maneira de transmitirlle instruccións a unha máquina
- Polo xeral un programa é un conxunto de liñas, onde cada liña é unha instrucción pra a máquina
- Os programas manexan a información como valores

O programa máis simple: Hola mundo

```
print('Hola mundo')
```

Explicación

- A instrucción que se lle da a máquina é mostrar/"imprimir" por pantalla un valor
- O valor neste caso é un texto: "Hola mundo"
- Executamos o programa e a máquina cumpre as instruccións mostrando "Hola mundo"

Explicación

- A instrucción que se lle da a máquina é mostrar/"imprimir" por pantalla un valor
- O valor neste caso é un texto: "Hola mundo"
- Executamos o programa e a máquina cumpre as instruccións mostrando "Hola mundo"

Explicación

- A instrucción que se lle da a máquina é mostrar/"imprimir" por pantalla un valor
- O valor neste caso é un texto: "Hola mundo"
- Executamos o programa e a máquina cumpre as instruccións mostrando "Hola mundo"

Un exemplo simplificado de programa: Unha calculadora

- Cómo se programaría unha calculadora?
- As instruccións do noso programa dinlle a máquina qué operacións matemáticas facer e como
- A maioría dos datos neste caso son números. Pra que un programa sexa lexible, normalmente asociamos os datos a nomes (= variables)
- As instruccións actuan sobre os datos/variables

Un exemplo simplificado de programa: Unha calculadora

- Cómo se programaría unha calculadora?
- As instruccións do noso programa dinlle a máquina qué operacións matemáticas facer e como
- A maioría dos datos neste caso son números. Pra que un programa sexa lexible, normalmente asociamos os datos a nomes (= variables)
- As instruccións actuan sobre os datos/variables

Un exemplo simplificado de programa: Unha calculadora

- Cómo se programaría unha calculadora?
- As instruccións do noso programa dinlle a máquina qué operacións matemáticas facer e como
- A maioría dos datos neste caso son números. Pra que un programa sexa lexible, normalmente asociamos os datos a nomes (= variables)
- As instruccións actuan sobre os datos/variables

Un exemplo simplificado de programa: Unha calculadora

- Cómo se programaría unha calculadora?
- As instruccións do noso programa dinlle a máquina qué operacións matemáticas facer e como
- A maioría dos datos neste caso son números. Pra que un programa sexa lexible, normalmente asociamos os datos a nomes (= variables)
- As instruccións actuan sobre os datos/variables

Como operar con datos: Variables

```
number1 = 10
number2 = 2

result = number1 + number2
print('Resultado: ', result)

result = number1 - number2
print('Resultado: ', result)

result = number1 * number2
print('Resultado: ', result)

result = number1 / number2
print('Resultado: ', result)
```

Tipos de datos básicos

- Os valores que podemos asociar a variables pra operar con eles teñen tipos. Un valor pode ser:

  int O valor é un número entero, sen decimais. Exemplos: 5, -12, 58...

  float O valor é un número decimal. Exemplos: 3.14, -5.25, 75.5

  str O valor son un ou máis caracteres. Se representan entre comillas ou Python non os recoñece. Exemplo: 'Hola mundo', "Resultado: "

  boolean Un valor lóxico, que pode valer True (verdadeiro) ou False (falso).

Tipo de dato string: Texto/cadenas de caracteres

```
# Isto é un comentario.
# Python ignora todo ata final de liña

question = 'Introduza o seu nome, por favor: '

# Isto recupera o valor introducido polo usuario
# e o asocia á variable <name>
name = input(question)
message = 'Gracias, ' + name + ' por introducir o teu nome'
```

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: <, <=, >, >=, ==, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: 6 < 1000 # Evalua a True
- Exemplo: 12 > 112 # Evalua a False
- Exemplo: 'Patata' == 'Batata' # Evalua a False
- Exemplo: 'Patata' < 'Patatas' # True
- Exemplo: 'Batata' <= 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' == 'Batata' # Evalua a False
- Exemplo: 'Patata' < 'Patatas' # True
- Exemplo: 'Batata' <= 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Tipo de dato boolean: Valores lóxicos

- Unha expresión booleana é calquera que teña como resultado True ou False (atención, Python é case sensitive)
- Operadores: $<$, $<=$, $>$, $>=$, $==$, not
- Exemplo: not True # Evalua a False e viceversa
- Exemplo: $6 < 1000$ # Evalua a True
- Exemplo: $12 > 112$ # Evalua a False
- Exemplo: 'Patata' $==$ 'Batata' # Evalua a False
- Exemplo: 'Patata' $<$ 'Patatas' # True
- Exemplo: 'Batata' $<=$ 'Patatas'

Comparacións (I)

Pra que serven os valores lóxicos

```
fire = input('Está o fogo encendido? (S/N): ')
temperature = input('Introduza a temperatura do auga: ')

# is_boiling = temperature <= 100
# TypeError:
# '<=' not supported between instances of 'int' and 'str'

is_boiling = int(temperature) <= 100 # True ou False

# Atención: indentacións!
if is_boiling:
    print('O auga está fervendo')
elif fire == 'N':
    print('Debe encender o fogo pra que o auga ferva')
else:
    print('O auga ainda non ferve. Espere por favor')
```

Comparacións (II)

```
# Queremos vender entradas de 10, 18 e 30 €
available_money = int(input('Introduza cantos € ten: '))
enough_money = (available_money >= 10)

if enough_money:
    if available_money >= 30:
        print('Pode adquirir a entrada de 30 €')
    elif available_money >= 18:
        print('Pode adquirir a entrada de 18 €')
    else:
        print('Pode adquirir a entrada de 10 €')
else:
    print('Non se pode permitir ningunha entrada')
```

Estructuras de datos: Listas

- As listas son conxuntos de datos ordenados que podemos asociar a variables e se denotan entre corchetes []
- Isto pode pensarse como que onde antes asociabamos un valor a un nome, agora asociamos moitos a un nome
- Podemos almacenar datos de distintos tipos en cada posición e agregar ou quitar datos en posicións específicas
- As posicións cóntanse desde 0

Estructuras de datos: Listas

- As listas son conxuntos de datos ordenados que podemos asociar a variables e se denotan entre corchetes []
- Isto pode pensarse como que onde antes asociabamos un valor a un nome, agora asociamos moitos a un nome
- Podemos almacenar datos de distintos tipos en cada posición e agregar ou quitar datos en posicións específicas
- As posicións cóntanse desde 0

Estructuras de datos: Listas

- As listas son conxuntos de datos ordenados que podemos asociar a variables e se denotan entre corchetes []
- Isto pode pensarse como que onde antes asociabamos un valor a un nome, agora asociamos moitos a un nome
- Podemos almacenar datos de distintos tipos en cada posición e agregar ou quitar datos en posicións específicas
- As posicións cóntanse desde 0

Estructuras de datos: Listas

- As listas son conxuntos de datos ordenados que podemos asociar a variables e se denotan entre corchetes []
- Isto pode pensarse como que onde antes asociabamos un valor a un nome, agora asociamos moitos a un nome
- Podemos almacenar datos de distintos tipos en cada posición e agregar ou quitar datos en posicións específicas
- As posicións cóntanse desde 0

Listas

```
# Exemplo de axenda
schedule_list = ['Espertar', 'Duchar', 'Traballar', 'Cear', ]
print(schedule_list[0])

schedule_list.append('Durmir')
schedule_list.insert(1, 'Almorzar')
schedule_list[3] = 'Descansar'

second_element = schedule_list.pop(1) # 'Almorzar'
del(schedule_list[3]) # Elimina 'Cear'
```

Iterar listas

```
schedule_list = ['Espertar', 'Duchar',
                 'Descansar', 'Cear', 'Durmir']

for element in schedule_list:
    print(element)
```

Bucle for

Estructuras de control repetitivas

```
# Suma dos 20 primeiros numeros (do 0 ao 19)
suma = 0
for i in range(20):
    suma = suma + i

print(suma)
```

Bucle while

Estructuras de control repetitivas

```
# Exemplo dun "reloxo" con alarma
time = 0
alarm_time = 15

while time < alarm_time:
    if time < 12:
        print(time + ' AM')
    else:
        print(time + ' PM')

    time = time + 1

print('Alarma!')
```

Funcións (I)

- As funcións son bloques de código reusables
- Poden chamarse repetidas veces e recibir valores distintos en cada chamada
- Levan un nome, como as variables, que ben usado axuda a ter un código lexible

Funcións (I)

- As funcións son bloques de código reusables
- Poden chamarse repetidas veces e recibir valores distintos en cada chamada
- Levan un nome, como as variables, que ben usado axuda a ter un código lexible

Funcións (I)

- As funcións son bloques de código reusables
- Poden chamarse repetidas veces e recibir valores distintos en cada chamada
- Levan un nome, como as variables, que ben usado axuda a ter un código lexible

Funcións (II)

```
def show_mark(alumn, mark):
    print(alumn + ' sacou un ' + mark + ' no exame')

marks_list = [5, 7, 3, 10, 9, ]
alumns_list = ['Ana', 'Paco', 'Bea', 'Pepe', 'Suso', ]
size = len(alumn_list)

for i in range(size):
    show_mark(alumns_list[i], marks_list[i])

# Imaxinade que a lista fose de 40 elementos!
```

Resumo

- Isto foi unha introducción sinxela e só inclue as ferramentas básicas. Hai moito ainda por descubrir
- Outras estructuras de datos (diccionarios, conxuntos, etc...)
- Acceso a BDs, ficheiros, POO
- Manexo de excepcións. Módulos e librerías

Resumo

- Isto foi unha introducción sinxela e só inclue as ferramentas básicas. Hai moito ainda por descubrir
- Outras estructuras de datos (diccionarios, conxuntos, etc...)
- Acceso a BDs, ficheiros, POO
- Manexo de excepcións. Módulos e librerías

Resumo

- Isto foi unha introducción sinxela e só inclue as ferramentas básicas. Hai moito ainda por descubrir
- Outras estructuras de datos (diccionarios, conxuntos, etc...)
- Acceso a BDs, ficheiros, POO
- Manexo de excepcións. Módulos e librerías

Resumo

- Isto foi unha introducción sinxela e só inclue as ferramentas básicas. Hai moito ainda por descubrir
- Outras estructuras de datos (diccionarios, conxuntos, etc...)
- Acceso a BDs, ficheiros, POO
- Manexo de excepcións. Módulos e librerías

Pra ler máis I

- https://docs.python.org/3
- https://tutorialesprogramacionya.com/pythonya/index.php