

03.1_ScalarDataTypes

September 6, 2023

1 Introduction to Python for Open Source Geocomputation



- Instructor: Dr. Wei Kang

Content:

- Scalar data types

2 Activities - *Translate that!*

- I will select students randomly to interpret what I have said in the lecture - we will write a python program to ensure the randomness
- Many students will be called on during the activity - our python program will ensure not one student is selected more than once.
- We will have this activity throughout this class.

3 Data Type

- Classification or categorization of knowledge items
- Associated with specific operations that are often performed on that data type.
 - * - + for numerical data type
 - `append` for string data type
 - `for` loop for container data types

3.1 Standard Data Types in Python

Category of Data type	Data type	Example
Numeric, scalar	Integer	1
	Floats	1.2
	Complex	1.5+0.5j
	Booleans	True
Container	strings	"Hello World"
	List	[1, "Hello World"]
	Tuple	(1, "Hello World")
	Set	{1, "Hello World"}
	Dictionary	{1: "Hello World", 2: 100}

3.2 Numeric, Scalar types

Category of Data type	Data type	Example
Numeric, scalar	Integer	1
	Floats	1.2
	Complex	1.5+0.5j
	Booleans	True

- `type()` function: examine the data type
- Understanding data types are important
 - associated operators or functions
 - some are not compatible with one another

3.2.1 Integer

```
[1]: 1 + 1
```

```
[1]: 2
```

```
[2]: a = 4
     type(a)
```

```
[2]: int
```

3.2.2 Floats

```
[3]: c = 2.1
     type(c)
```

```
[3]: float
```

```
[4]: c1 = 2.0
     type(c1)
```

```
[4]: float
```

3.2.3 Complex

```
[5]: a = 1.5 + 0.5j
[6]: type(a)
[6]: complex
[7]: a.real
[7]: 1.5
[8]: a.imag
[8]: 0.5
[9]: a
[9]: (1.5+0.5j)
[10]: a.real
[10]: 1.5
[11]: type(1.5)
[11]: float
[12]: type(a.real)
[12]: float
[13]: type(a.imag)
[13]: float
```

3.2.4 Booleans

- Represent truth values
- Can take one of two possible values: **True** and **False**
- Very useful in conditional execution
- Ways of creating a Booleans variable
 - *Assignment Statements*
 - Python function `bool()`
 - * returns **False** if the input is 0 or any empty string or list
 - * returns **True** otherwise
 - *Logical and comparison expressions*

```
[14]: a = True
      type(a)
```

```
[14]: bool
```

```
[15]: b = False
      type(b)
```

```
[15]: bool
```

```
[16]: a = true
```

```
-----
NameError                                Traceback (most recent call last)
Cell In[16], line 1
----> 1 a = true

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

```
[17]: bool(0)
```

```
[17]: False
```

```
[18]: bool(2.2)
```

```
[18]: True
```

```
[19]: bool("happy")
```

```
[19]: True
```

```
[20]: bool("")
```

```
[20]: False
```

```
[21]: bool(" ")
```

```
[21]: True
```

3.2.5 *Translate that!*

What is a boolean data type in python?

3.2.6 Comparison operators

<, <=, >, >=, ==, !=

- compare two objects and return either True or False

- compare both numbers and strings
- smaller than, smaller or equal, greater than, greater or equal, equal, not equal

```
[22]: x = 3  
      y = 5  
      x > y
```

```
[22]: False
```

```
[23]: x == y
```

```
[23]: False
```

```
[24]: x != y
```

```
[24]: True
```

We can assign the truth value of a comparison operation to a new variable:

```
[25]: z = x > y
```

```
[26]: z
```

```
[26]: False
```

```
[27]: z = (x > y)
```

```
[28]: z
```

```
[28]: False
```

```
[29]: type(z)
```

```
[29]: bool
```

```
[30]: x
```

```
[30]: 3
```

```
[31]: y
```

```
[31]: 5
```

```
[32]: z = x < y  
      z
```

```
[32]: True
```

```
[33]: 1 == 1
```

```
[33]: True
```

```
[34]: 1 != 1
```

```
[34]: False
```

3.2.7 Comparing strings

- Python string comparison compares the characters in both strings one by one.
- When different characters are found, then their Unicode code point values are compared. The character with the lower Unicode value is considered to be smaller.
 - `ord()`: function to check the unicode
 - Order in the dictionary
 - * upper case alphabets, lower case alphabets, numbers
- If you compare strings that contain the same substring, such as "Apple" and "ApplePie", then the longer string is considered larger.

```
[35]: "apple" == "banan"
```

```
[35]: False
```

```
[36]: "apple" > "banan"
```

```
[36]: False
```

```
[37]: ord("a")
```

```
[37]: 97
```

```
[38]: ord("b")
```

```
[38]: 98
```

```
[39]: "apple" > "banan"
```

```
[39]: False
```

```
[40]: "apple" > "banana"
```

```
[40]: False
```

```
[41]: "apple" == "banan"
```

```
[41]: False
```

```
[42]: "apple" < "banan"
```

```
[42]: True
```

```
[43]: "apple" < "banana"
```

```
[43]: True
```

```
[44]: "apple" < "apply"
```

```
[44]: True
```

```
[45]: "apple" < "apple"
```

```
[45]: False
```

```
[46]: "apple" == "apple"
```

```
[46]: True
```

```
[47]: "1" > "a"
```

```
[47]: False
```

```
[48]: ord("1")
```

```
[48]: 49
```

```
[49]: ord("a")
```

```
[49]: 97
```

```
[50]: "apple" > "APPLE"
```

```
[50]: True
```

```
[51]: "1000" < "a"
```

```
[51]: True
```

```
[52]: "2" > "100"
```

```
[52]: True
```

```
[53]: 2 > 100
```

```
[53]: False
```

```
[54]: 2 > "100"
```

```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[54], line 1  
----> 1 2 > "100"  
  
TypeError: '>' not supported between instances of 'int' and 'str'
```

```
[55]: 1.3 > 1
```

```
[55]: True
```

3.2.8 *Translate that!*

What are comparison operators? What do they do?

3.2.9 Logical operators

and, or, not

- work just like English
- **and**: return **True** if both operands are true
- **or**: return **True** if either operands are true
- **not**: always negates the expression that follows

```
[56]: True and False
```

```
[56]: False
```

```
[57]: True or False
```

```
[57]: True
```

```
[58]: True and True
```

```
[58]: True
```

```
[59]: False and False
```

```
[59]: False
```

```
[60]: False or False
```

```
[60]: False
```

```
[61]: not True
```



```
[61]: False
```

```
[62]: 5>3 and 5<3
```

```
[62]: False
```

```
[63]: 5>3 or 5<3
```

```
[63]: True
```

```
[64]: not 5<3 or 5<3
```

```
[64]: True
```

```
[65]: (not 5<3) or 5<3
```

```
[65]: True
```

```
[66]: not 5<3 or 5>3
```

```
[66]: True
```

3.2.10 *Translate that!*

What are logical operators? What do they do?

3.2.11 Arithmetic operators on Boolean variables

```
[67]: True * False
```

```
[67]: 0
```

```
[68]: True + False
```

```
[68]: 1
```

```
[69]: True / False
```

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
Cell In[69], line 1  
----> 1 True / False  
  
ZeroDivisionError: division by zero
```

```
[70]: False/ True
```

```
[70]: 0.0
```

3.2.12 Order of the operators

1. Math
2. Comparison
3. not
4. and
5. or

use parentheses to indicate the order you want.

```
[71]: a = 5
      b = 3
      c = 10

      a < b or a < c and not b > c
```

```
[71]: True
```

```
[72]: (a < b) or ((a < c) and (not (b > c)))
```

```
[72]: True
```

```
[73]: a = 5
      b = 3
      c = 10
      a+b > c
```

```
[73]: False
```

```
[74]: a + (b > c)
```

```
[74]: 5
```

3.2.13 *Translate that!*

What are the order of arithmetic (e.g., +), comparison (e.g., >), and logical (e.g., “not”) operators?

3.3 Conversion between Numeric, Scalar Data Types in Python

Building blocks of container data types

Data type name	Description	Example	Conversion function
integer	Whole integer values	1	<code>int</code>
floats	Decimal values	1.2	<code>float</code>

Data type name	Description	Example	Conversion function
complex	Complex numbers	1.5+0.5j	<code>complex</code>
booleans	True/false values	True	<code>bool</code>

Converting an integer to other scalar types

```
[75]: a_int = 1
      type(a_int)
```

```
[75]: int
```

```
[76]: float(a_int)
```

```
[76]: 1.0
```

```
[77]: type(a_int)
```

```
[77]: int
```

```
[78]: type(float(a_int))
```

```
[78]: float
```

```
[79]: a_float = float(a_int)
      type(a_float)
```

```
[79]: float
```

```
[80]: a_int
```

```
[80]: 1
```

```
[81]: a_complex = complex(a_int)
      print(a_complex)
      type(a_complex)
```

```
(1+0j)
```

```
[81]: complex
```

```
[82]: int(9.2)
```

```
[82]: 9
```

```
[83]: int(a_complex)
```

```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[83], line 1  
----> 1 int(a_complex)  
  
TypeError: int() argument must be a string, a bytes-like object or a real_  
↳number, not 'complex'
```

```
[84]: a_int
```

```
[84]: 1
```

```
[85]: a_boolean = bool(a_int)  
      print(a_boolean)
```

```
True
```

```
[86]: type(a_boolean)
```

```
[86]: bool
```

Converting other types to integer

```
[87]: int(1.2)
```

```
[87]: 1
```

```
[88]: int(1.8)
```

```
[88]: 1
```

```
[89]: int(True)
```

```
[89]: 1
```

```
[90]: int(False)
```

```
[90]: 0
```

```
[91]: int(1+1j)
```

```
-----  
TypeError                                Traceback (most recent call last)  
Cell In[91], line 1  
----> 1 int(1+1j)
```

```
TypeError: int() argument must be a string, a bytes-like object or a real
↳number, not 'complex'
```

4 Next Class

- Topics:
 - conditionals
 - strings
- Readings:
 - Chapter 8