# 03.2_ScalarDataTypes

February 8, 2023

## 1 Introduction to Python for Open Source Geocomputation


- Instructor: Dr. Wei Kang
- Class Location and Time: ENV 336, Mon & Wed 12:30 pm - 1:50 pm

Content:

- Numerical Data Type

## 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: exmaine 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]:
```
c = 2.1
type(c)
```

[8]: float

[9]:
```
c.real
```

[9]: 2.1

[10]:
```
a.imag
```

[10]: 0.5

[11]:
```
c.imag
```

[11]: 0.0

[12]:
```
a
```

[12]: (1.5+0.5j)

[13]:
```
a.real
```

[13]: 1.5

[14]:
```
type(1.5)
```

[14]: float

[15]:
```
type(a.real)
```

[15]: float

[16]:
```
type(a.imag)
```

[16]:
```
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*

[17]:
```
a = True
type(a)
```

[17]:
```
bool
```

[18]:
```
b = False
type(b)
```

[18]:
```
bool
```

[19]:
```
a = true
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_24778/1480204962.py␣
 ↪in <module>
----> 1 a = true

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

[20]:
```
bool(0)
```

[20]:
```
False
```

[21]:
```
bool(2.2)
```

[21]:
```
True
```

[22]:
```
bool("happy")
```

[22]:
```
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

[23]:
```
x = 3
y = 5
x > y
```

[23]:
```
False
```

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

[24]:
```
False
```

[25]:
```
x != y
```

[25]:
```
True
```

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

[26]:
```
z = x > y
```

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

[28]:
```
z
```

[28]:
```
False
```

[29]:
```
(z = x) >y
```

```
  File "/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_24778/
↪2082343905.py", line 1
    (z = x) >y
       ^
SyntaxError: invalid syntax
```

[30]:
```
type(z)
```

[30]:
```
bool
```

[31]:
```
x
```

[31]: 3

[32]:
```
y
```

[32]: 5

[33]:
```
z = x < y
z
```

[33]: True

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

[34]: True

[35]:
```
1 != 1
```

[35]: False

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

[36]: False

[37]:
```
"apple" > "banana"
```

[37]: False

[38]:
```
"apple" == "banan"
```

[38]: False

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

[39]: True

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

[40]: True

[41]:
```
"apple" < "apply"
```

[41]: True

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

[42]: False

```
[43]: "apple" == "apple"
```

[43]: True

```
[44]: "1" > "a"
```

[44]: False

```
[45]: "1000" < "a"
```

[45]: True

### 3.2.7 *Translate that!*

What are comparison operators? What do they do?

### 3.2.8 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

```
[46]: True and False
```

[46]: False

```
[47]: True or False
```

[47]: True

```
[48]: True and True
```

[48]: True

```
[49]: False and False
```

[49]: False

```
[50]: False or False
```

[50]: False

```
[51]: not True
```

[51]: False

```
[52]: 5>3 and 5<3
```

[52]: False

```
[53]: 5>3 or 5<3
```

[53]: True

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

[54]: False

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

[55]: True

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

[56]: True

#### 3.2.9 *Translate that!*

What are logical operators? What do they do?

#### 3.2.10 Arithmetric operators on Boolean variables

```
[57]: True * False
```

[57]: 0

```
[58]: True + False
```

[58]: 1

```
[59]: True / False
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_24778/2917913318.py␣
 ↪in <module>
----> 1 True / False

ZeroDivisionError: division by zero
```

[60]:
```
False/ True
```

[60]:
```
0.0
```

### 3.2.11 Order of the operators

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

use parentheses to indicate the order you want.

[61]:
```
a = 5
b = 3
c = 10

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

[61]:
```
True
```

[62]:
```
(a < b) or ((a<c) and (not (b > c)))
```

[62]:
```
True
```

[63]:
```
a = 5
b = 3
c = 10
a+b > c
```

[63]:
```
False
```

[64]:
```
a + (b>c)
```

[64]:
```
5
```

### 3.2.12 *Translate that!*

What are the order of arithmetric (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 | `int` |
| floats | Decimal values | 1.2 | `float` |
| complex | Complex numbers | 1.5+0.5j | `complex` |
| booleans | True/false values | True | `bool` |

**Converting an integer to other scalar types**

[65]:
```
a_int = 1
type(a_int)
```

[65]:
```
int
```

[66]:
```
float(a_int)
```

[66]:
```
1.0
```

[67]:
```
type(a_int)
```

[67]:
```
int
```

[68]:
```
type(float(a_int))
```

[68]:
```
float
```

[69]:
```
a_float = float(a_int)
type(a_float)
```

[69]:
```
float
```

[70]:
```
a_int
```

[70]:
```
1
```

[71]:
```
a_complex = complex(a_int)
print(a_complex)
type(a_complex)
```

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

[71]:
```
complex
```

[72]:
```
int(a_complex)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
```

```
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_24778/3628853452.py␣
 ↪in <module>
----> 1 int(a_complex)

TypeError: can't convert complex to int
```

[73]:
```
a_int
```

[73]:
```
1
```

[74]:
```
a_boolean = bool(a_int)
print(a_boolean)
```

```
True
```

[75]:
```
type(a_boolean)
```

[75]:
```
bool
```

### Converting other types to integer

[76]:
```
int(1.2)
```

[76]:
```
1
```

[77]:
```
int(1.8)
```

[77]:
```
1
```

[78]:
```
int(True)
```

[78]:
```
1
```

[79]:
```
int(False)
```

[79]:
```
0
```

[80]:
```
int(1+1j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/var/folders/6m/8n2ktxl566j8yp0n_qx7x5bw0000gt/T/ipykernel_24778/3106831245.py␣
 ↪in <module>
----> 1 int(1+1j)

TypeError: can't convert complex to int
```

# 4 Next Class

- Git, GitHub and Visual Studio Code