Auswärtiges Amt

DAAD Deutscher Akademischer Austauschdienst
German Academic Exchange Service

Geo-IT Online Seminar
Freie Universität Berlin
Institute of Geographical Sciences

Freie Universität Berlin

TRAINING ON GOOGLE EARTH ENGINE MODULE 2

Prof. Dr. Eng. Ayman Abdulrahman

Coordinated from

Dr. Wahib Sahwan (Freie Universität Berlin)

Geo-IT

The Technology of Data Acquisition for Sustainable Development and Crisis Management

(Germany, Jordan, Lebanon and Syria)

MODULE 2 : PROGRAMMING in Google Earth Engine

OOP Object/Class (Objects & Methods)

- Client and Server
- GEE Numbers
- GEE String
- GEE Dictionary
- GEE List
- GEE Array
- GEE Date
- Examples and exercises

OBJECT

Real Life Objects, Properties, and Methods

In real life, a car is an object.

A car has properties like weight and color, and methods like start and stop:

Object	Properties	Methods
car	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

All cars have the same properties, but the property values differ from car to car.

All cars have the same methods, but the methods are performed at different times.

JavaScript variables are containers for data values
var car = "Fiat";

Objects are variables too. But objects can contain many values.
This code assigns many values (Fiat, 500, white) to a variable named car:

const car = {type:"Fiat", model:"500", color:"white"};

The values are written as name:value pairs (name and value separated by a colon).
It is a common practice to declare objects with the const keyword.

Object Definition

You define (and create) a JavaScript object with an object literal:

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Object Properties
The name:values pairs in JavaScript objects are called properties:

Property	Property Value
firstNameJohn	
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

```
objectName.propertyName
objectName["propertyName"]
```

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

JavaScript Comparison Operators

Operator	Description
==	equal to
!=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JavaScript Bitwise Operators
Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

ee.Number

Constructs a new number.
ee.Number(number) >>>>>>>> A number or a computed object.

Example :

```
print(ee.Number(0));  // 0
print(ee.Number(1));  // 1
print(ee.Number(0.0));  // 0
print(ee.Number(1.0));  // 1
print(ee.Number(-1.0));  // -1
print(ee.Number(Math.PI));  // 3.141592653589793
print(ee.Number(1.2e-35));  // 1.2e-35
print(ee.Number(3.4e10));  // 34000000000
```

ee.Number

Function (arithmatic) : add sutract maltiply divide mod

Fuction (operation) : abs cbrt exp log log10 pow sqrt round

Function (special) : ceil floor clamp hypot max min parse signum unitScale

Function (logic) : and or not
gt gte lt lte eq neq

Function (trigmetry) : sin cos tan tan2 acos asin atan cosh sinh tanh

Function (comply) erf erfinv erfc erfcinv gamma digamma trigamma lanczos evaluate

MATH

int float double short long

int8 int16 int32 int64

uint8 uint16 uint32

byte bitCount bitwiseAnd bitwiseOr bitwiseNot bitwiseXor

ee.String

Constructs a new String.
ee.String(string) >>>>>>> A string or a computed object.

```
print(ee.String('I am a string'));  // I am a string

// Strings can use emoji.
print(ee.String('  '));  //

// Empty string.
var empty = ee.String('');
print(empty);  // ''
print(empty.length());  // 0
```

ee.String

Functions :

String.length() Returns the length of a string.

String.index(pattern) Searches a string for the first occurrence of a substring. Returns the index of the first match, or -1.

String.replace(regex, replacement, flags) Returns a new string with some or all matches of a pattern replaced.

String.cat(string2) Concatenates two strings.

String.toLowerCase() : Converts all of the characters in a string to lower case.

String.toUpperCase() : Converts all of the characters in a string to upper case.

String.split(regex, flags) : Splits a string on a regular expression, Returning a list of strings.

String.slice(start, end) : Returns a substring of the given string. If the specified range exceeds the length of the string, returns a shorter substring.

String.compareTo(string2) :

String.match(regex, flags)

ee.Dictionary

Constructs a new Dictionary.
ee.Dictionary(dict) >>>>>>> dict : ComputedObject|Object, optional
An object to convert to a dictionary. This constructor accepts the following types:
1) Another dictionary.
2) A list of key/value pairs.
3) A null or no argument (producing an empty dictionary)

```
// A dictionary input (e.g. results of ee.Image.reduceRegion of an S2 image).
var dict = {
  B1: 182,
  B2: 219,
  B3: 443
};
// A list of key/value pairs (from previous dictionary).
var list = [
  'B1', 182,
  'B2', 219,
  'B3', 443
];
```

ee.Dictionary

Functions :

Dictionary.size() : Returns the number of entries in a dictionary.
Dictionary.keys() : Retrieve the keys of a dictionary as a list. The keys will be sorted in natural order.
Dictionary.values(keys) : Returns the values of a dictionary as a list. If no keys are specified, all values are returned in the natural ordering of the dictionary's keys.
Dictionary.get(key, defaultValue) : Extracts a named value from a dictionary. If the dictionary does not contain the given key, then defaultValue is returned, unless it is null.
Dictionary.select(selectors, ignoreMissing) : Returns a dictionary with only the specified keys.
Dictionary.set(key, value) : Set a value in a dictionary.
Dictionary.remove(selectors, ignoreMissing) : Returns a dictionary with the specified keys removed.
Dictionary.rename(from, to, overwrite) : Rename elements in a dictionary.
Dictionary.combine(second, overwrite) : Combines two dictionaries. In the case of duplicate names, the output will contain the value of the second dictionary unless overwrite is false. Null values in both dictionaries are ignored / removed.
Dictionary.contains(key) : Returns true if the dictionary contains the given key.

ee.Dictionary

Functions :

Dictionary.getInfo(callback) : Retrieves the value of this object from the server.
ee.Dictionary.fromLists(keys, values) : Construct a dictionary from two parallel lists of keys and values.
Dictionary.getNumber(key) : Extracts a named number value from a dictionary.
Dictionary.getString(key) : Extracts a named string value from a dictionary.
Dictionary.getArray(key) : Extracts a named array value from a dictionary.
Dictionary.getGeometry(key) : Extracts a named geometry value from a dictionary.
Dictionary.toArray(keys, axis) : Returns numeric values of a dictionary as an array. If no keys are specified, all values are returned in the natural ordering of the dictionary's keys. The default 'axis' is 0.
Dictionary.toImage(names) : Creates an image of constants from values in a dictionary. The bands of the image are ordered and named according to the names argument. If no names are specified, the bands are sorted alpha-numerically.

ee.List

Constructs a new list.
ee.List(list) >>>>>>>> A list or a computed object

Example :

```
print(ee.List([]));  // []
print(ee.List([null]));  // [null]
print(ee.List([true]));  // [true]
print(ee.List([1]));  // [1]
print(ee.List([ee.Number(1)]));  // [1]
print(ee.List(['a']));  // ["a"]
print(ee.List([[]]));  // [[]]
print(ee.List([ee.List([])]));  // [[]]
print(ee.List([[], [[]], [1, [], 'a']]));  // [[],[[]],[1,[],"a"]]
```

ee.List

List.length() : Returns the number of elements in list.
List.size() :
List.get(index) : Returns the element at the specified position in list. A negative index counts backwards from the end of the list
List.getNumber(index) List.getString(index) List.getArray(index) List.getGeometry(index)
List.indexOf(element) : Returns the position of the first occurrence of target in list, or -1 if list does not contain target.
List.add(element) : Appends the element to the end of list.
List.insert(index, element) : Inserts element at the specified position in list. A negative index counts backwards from the end of the ls
ee.List.sequence(start, end, step, count) : Generate a sequence of numbers from start to end (inclusive) in increments of step, or in count equally-spaced increments. If end is not specified it is computed from start + step * count, so at least one of end or count must be specified.
List.replace(oldval, newval) : Replaces the first occurrence of oldVal in list with newVal.
List.replaceAll(oldval, newval) : Replaces all occurrences of oldVal in list with newVal.
List.cat(other) : Concatenates the contents of other onto list.
List.slice(start, end, step) : Returns a portion of list between the start index, inclusive, and end index, exclusive.
List.splice(start, count, other) : Starting at the start index, removes count elements from list and insert the contents of other at that location. If start is negative, it counts backwards from the end of the list.
List.swap(pos1, pos2) : Swaps the elements at the specified positions. A negative position counts backwards from the end of the list
List.remove(element) : Removes the first occurrence of the specified element from list, if it is present.
List.removeAll(other)

ee.Date

Constructs a new Date object.

ee.Date(date, tz)

date **ComputedObject|Date|Number|String**
The date to convert, one of: a number (number of milliseconds since the epoch), an ISO Date string, a JavaScript Date or a ComputedObject.

tz **String, optional**
An optional timezone only to be used with a string date.

Example

```
print(ee.Date(0));              // Date (1970-01-01 00:00:00)
print(ee.Date(60000));          // Date (1970-01-01 00:01:00) - 1 hour past the epoch

// Multiply seconds by 1000 to get milliseconds
print(ee.Date(1498263286 * 1000));    // Date (2017-06-24 00:14:46

// Convert JavaScript now to Earth Engine Date
print(ee.Date(Date.now()));
```

ee.Date.advance
Create a new Date by adding the specified units to the given Date.
Date.advance(delta, unit, timeZone)

ee.Date.difference
Returns the difference between two Dates in the specified units; the result is floating-point and based on the average length of the unit.
Date.difference(start, unit)

ee.Date.fromYMD
Returns a Date given year, month, day.
ee.Date.fromYMD(year, month, day, timeZone)

ee.Date.get
Returns the specified unit of this date.
Date.get(unit, timeZone)

ee.Date.getFraction
Returns this date's elapsed fraction of the specified unit (between 0 and 1).
Date.getFraction(unit, timeZone)

ee.Date.millis
The number of milliseconds since 1970-01-01T00:00:00Z
Date.millis(

ee.Date.parse
Parse a date string, given a string describing its format.
e.Date.parse(format, date, timeZone)

ee.Date.unitRatio
Returns the ratio of the length of one unit to the length of another, e.g. unitRatio('day', 'minute') returns 1440. Valid units are 'year', 'month' 'week', 'day', 'hour', 'minute', and 'second'.
ee.Date.unitRatio(numerator, denominator)

ee.Date.update
Create a new Date by setting one or more of the units of the given Date to a new value. If a timeZone is given the new value(s) is interpreted in that zone.
Date.update(year, month, day, hour, minute, second, timeZone)

ee.PixelType
Returns a PixelType of the given precision with the given limits per element, and an optional dimensionality.
ee.PixelType(precision, minValue, maxValue, dimensions)

ee.PixelType.dimensions
Returns the number of dimensions for this type. Will be 0 for scalar values and >= 1 for array values.
PixelType.dimensions()

ee.PixelType.double
Returns the 64-bit floating point pixel type.
ee.PixelType.double()

ee.PixelType.float
Returns the 32-bit floating point pixel type.
ee.PixelType.float()

ee.PixelType.int8
Returns the 8-bit signed integer pixel type.
ee.PixelType.int8()

ee.PixelType.int16
Returns the 16-bit signed integer pixel type.
ee.PixelType.int16()

ee.PixelType.int32
Returns the 32-bit signed integer pixel type.
ee.PixelType.int32()

ee.PixelType.int64
Returns the 64-bit signed integer pixel type.
ee.PixelType.int64()

ee.PixelType.maxValue
Returns the maximum value of the PixelType.
PixelType.maxValue()

ee.PixelType.maxValue
Returns the maximum value of the PixelType.
PixelType.maxValue()

ee.PixelType.precision
Returns the precision of the PixelType. One of 'int', 'float', or 'double'.
PixelType.precision()

ee.PixelType.uint8
Returns the 8-bit unsigned integer pixel type.
ee.PixelType.uint8()

ee.PixelType.uint16
Returns the 16-bit unsigned integer pixel type.
ee.PixelType.uint16()

ee.PixelType.uint32
Returns the 32-bit unsigned integer pixel type.
ee.PixelType.uint32()

ee.Array

ee.Array(values, pixelType) >>>>> Returns an array with the given coordinates.

values Object
An existing array to cast, or a number/list of numbers/nested list of numbers of any depth to create an array from. For nested lists, all inner arrays at the same dep

pixelType PixelType, default: null
The type of each number in the values argument. If the pixel type is not provided, it will be inferred from the numbers in 'values'. If there aren't any numbers in 'values', this type must be provided.

Thank You