

Introduction to Python Objects and Expressions

Lecture 1: Introduction to Python

Daniel Kadyrov

What is Python?

- Python is a high-level, interpreted, general-purpose programming language.
- Python supports multiple programming paradigms, such as object-oriented, imperative, functional, and procedural.
- Python has a large and comprehensive standard library that provides built-in modules for various tasks, such as data structures, file handling, networking, etc.

Where is Python used?

Python is used in a wide variety of fields, including:

- Web development (server-side) with frameworks such as Django and Flask
- Software development (build control, testing, etc.)
- Mathematics and science
- System scripting (automation, etc.)
- Internet of Things (Raspberry Pi, MicroPython, etc.)
- **Data science (machine learning, data analysis, data visualization)**

Running Python

- Python is an interpreted language, which means that Python code is executed line by line.
- Python programs, also called scripts, are plain text files with the .py extension. You can run the programs through the following methods:
 - Running Python scripts from the command line or terminal by typing `python script.py` where script.py is the name of the script.
 - Within VSCode with the jupyter notebook extension or the debug tool

Hello World!

The first code you will run in almost every programming course is Hello World. This is a simple program that prints Hello World! to the screen. In Python, this can be done with a single line of code:

```
1 print("Hello World!")
```

- The `print()` function prints the specified message to the screen.
- The message can be a string, or any other object, the `print()` function will try to convert it to a string.

- In Python, everything is an object.
- An object is a collection of data and methods that operate on that data.
- An object has a type that determines what kind of data it can store and what methods it can use (string, integer, float, etc.)
- For example, a string object can store a sequence of characters and has methods for manipulating strings, such as `upper()`, `lower()`, `replace()`, etc.

- An expression is a piece of code that evaluates to a value.
- An expression can consist of literals, variables, operators, function calls, etc.
- For example, $2 + 3$ is an expression that evaluates to 5.
- Expressions can be used to assign values to variables, pass arguments to functions, return values from functions, etc.

Python Comments

- Comments are used to explain Python code.
- Comments are ignored by the Python interpreter.
- Comments can be used to prevent execution when testing code.
- Comments start with a `#` and end at the end of the line.
- Comments can be placed on a line by themselves, or at the end of a line of code.

```
1  # This is a comment  
2  print("Hello World!") # This is also a comment
```


Python Variables

- Variables are used to store data in memory.
- Variables are created when they are assigned a value.
- Variables can be assigned a new value at any time.
- Variables are assigned using the assignment operator `=`.
- Variable names can contain letters, numbers, and underscores, but cannot start with a number.
- [PEP8](#) style guide recommends using lowercase letters and underscores for variable names.

```
1  x = 5
2  y = 10
3  z = x + y
4  print(z)
5  z = y - x
6  print(z)
```

```
15
5
```

Python Data Types

Python has several built-in data types, including:

- **Numeric types:** int, float, complex
- **Boolean type:** bool
- **Sequence types:** list, tuple, range
- **Mapping type:** dict
- **Set types:** set, frozenset
- **String type:** str

The variable type can be checked with the `type()` function.

Data Types

Numeric

- Python has three numeric types: `int`, `float`, and `complex`.
- Integers are whole numbers, positive or negative, without decimals, of unlimited length.
- Floats are numbers with a decimal point and can be used to represent real numbers.
- Complex numbers are written with a "j" as the imaginary part.

```
1  x = 1      # int
2  y = 2.8    # float
3  z = 1j     # complex
```

Mathematical Expressions

Python supports the following mathematical operators:

```
1  x = 5
2  y = 2
3  print(x + y) # Addition
4  print(x - y) # Subtraction
5  print(x * y) # Multiplication
6  print(x / y) # Division
7  print(x % y) # Modulus
8  print(x ** y) # Exponentiation
9  print(x // y) # Floor division
```

```
7
3
10
2.5
1
25
2
```

Data Types

Boolean

- Boolean values are the two constant objects False and True.
- Boolean values are used to evaluate conditions.
- The comparison operators `==`, `!=`, `>`, `<`, `>=`, `<=` return boolean values.
- The boolean operators `and`, `or`, and `not` are used to combine boolean values.

```
1  x = True
2  y = False
3  print(x and y)
4  print(x or y)
5  print(not x)
```

```
False
True
False
```

Data Types

Sequence

- Python has three sequence types: list, tuple, and range.
- Python is a zero-indexed language, meaning the first item in a sequence is at index 0.
- Lists are ordered and changeable sequences of items. They are the most commonly used sequence type.
- Lists have several methods for manipulating them including:
 - `append()` to add an item to the end of the list.
 - `insert()` to insert an item at a specified index.
 - `remove()` to remove an item from the list.
 - `pop()` to remove an item at a specified index.
- Lists can also be indexed and sliced like strings through the use of square brackets `[]`
- There are many more methods available for lists available in the Python documentation at <https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>

Data Types

Sequence Examples

The following code demonstrates some of the methods available for lists:

```
1      x = [1, 2, 3, 4, 5]
2      print(x)
3      x.append(6)
4      print(x)
5      print(x[0])
6      print(x[1])
7      print(x[-1])
8      print(x[-2])
9      x[0] = 0
10     print(x)
11     print(len(x))
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6]
1
2
6
5
[0, 2, 3, 4, 5, 6]
6
```

Data Types

Dictionaries

- Dictionaries are unordered, changeable, and indexed collections of key-value pairs.
- Dictionaries are indexed by keys, which can be any immutable type.
- Dictionaries are created using curly brackets `{}` and key-value pairs separated by commas.
- Dictionaries have several methods for manipulating them including:
 - `get()` to get the value of a specified key.
 - `pop()` to remove an item with a specified key.
 - `keys()` to get a list of all the keys in the dictionary.
 - `values()` to get a list of all the values in the dictionary.
- There are many more methods available for dictionaries available in the Python documentation at <https://docs.python.org/3/library/stdtypes.html#mapping-types-dict>

Data Types

Dictionaries Examples

The following code demonstrates some of the methods available for dictionaries:

```
1  x = {  
2      "name": "John",  
3      "age": 36,  
4      "country": "  
        Norway"  
5  }  
6  print(x)  
7  print(x["name"])  
8  print(x.get("age"))  
9  x["age"] = 37  
10 print(x)  
11 print(x.keys())  
12 print(x.values())
```

```
{'name': 'John', 'age': 36, 'country':  
    'Norway'}  
John  
36  
{'name': 'John', 'age': 37, 'country':  
    'Norway'}  
dict_keys(['name', 'age', 'country'])  
dict_values(['John', 37, 'Norway'])
```

- Strings in Python are sequences of characters enclosed in single or double quotes.
- A multitude of methods are available for manipulating strings including:
 - `upper()` and `lower()` to convert the string to uppercase or lowercase.
 - `replace()` to replace a substring with another substring.
 - `split()` to split the string into a list of substrings.
 - `join()` to join a list of strings into one string.
 - Strings can also be indexed and sliced like lists through the use of square brackets `[]`
- There are many more methods available for strings available in the Python documentation at <https://docs.python.org/3/library/stdtypes.html#string-methods>

Data Types

String: Examples

The following code demonstrates some of the methods available for strings:

```
1  s = "Hello World!"
2  print(s)
3  print(s.upper())
4  print(s.lower())
5  print(s.replace("World", "Python"))
6  print(s.split(" "))
7  print(" ".join(["Hello", "World!"]))
8  print(s[0])
9  print(s[0:5])
```

```
Hello World!
HELLO WORLD!
hello world!
Hello Python!
['Hello', 'World!']
Hello World!
H
Hello
```