



DSHL Summer School 2022

Day 1: Python Intro

Martin Antenreiter

Python

int & float

- `# this is the a comment`
- `1 + 3 # int values`
- The operators `+`, `-`, `*` and `/` work just like in most other languages
- `8.0 # floating point number`
- `8 / 5 # division always returns a floating point number`
- `3 ** 2 # 3 to the power of 2`

Python

Boolean Type: bool

- You can evaluate any expression in Python, and get one of two answers, True or False.

`1 < 2`

`bool(12)`

`1 == 2`

`bool(0)`

`1 != 2`

`bool(3.14)`

`1 >= 2`

`bool([])`

Assignment

- The equal sign (=) is used to assign a value to a variable.
- `a = 3 ** 2 # assign int result to variable a`
- `b = 3.0 ** 2 # assign float result`
- Compare output of `type(a)` and `type(b)`!
- `type(...)` is a function call

Python Strings

- `name = 'Martin'` `# single quotes`
- `name = "Martin"` `# double quotes`
- `name = "Martin's notebook"`
- `astr = str()`
- Escape character `\`
`\n` (newline), `\t` (tab), `\a` (bell)
`\\`, `\'`, `\"`

Python

Complex Numbers

- $x = 3 + 5j$
- $y = 5j$
- $z = x + y$
- `type(x)` # gives `<class 'complex'>`
- `dir(x)` # lists all properties and methods of the object `x`
- `help(x)` # help function

Build-in Function `print(x)`

```
print(x)
```

```
print(x, y)
```

```
print(f'result is {x}!')
```

```
print(f'result is {x:.2f}!')
```

```
print(f'result is {a:04d}!') # a = 12
```

Print objects to the default text stream.

Python if

- Syntax

```
if test-expression:
    statement(s)
```

```
a, b = 33, 200
```

```
if b > a:
```

```
    print("b is greater than a")
```

- Python relies on indentation to define scope in the code.

Python if - else

- Syntax

if test-expression:

 statement(s) for True

else:

 statement(s) for False

```
a = 33
```

```
if a >= 0:
```

```
    print("Positive or Zero")
```

```
else:
```

```
    print("Negative number")
```

Python if – elif - else

```
a = 33
```

```
if a == 0:
```

```
    print("Zero number")
```

```
elif a > 0:
```

```
    print("Positive number")
```

```
else:
```

```
    print("Negative number")
```

Python Collections

- There are four collection data types
 - List = [1, 2, 3]
 - List is a collection which is ordered and changeable. Allows duplicate members.
 - Tuple = (1, 2, 3)
 - Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
 - Set = {1, 2, 3}
 - Set is a collection which is unordered and unindexed. No duplicate members.
 - Dictionary = { 'a' : 1, 'b' : 2, 'c' : 3 }
 - Dictionary is a collection which is ordered and changeable. No duplicate members.

Python Lists

- `a = list()` # empty list
- `a = []`
- `a = list(range(10))`

- `a[0]` # first element
- `a[-1]` # last element
- `a[2:5]` # third, fourth, and fifth element
- `a[::-1]` # reverse list
- `help(list)` # append, insert, remove,
pop, sort, reverse, ...

Python Loops

- Python has two loop commands
 - while loops
 - for loops

```
i = 1
```

```
while i < 6:
```

```
    print(i)
```

```
    i = i + 1
```

```
arr = [1, 2, 3]
```

```
for x in arr:
```

```
    print(x)
```

Build-in Function len(x)

len(x)

Return the length (the number of items) of an object. The argument may be a sequence (such as a string, bytes, tuple, list, or range) or a collection (such as a dictionary, set, or frozen set).

Build-in Function `abs(x)`

`abs(x)`

Return the absolute value of a number. The argument may be an integer, a floating point number, or an object implementing `__abs__()`. If the argument is a complex number, its magnitude is returned.

Build-in Functions

`max(...)`, `min(...)`

`max(x, y)` or `max(aList)`

Return the largest item in an iterable or the largest of two or more arguments.

`min(x, y)` or `min(aList)`

Return the smallest item in an iterable or the smallest of two or more arguments.

Python Functions

- A function is a block of code which only runs when it is called.

```
def my_func(name, country = "Norway"):  
    print(f"{name} is from {country}".)
```

```
my_function("Mark", "Sweden")
```

```
my_function("Bob", "India")
```

```
my_function("Peter")
```

Python Functions - Return

- The Python return statement is a statement that you can use inside a function or method to send the function's result back to the caller.

```
def my_function(x):
    return 3 * x
```

```
print(my_function(1))
print(my_function(3))
```

Python Classes

- Python is an object oriented programming language.

```
class Person:
```

```
    def __init__(self, name, age = 0):      # constructor
        self.name = name                   # define attributes
        self.age = age
```

```
    def hello_func(self):                  # methode
        print(f"Hello my name is {self.name}")
```

```
p1 = Person("John", 20)                  # calls __init__(...)
p1.hello_func()
```

Modules

- Python has a library of standard modules
- Add modules with pip
 - pip is the package installer for Python

```
import math
```

```
import random
```

see

<https://docs.python.org/3.9/library/index.html>

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
import tensorflow as tf
```