

---

# Introduction to Python

Artificial Intelligence TN I (YS 02) - Fall 2025

---

---

# Course links

- Course Material: Course [page](#)
  - Project Submission: Eclass [page](#)
  - Announcements, questions, etc: Piazza [page](#)
-

---

---

# Course grading

- 5 homework projects (90%)
  - Final exam (10%)
-

---

# Download Python

## Download:

- <https://www.python.org/downloads/>
- or
- <https://www.anaconda.com/products/individual>

Version: Python 3.6.x

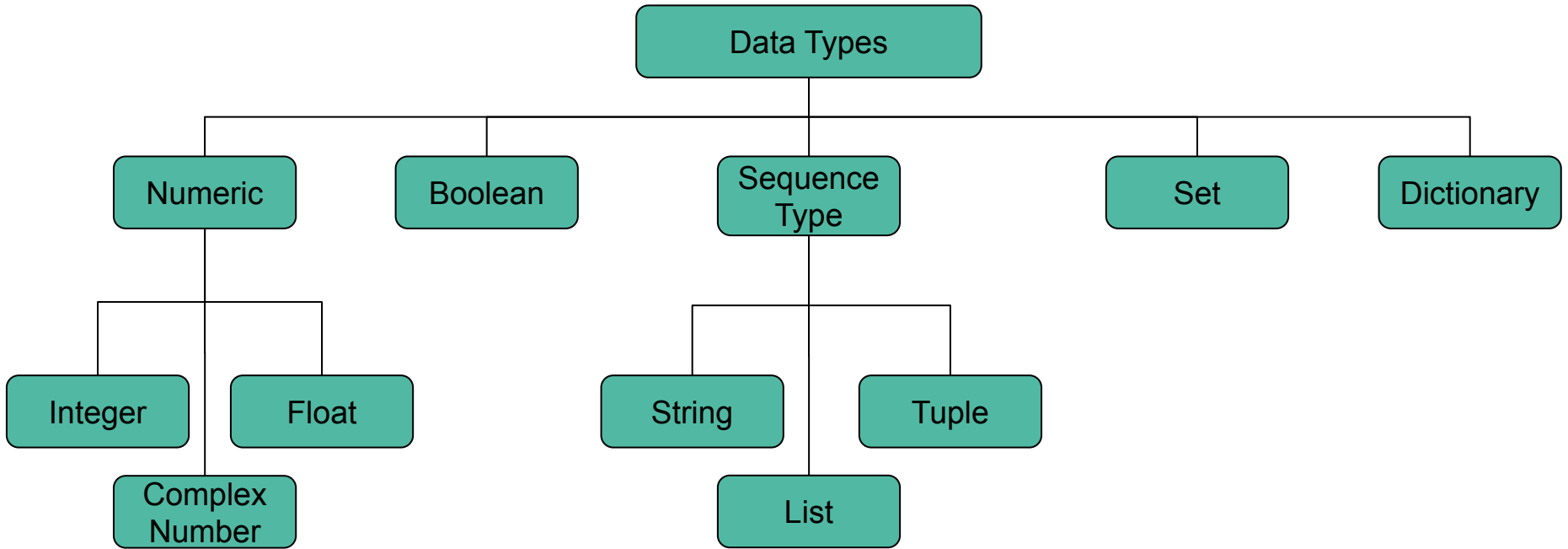
## Editors:

- 1) GEdit (text)
- 2) Notepad++ (text)
- 3) Idle (text)
- 4) Spyder (IDE)
- 5) VSCode (IDE)
- 6) Atom (IDE)
- 7) Python shell
- 8) PyCharm

# My first program

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

# What are my Data Types?



# Data Types - Known

## Integer

```
>>> x = 3
>>> y = 5
>>> z = x+y
>>> z * z
64
```

## Float

```
>>> x = 3.4
>>> y = 5.6
>>> z = x+y
>>> z * z
81.0
```

## Boolean

```
>>> x = True
>>> y = False
>>> z = (x and y) or x
>>> z
True
```

# Data Types - Strings

Ordered Sequence of characters (Immutable)

```
>>> x = "Hello "  
>>> y = "Everyone"  
>>> z = x+y  
>>> z  
Hello Everyone  
>>> x[4] + " " + y[0]  
o E  
>>> x[-1] + y[2:] + y[4:6]  
eryoneyo
```

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| H | e | l | l | o |   |
| 0 | 1 | 2 | 3 | 4 | 5 |

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| E | v | e | r | y | o | n | e |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

```
>>> print("Hello ", "World!")  
Hello World!  
>>> s = "World!"  
>>> print(f"Hello {s}")  
Hello World!  
>>> print("Hello {}".format(s))  
Hello World!  
>>> print("Hello %s"% s)  
Hello World!
```



# Data Types - List

Ordered Sequence of items (Mutable) - Indexing same as strings

```
>>> l1 = [0,1,2,3,4,5]
>>> l2 = ['a', '2', 'b', True, 2.4, [2]]
>>> len(l1)          #Number of items in l1
6
>>> l2.index('b')     #Find index of 'b'
2
>>> min(l1)          #Minimum item of l1
0
>>> sum(l1)          #Sum of elements of l1
15
>>> list(range(2,10)) #Generate list of
sequential items
[2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> l1 = [0,1,2,3,4,5]
>>> l1.append(-1)     #Add 1 item at end of list
>>> l1
[0, 1, 2, 3, 4, 5, -1]
>>> l1 += [-2, 15, 3, 27] #Concatenate lists
[0, 1, 2, 3, 4, 5, -1, -2, 15, 3, 27]
>>> l1.sort()         #Sort list inplace
>>> l1
[-2, -1, 0, 1, 2, 3, 3, 4, 5, 15, 27]
>>> l1.reverse()      #Sort list inplace different order
>>> l1
[27, 15, 5, 4, 3, 3, 2, 1, 0, -1, -2]
>>> l1.pop()          #Remove last item
-2
```

# Data Types - Tuples

Ordered Sequence of items (Immutable) - Indexing same as before

```
>>> l1 = ('a', '2', 'b', True, 2.4, [2])
>>> len(l1)
6
>>> l1[0]
'a'
>>> l1[0] = 0      #Error
```

# Data Types - Set

Unordered collection of items (Mutable) - No Indexing, unique items

```
>>> s1 = {'a', 'b', 'b'}
>>> s1
{'a', 'b'}
>>> s1.update(['c', 'd', 'a', 'b'])
>>> s1
{'a', 'b', 'c', 'd'}
```

# Data Types - Dictionary

Map of items (Mutable) - Unique Keys, possibly duplicate Values

```
>>> d1 = {'a': 1, 'b': 10, 'c': 1}
>>> d1['a']
1
>>> del d1['b']
>>> d1
{'a': 1, 'c': 1}
>>> d1['a'] = 2
>>> d1
{'a': 2, 'c': 1}
```

```
>>> d1 = {'a': 1, 'b': 10, 'c': 1}
>>> d1.keys()
dict_keys(['a', 'b', 'c'])
>>> d1.values()
dict_values([1, 10, 1])
>>> d1.items()
dict_items([('a', 1), ('b', 10), ('c', 1)])
```

# Functions

```
def square(x):  
    return x*x
```

```
>>> square(9)
```

```
81
```

```
>>> square(15)
```

```
225
```

**Note:** No brackets, we use indentation (tabs or spaces)!

# Control Flows - If

```
def foo(x):  
    if x % 2 == 0:  
        print(f'{x} is even!')  
    elif x % 2 != 0:  
        print(f'{x} is odd!')  
    else:  
        print(Did you typed a  
number?)
```

```
>>> foo(15)  
15 is odd!  
>>> foo(228)  
228 is even!
```

```
invited = ["Alice", "Bob", "Carol"]  
guest = 'David'  
if guest in invited:  
    print(f"Welcome, {guest}")  
else:  
    print(f"You are not invited, {guest}")
```

You are not invited, David

in works for string, list, tuple, set and dict (keys)

# Control Flows - Iterations

## For Loop

```
invited = ["Alice", "Bob", "Carol"]

def is_invited(guest):
    if guest in invited:
        print(f"Welcome, {guest}")
    else:
        print(f"You are not invited, {guest}")

guests = ["Alice", "David"]
for guest in guests:
    is_invited(guest)

Welcome, Alice
You are not invited, David
```

## While Loop

```
invited = ["Alice", "Bob", "Carol"]

def is_invited(guest):
    if guest in invited:
        print(f"Welcome, {guest}")
    else:
        print(f"You are not invited, {guest}")

guests = ["Alice", "David"]
while len(guests) != 0:
    guest = guests.pop()
    is_invited(guest)

You are not invited, David
Welcome, Alice
```

**Note:** No do - while in Python!

# Control Flows - List Comprehension

```
>>> l = list(range(10))
>>> l
[0, 1, 2, 4, 5, 6, 7, 8, 9]
>>> squares = [x * x for x in l]
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
def factorial(x):
    fact = 1
    if x < 0:
        fact = -1
    elif x == 0:
        pass
    else:
        for i in range(1, x + 1):
            fact = fact * i
    return fact

>>> l = [1, 5, 10]
>>> facts = [factorial(x) for x in l]
>>> facts
[1, 120, 3628800]
```



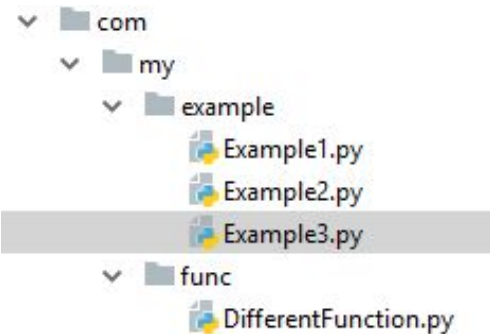
# Useful Functionalities & Libraries

Try - Catch - Finally

```
try:  
    main()  
  
catch Exception as e:  
    print(e)  
  
finally:  
    print("Program finished")
```

DOCSTRINGS  
Google docstrings

```
Import numpy as np  
Import pandas as pd  
From src.py_functions import * -> (wildcard)
```



# Useful Functionalities & Libraries (2)

## DOCSTRINGS

Google docstrings for Python style

```
"""_Summary_

    _Args_:
        param1 (int): The first parameter.
        param2 (str): The second parameter.

    _Returns_:
        bool: The return value. True for success, False
        otherwise.
"""
```

## Example

```
def hello_user(name:str) -> str:
    """This function will salut a user.

    Args:
        name (str): A string with a name.

    Returns:
        str: The salutation to a user name.
    """
    return print("Hello",name)
```

# Classes

```
class Person:
    def __init__(self, name, inv):
        self.name = name
        self.inv = inv

    def is_invited(self):
        if self.inv:
            msg = f"Welcome, {self.name}"
        else:
            msg = f"You are not invited, {self.name}"
        return msg
```

```
persons = {}
guests = ["Alice", "Bob", "Carol", "David" ]
invites = [1] * 3 + [0]
for no, (guest, inv) in enumerate(zip(guests, invites)):
    persons[no] = Person(guest, inv)

for no, person in persons.items():
    print(f'Guest {no}: {person.is_invited()}')
```

Guest 0: Welcome, Alice

Guest 1: Welcome, Bob

Guest 2: Welcome, Carol

Guest 3: You are not invited, David

# Call python script from Shell

```
If __name__ == "__main__":  
    my_main()
```

# Python Environments

Create an isolated environment containing a specific python version and the libraries needed for a project.

- venv Built in with python
- conda Install with [anaconda](#)  
Install with [miniconda](#)  
Install with [miniforge](#)
- mamba Install with [miniforge](#)

# Python Environments (instructions)

## venv

```
$ cd /home/user/python_project  
$ python -m venv test_env  
$ source test_env/bin/activate  
$ pip install numpy  
$ deactivate  
$ rm -rf test_env
```

## conda

```
$ conda create --name test_env python=3.12  
$ conda activate test_env  
$ conda install numpy  
$ conda deactivate  
$ conda remove --name test_env --all
```

## mamba

```
$ mamba create --name test_env python=3.12  
...  
same as conda
```

# Useful links

- <https://medium.com/fintechexplained/everything-about-python-from-beginner-to-advance-level-227d52ef32d2>
- <https://towardsdatascience.com/tagged/python>
- <https://scikit-learn.org/stable/> -> Data Science, Machine Learning
- **Stackoverflow**