Introduction to Python3 for Scientists and Engineers

Muhammad Izham

Universiti Malaysia Perlis

izham@unimap.edu.my
sugita5019@gmail.com
https://github.com/izham-sugita

Overview

Introduction

- Created by: Guido van Rossum, 1989-1991
- Why: The creator wanted something easy to use.
- Is it really that easy? Yes (and no)
- Very readable with little memory managment.
- Lots of proven libraries. Open source.
- `https://docs.python.org/3/tutorial/index.html`
- `https://www.tutorialsteacher.com/python`
- `https://www.w3schools.com/python/default.asp`

Introduction

- Download from `https://www.python.org/`
- For Ubuntu download from repository:

```
user@pc-name:~/apt install python3
```

- For Ubuntu installing packages:

```
user@pc-name:~/pip3 install numpy
```

- For Windows installing packages if $C:\backslash Python\backslash Scripts\backslash$ is in the path:

```
C:\User\dummy\> pip3 install numpy
```

- For Windows installing packages:

```
C:\User\dummy\> python3 -m pip3 install numpy
```

The Zen of Python

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one – and preferably only one – obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!

The "Hello, World!"

Greetings!!

```
1 print("Hello,Python!")
```

file: helloPython.py

Variables in Python3

Importing a module:

```
1 import numpy as np
```

Print to screen:

```
1 a = 5
2 print("a=%f"%a)
3 a = "Any character"
4 print(a)
```

Taking an input:

```
1 x=int(input("Enter an integer: "))
2 print(x)
```

file: variables.py

Variables in Python3

How to use module:

```
1 x = np.float32(input("Enter a single precision number ")
    )
2 print(x)
3
4 print("\nWill print numbers with different precision\n")
    )
5
6 x = np.float64(np.random.random())
7 print(x)
8 x = np.float32(np.random.random())
9 print(x)
```

file: variables.py

Container: List, Dictionaries, Set, Tupples

Container: List

- A list can contains any type of variable
- Unlike the normal practice of array where an array contains just one type of variable

```
1 xs = [3, 1, 2]    # Create a list
2 print(xs, xs[2])  # Prints "[3, 1, 2] 2"
3 print(xs[-1])     # Negative indices count from the end
      of the list; prints "2"
4 xs[2] = 'foo'     # Lists can contain elements of
     different types
5 print(xs)         # Prints "[3, 1, 'foo']"
```

file: python-container.py

Class in Python3

- Variables in class are public by default.
-

Defining a class:

```
1  #define a class
2  class vehicle:
3      name = ""
4      kind = "car"
5      color = ""
6      value = 100.0
7      def description(self):
8          desc_str = \
9          "My %s is a %s %s worth $%.2f." \
10         %(self.name, self.color, self.kind, self.value)
```

file: py-class0.py

Template slide

Template slide

The End
Questions?