COMP702: Classical Artificial Intelligence

12: Introduction to Python

FALMOUTH
UNIVERSITY

Installing Python

Getting Python for Windows

Getting Python for Windows

- Go to `https://www.python.org/`

Getting Python for Windows

- Go to `https://www.python.org/`
- Download the Windows x64 installer

Getting Python for Windows

- Go to `https://www.python.org/`
- Download the Windows x64 installer
- Make sure to enable “Add Python to PATH” when installing

Getting Python for other OSes

Getting Python for other OSes

Mac OSX:

Getting Python for other OSes

Mac OSX:

- Comes with an outdated version of Python

Getting Python for other OSes

Mac OSX:

- Comes with an outdated version of Python
- Download the latest from `https://www.python.org/`, or install using Homebrew

Getting Python for other OSes

Mac OSX:

- Comes with an outdated version of Python
- Download the latest from `https://www.python.org/`, or install using Homebrew

Linux:

Getting Python for other OSes

Mac OSX:

- Comes with an outdated version of Python
- Download the latest from `https://www.python.org/`, or install using Homebrew

Linux:

- May come preinstalled

Getting Python for other OSes

Mac OSX:

- Comes with an outdated version of Python
- Download the latest from `https://www.python.org/`, or install using Homebrew

Linux:

- May come preinstalled
- If not, check the package manager for your distribution

Choosing an IDE

Choosing an IDE

- **IDLE**

Choosing an IDE

- **IDLE**
  - Comes with Python

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension
  - Useful features: syntax higlighting, autocomplete, linting, basic debugging

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension
  - Useful features: syntax higlighting, autocomplete, linting, basic debugging
- **PyCharm**

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension
  - Useful features: syntax higlighting, autocomplete, linting, basic debugging
- **PyCharm**
  - `https://www.jetbrains.com/pycharm/`

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension
  - Useful features: syntax higlighting, autocomplete, linting, basic debugging
- **PyCharm**
  - `https://www.jetbrains.com/pycharm/`
  - Sign up for an educational account at `https://www.jetbrains.com/community/education/` to get the Professional version for free

Choosing an IDE

- **IDLE**
  - Comes with Python
  - Basic functionality
- **Visual Studio Code**
  - Install the Python extension
  - Useful features: syntax higlighting, autocomplete, linting, basic debugging
- **PyCharm**
  - `https://www.jetbrains.com/pycharm/`
  - Sign up for an educational account at `https://www.jetbrains.com/community/education/` to get the Professional version for free
  - Fully featured: advanced debugging, package management

Three ways to run Python code

Three ways to run Python code

- Write code in a `.py` file and run it

Three ways to run Python code

- Write code in a `.py` file and run it
- Type code into the interactive interpreter

Three ways to run Python code

- Write code in a `.py` file and run it
- Type code into the interactive interpreter
- Use a more advanced interactive environment e.g. Jupyter Notebook

Package management

Package management

- Python has many useful **packages** available

Package management

- Python has many useful **packages** available
- Can be installed using **pip**

Package management

- Python has many useful **packages** available
- Can be installed using **pip**
- From the command line: `pip install ...`

Package management

- Python has many useful **packages** available
- Can be installed using **pip**
- From the command line: `pip install ...`
- In PyCharm: File → Settings → Project → Python Interpreter

FALMOUTH
UNIVERSITY

Python for C# programmers

Hello World

```
print("Hello, world!")
```

Hello World

```
print("Hello, world!")
```

- Code does not have to be inside a class or function

Hello World

```
print("Hello, world!")
```

- Code does not have to be inside a class or function
- No semicolons

Hello World

```
print("Hello, world!")
```

- Code does not have to be inside a class or function
- No semicolons
- `print` is a built-in function

Comments

```
# This is a comment
```

Variables

```
a = 1
b = 2
c = a + b
```

Variables

```
a = 1
b = 2
c = a + b
```

- Variables do not need to be declared

Variables

```
x = 7
x = "Hello"
```

Variables

```
x = 7
x = "Hello"
```

- Variables can hold values of any type

If statement

```
if x < 10:
    print("asdf")
elif x < 20:
    print("qwerty")
else:
    print("zxcv")
```

If statement

```
if x < 10:
    print("asdf")
elif x < 20:
    print("qwerty")
else:
    print("zxcv")
```

- Indentation matters

If statement

```
if x < 10:
    print("asdf")
elif x < 20:
    print("qwerty")
else:
    print("zxcv")
```

- Indentation matters
- Note the colons and the lack of parentheses

Lists

```
my_list = ["Hello", "World", "Foo", 42]
print(my_list[0])
```

Lists

```
my_list = ["Hello", "World", "Foo", 42]
print(my_list[0])
```

- Can store values of any type

For loop

```
for x in my_list:
    print(x)
```

For loop

```
for x in my_list:
    print(x)
```

- Works like `foreach` in C#

For loop

```
for i in range(10):
    print(i)
```

For loop

```
for i in range(10):
    print(i)
```

- Python doesn't have C-style `for` loops

For loop

```
for i in range(10):
    print(i)
```

- Python doesn't have C-style `for` loops
- Built-in function `range(n)` gives numbers from 0 to $n - 1$

Functions

```
def add(a, b):
    return a + b
```

Functions

```
def add(a, b):
    return a + b
```

- Can return any value, or nothing

Functions are values

```
def add(a, b):
    return a + b

x = add

print(x(3, 4))
```

Classes

```
class Thing:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

x = Thing(2, 3)
```

Classes

```
class Thing:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

x = Thing(2, 3)
```

- `__init__` is the constructor

Classes

```
class Thing:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

x = Thing(2, 3)
```

- `__init__` is the constructor
- `self` is equivalent to `this`

Classes

```
class Thing:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def add(self):
        return self.a + self.b

x = Thing(2, 3)
```

- `__init__` is the constructor
- `self` is equivalent to `this`
- `self` is never implicit, unlike `this`

List comprehensions

```
my_list = [1, 3, 6, 10]
my_other_list = [x*2 for x in my_list if x < 10]
```

List comprehensions

```
my_list = [1, 3, 6, 10]
my_other_list = [x*2 for x in my_list if x < 10]
```

- Similar to LINQ queries in C#

Python and C

Python and C

- Python has many advantages, but speed is not one of them...

Python and C

- Python has many advantages, but speed is not one of them...
- For intensive calculations we generally rely on external libraries written in C/C++

Python and C

- Python has many advantages, but speed is not one of them...
- For intensive calculations we generally rely on external libraries written in C/C++
- It is also possible to embed the Python interpreter in a C/C++ program

Useful libraries

Useful libraries

- **NumPy**: fast calculation with *N*-dimensional numerical arrays

Useful libraries

- **NumPy**: fast calculation with *N*-dimensional numerical arrays
- **SciPy**: various scientific tools, including statistical analysis

Useful libraries

- **NumPy**: fast calculation with *N*-dimensional numerical arrays
- **SciPy**: various scientific tools, including statistical analysis
- **Pandas**: importing and manipulation of large datasets

Useful libraries

- **NumPy**: fast calculation with *N*-dimensional numerical arrays
- **SciPy**: various scientific tools, including statistical analysis
- **Pandas**: importing and manipulation of large datasets
- **Matplotlib**: plotting of charts and graphs

Useful libraries

- **NumPy**: fast calculation with *N*-dimensional numerical arrays
- **SciPy**: various scientific tools, including statistical analysis
- **Pandas**: importing and manipulation of large datasets
- **Matplotlib**: plotting of charts and graphs
- Various libraries for machine learning, which you will learn about in COMP704...

FALMOUTH
UNIVERSITY

Pygame

Pygame

Pygame

- 2D game development library

Pygame

- 2D game development library
- Based on SDL (Simple DirectMedia Layer)

Pygame

- 2D game development library
- Based on SDL (Simple DirectMedia Layer)
- Allows low-level control over the game loop

The main game loop

The main game loop

Most main loops in games follow the same basic pattern:

The main game loop

Most main loops in games follow the same basic pattern:

repeat

The main game loop

Most main loops in games follow the same basic pattern:

repeat

    handle events

The main game loop

Most main loops in games follow the same basic pattern:

repeat

handle events

update game state

The main game loop

Most main loops in games follow the same basic pattern:

repeat

handle events
update game state
draw graphics

The main game loop

Most main loops in games follow the same basic pattern:

repeat

- handle events
- update game state
- draw graphics
- sleep to maintain frame rate

The main game loop

Most main loops in games follow the same basic pattern:

repeat

- handle events
- update game state
- draw graphics
- sleep to maintain frame rate
- swap buffers

The main game loop

Most main loops in games follow the same basic pattern:

repeat
- handle events
- update game state
- draw graphics
- sleep to maintain frame rate
- swap buffers

until "quit" event received

Double buffering

Double buffering

- Pygame (and other frameworks) use **two** graphics buffers

Double buffering

- Pygame (and other frameworks) use **two** graphics buffers
- One is displayed on the screen

Double buffering

- Pygame (and other frameworks) use **two** graphics buffers
- One is displayed on the screen
- The other is used to draw the next frame

Double buffering

- Pygame (and other frameworks) use **two** graphics buffers
- One is displayed on the screen
- The other is used to draw the next frame
- When drawing is finished, they are **swapped** — the drawn buffer is displayed, and the old buffer is used to draw the next frame

FALMOUTH
UNIVERSITY

Workshop

Workshop

- Begin working through the Python resources linked on LearningSpace
- Suggested Christmas project: use Pygame to reimplement your favourite 80s arcade game!