2G03: Introduction to Python

2019

Why Python?

Why Python?

- Easy to read and write code.

Why Python?

- Easy to read and write code.
- Low level tasks handled automatically (e.g. memory allocation, type casting)

Why Python?

- Easy to read and write code.
- Low level tasks handled automatically (e.g. memory allocation, type casting)
- Extensive libraries for diverse range of problems (NumPy, SciPy, NetKet, TensorFlow, Matplotlib)

Why Python?

- Easy to read and write code.
- Low level tasks handled automatically (e.g. memory allocation, type casting)
- Extensive libraries for diverse range of problems (NumPy, SciPy, NetKet, TensorFlow, Matplotlib)
- Becoming ubiquitous in scientific computing for data processing, plotting, analysis

Goals for Python Lectures

Goals for Python Lectures

1. Learn how python for loops, while loops, if statements etc. . . compare to C++

Goals for Python Lectures

1. Learn how python for loops, while loops, if statements etc. . . compare to C++
2. Run some simple programs in C++ and in Python

What's different?

What's different?

1. Interpreted vs compiled.

What's different?

1. Interpreted vs compiled.
2. Dynamically typed vs. statically typed.

What's different?

1. Interpreted vs compiled.
2. Dynamically typed vs. statically typed.
3. Whitespace senstive vs. braces.

Defining variables

Defining variables

```
int main() {
  int a=1;                                  // an integer
  float b=2.5;                              // a float
  char greeting[]="Hello World"; // a string
  return 0;
}
```

Defining variables

```c
int main() {
  int a=1;                                  // an integer
  float b=2.5;                              // a float
  char greeting[]="Hello World"; // a string
  return 0;
}
```

```
a=1                         # an integer
b=2.5                       # a float
greeting="Hello World" # a string
```

No need to declare types!

What about arrays?

What about arrays?

In C++ we must declare type and size:

```
int main() {
  int arr[5]={1, 2, 3, 10, 12};
  return 0;
}
```

What about arrays?

In C++ we must declare type and size:

```
int main() {
  int arr[5]={1, 2, 3, 10, 12};
  return 0;
}
```

In python we don't care:

```
arr=[1, 2, 3, 10, 12]
```

In python "arrays" are actually called "lists".

More on lists

More on lists

Let's say we want to print an array element in C++

```cpp
#include <iostream>
using namespace std;
int main() {
  int arr[5]={1, 2, 3, 10, 12};
  cout << arr[2] << endl; // print the number 3
  return 0;
}
```

In python:

```python
arr=[1, 2, 3, 10, 12]
print(arr[2])
```

Some sweet extras

Some sweet extras

Unlike C++ arrays, python lists can be appended to easily.

```
colours=["red", "blue", "green"]
colours.append("purple")
```

Some sweet extras

Unlike C++ arrays, python lists can be appended to easily.

```
colours=["red", "blue", "green"]
colours.append("purple")
```

And can be subdivided easily. For example:

```
lessColours=colours[1:]
# lessColours=["blue", "green", "purple"]
```

Some sweet extras

Unlike C++ arrays, python lists can be appended to easily.

```
colours=["red", "blue", "green"]
colours.append("purple")
```

And can be subdivided easily. For example:

```
lessColours=colours[1:]
# lessColours=["blue", "green", "purple"]
```

Plus, output is a breeze. This,

```
print(colours)
```

will print the whole array.

For loops

For loops

For loops are MUCH more flexible in Python but today we'll just look at the basics. Let's print all the numbers from 1 to 10.

For loops

For loops are MUCH more flexible in Python but today we'll just look at the basics. Let's print all the numbers from 1 to 10.

```
#include <iostream>
using namespace std;
int main() {
  for(int i=1; i<11; i++) {
    cout << i << endl;
  }
}
```

For loops

For loops are MUCH more flexible in Python but today we'll just look at the basics. Let's print all the numbers from 1 to 10.

```
#include <iostream>
using namespace std;
int main() {
  for(int i=1; i<11; i++) {
    cout << i << endl;
  }
}
```

In python we use the "range" function (NOTICE THE WHITESPACE)

```
for i in range(1, 11):
  print(i, '\n')
```

A taste of the good life

A taste of the good life

Python syntax can be very intuitive if the code is well written.

A taste of the good life

Python syntax can be very intuitive if the code is well written. For example, (NOTICE THE WHITESPACE)

```
colours=['red', 'blue', 'green', 'purple']
for colour in colours:
  print(colour, '\n')
```

If-Else statements

If-Else statements

In C++ an if-statement took the following form. Let's print the numbers from 1 to 10... again.

```
#include <iostream>
using namespace std;
int main() {
  int n=1;
  while(n<6) {
    cout << n << endl;
    n+=1;
  }
}
```

While Loops

Let's print the numbers from 1 to 10... again.

```
#include <iostream>
using namespace std;
int main() {
  int n=1;
  while(n<6) {
    cout << n << endl;
    n+=1;
  }
}
```

Now in python (NOTICE THE WHITESPACE)

```
n=1
while n < 6:
  print(n)
  n+=1
```

If-Else Statements

A simple if statement in c++

```
int main() {
  bool switch=True;
  if(switch) {switch=False;}
  else {switch=True;}
  return 0;
}
```

In python (NOTICE THE WHITESPACE):

```
switch=True
if switch:
  switch=False
else:
  switch=True
```