Python Overview

Chapter 1

`Start Python.pyw`
This is the icon you double-click on to start a Python Shell (IDLE).

`>>>`
The Python prompt. This is where you type in a Python command.

Note: All commands you type (including the Myro commands listed above) are essentially Python commands. Later, in this section we will list those commands that are a part of the Python language.

Chapter 2

```
def <FUNCTION NAME>(<PARAMETERS>):
    <SOMETHING>
    ...
    <SOMETHING>
```
Defines a new function named <`FUNCTION NAME`>. A function name should always begin with a letter and can be followed by any sequence of letters, numbers, or underscores (_), and not contain any spaces. Try to choose names that appropriately describe the function being defined.

Chapter 3

Values
Values in Python can be numbers (integers or floating point numbers) or strings. Each type of value can be used in an expression by itself or using a combination of operations defined for that type (for example, +, -, *, /, % for numbers). Strings are considered sequences of characters (or letters).

Names
A name in Python must begin with either an alphabetic letter (a-z or A-Z) or the underscore (i.e. _) and can be followed by any sequence of letters, digits, or underscore letters.

```
input(<prompt string>)
```

This function prints out `<prompt string>` in the IDLE window and waits for the user to enter a Python expression. The expression is evaluated and its result is returned as a value of the input function.

```
from myro import *
initialize("comX")

<any other imports>
<function definitions>
def main():
    <do something>
    <do something>
    ...

main()
```

This is the basic structure of a robot control program in Python. Without the first two lines, it is the basic structure of all Python programs.

```
print <expression1>, <expression2>, ...
```

Prints out the result of all the expressions on the screen (in the IDLE window). Zero or more expressions can be specified. When no expression is specified, it prints out an empty line.

`<variable name> = <expression>`
This is how Python assigns values to variables. The value generated by <expression> will become the new value of <variable name>.

`range(10)`
Generates a sequence, a list, of numbers from 0..9. There are other, more general, versions of this function. These are shown below.

`range(n1, n2)`
Generates a list of numbers starting from n1…(n2-1). For example, range(5, 10) will generate the list of numbers [5, 6, 7, 8, 9].

`range(n1, n2, step)`
Generates a list of numbers starting from n1…(n2-1) in steps of step. For example, range(5, 10, 2) will generate the list of numbers [5, 7, 9].

Repetition

```
for <variable> in <sequence>:
   <do something>
   <do something>
   ...

while timeRemaining(<seconds>):
    <do something>
    <do something>
    ...

while True:
   <do something>
   <do something>
   ...
```

These are different ways of doing repetition in Python. The first version will assign `<variable>` successive values in `<sequence>` and carry out the body once for each such value. The second version will carry out the body for `<seconds>` amount of time. `timeRemaining` is a Myro function (see above). The last version specifies an un-ending repetition.

Chapter 4

`True, False`
These are Boolean or logical values in Python. Python also defines `True` as 1 and `False` as 0 and they can be used interchangeably.

`<, <=, >, >=, ==, !=`
These are relational operations in Python. They can be used to compare values. See text for details on these operations.

`and, or not`
These are logical operations. They can be used to combine any expression that yields Boolean values.
`random()`
Returns a random number between 0.0 and 1.0. This function is a part of the `random` library in Python.

`randRange(A, B)`
Returns a random number in the range A (inclusive) and B (exclusive). This function is a part of the `random` library in Python.

Chapter 5

```
if <CONDITION>:
   <statement-1>
  ...
   <statement-N>
```
If the condition evaluates to True, all the statements are are performed. Otherwise, all the statements are skipped.

`return <expression>`
Can be used inside any function to return the result of the function.

`<string>.split()`
Splits `<string>` into a list.

`urlopen(<URL>)`
Establishes a stream connection with the `<URL>`. This function is to be imported from the Python module `urlopen`.

`<stream>.read()`
Reads the entire contents of the `<stream>` as a string.

Lists:
`[]` is an empty list.

`<list>[i]`
Returns the `i`th element in the `<list>`. Indexing starts from 0.

`<value> in <list>`
Returns `True` if `<value>` is in the `<list>`, `False` otherwise.

`<list1> + <list2>`
Concatenates `<list1>` and `<list2>`.

`len(<list>)`
Returns the number of elements in a list.

`range(N)`
Returns a list of numbers from 0..N

`range(N1, N2, N3)`
Returns a list of numbers starting from N1 and less than N3 incrementing by N3.

`<list>.sort()`
Sorts the `<list>` in ascending order.

`<list>.append(<value>)`
Appends the `<value>` at the end of `<list>`.

`<list>.reverse()`
Reverses the elements in the list.

Chapter 6

The if-statement in Python has the following forms:

```
if <condition>:
   <this>

if <condition>:
   <this>
else:
   <that>

if <condition-1>:
   <this>
elif <condition-2>:
   <that>
elif <condition-3>:
   <something else>
...
...
else:
   <other>
```

The conditions can be any expression that results in a True, False, 1, or 0 value. Review Chapter 4 for details on writing conditional expressions.

Chapter 7

The math library module provides several useful mathematics functions. Some of the commonly used functions are listed below:

`ceil(x)` Returns the ceiling of x as a float, the smallest integer value greater than or equal to x.

`floor(x)` Returns the floor of x as a float, the largest integer value less than or equal to x.

`exp(x)` Returns e**x.

`log(x[, base])` Returns the logarithm of x to the given base. If the base is not specified, return the natural logarithm of x (i.e., the logarithm to base e).

`log10(x)` Returns the base-10 logarithm of x.

`pow(x, y)` Returns x**y.

`sqrt(x)` Returns the square root of x.

Trigonometric functions

`acos(x)` Returns the arc cosine of x, in radians.

`asin(x)` Returns the arc sine of x, in radians.

`atan(x)` Returns the arc tangent of x, in radians.

`cos(x)` Returns the cosine of x radians.

`sin(x)` Returns the sine of x radians.

`tan(x)` Returns the tangent of x radians.

`degrees(x)` Converts angle x from radians to degrees.

`radians(x)` Converts angle x from degrees to radians.

The module also defines two mathematical constants:

`pi` The mathematical constant pi.

`e` The mathematical constant e.

Chapter 8

In this chapter we presented informal *scope rules* for names in Python programs. While these can get fairly complicated, for our purposes you need to know the distinction between a *local name* that is local within the scope of a function versus a *global name* defined outside of the function. The text ordering defines what is accessible.

Chapter 9 & 10

There were no new Python features introduced in this chapter.

Chapter 11

The only new Python feature introduced in this chapter was the creation of modules. Every program you create can be used as a library module from which you can import useful facilities.