CSC1102
STRUCTURED AND OBJECT-ORIENTED PROGRAMMING

Course Outline

– **Module 1 (16 hours): Structured Programming**

- Introduction to programming
- **Program structure**
- Branching and Iterations: Control flow statements

– Module 2 (16 hours): Data manipulation

- Functions

Previously..

- Introduction to programming
- Introduction to python

In This Lecture

- Python program structure
- Python Syntax
- Variables and Constants

Python Program Structure

- Python interpreter executes from top of the file (no need for a main function as the case in other languages such as `java` and C)
- A main function is a special function which is executed first every time the program is run.
- Python community has build a number of re-usable libraries. These can be installed and imported as modules into your program.
- Some modules are included in the standard library but also need to be imported into your program if needed.
- A python program there normally contains import statements

Python Program Structure: Sample program

```
import datetime
today = datetime.datetime.today()
year = today.year
print("We are in "+str(year))
```

Python Syntax

- All programming languages define their syntax.
- A statement an instruction that can be executed e.g `age=5, if age>10`
- In python:
  - Comments begin with a hash (#) symbol and the interpreter ignores them. Python has no multi-line comment symbols.
  - Blocks of code are indicated by indentation. Length of indentation in the same block must be constant.
  - White spaces are ignored if not used for indentation
  - Python reserves special keywords that must be used in a special way.

Python Syntax

```
#This program informs the user
#if he is an adult or not
age = input("Enter your age\n")

#space above and below is ignored

if int(age)<18:
    print('Your are a kid')
else:
    print('You are an adult')
```

Python Syntax

- A new line character marks the end of a statement in python.
- You can make a statement extend over multiple lines with the line continuation character (\). Parentheses ( ), brackets [ ], and braces { } can also be used

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

```
a = (1 + 2 + 3 +
     4 + 5 + 6 +
     7 + 8 + 9)
```

```
colors = ['red',
          'blue',
          'green']
```

Python Syntax: Docstrings

- Python uses Triple quotes to write docstrings.
- Python docstrings (documentation strings) are the string literals that appear right after the definition of a function, method, class, or module.

```
def area(l,w):
    """
    This function computes are of a rectangle
    :param l: length of the rectangle
    :param w: width
    :return: area of the rectangle
    """
    return l*w
```

Python Syntax: Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 33 keywords in Python 3.7. This number can vary slightly over the course of time.
- All keywords are lower case except True, False and None

Python Syntax: Keywords

- Keywords are the reserved words in Python.
- We cannot use a keyword as a variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 33 keywords in Python 3.7. This number can vary slightly over the course of time.
- All keywords are lower case except True, False and None

Python Syntax: Keywords

- You can view a list of keywords in python using:

```
import keyword
print(keyword.kwlist)
```

Python Syntax: Identifiers

- An identifier is a name given to entities like class, functions, variables, etc.
- It helps to differentiate one entity from another
- Identifiers can be a combination of letters in lowercase (**`a to z`**) or uppercase (**`A to Z`**) or digits (**`0 to 9`**) or an underscore
- An identifier cannot start with a digit and cant be a keyword

Python Variables

- A variable is a named location used to store data in the memory.
- We create a variable by simply giving it a name and value e.g:
  - `my_name = “john”`
  - `my_age = 12`
- You can view the memory location where your variable is stored by `typing id(variable_name)` e.g `id(my_age)`
- It is good practice to give variables meaningful identifiers. Can be achieved using:

```
snake_case
MACRO_CASE
camelCase
CapWords
```

Python Variables

- We can change variables by re-assigning them values
  - `my_name="Mary"`
- We can assign multiple variables in a single line e.g
  - `first_name,last_name,age = "Rashida", "Nagawa",30`
  - `x, y, z = 7`
- Unlike other languages, python doesn't explicitly define constants but you can conventionally create constants using upper case identifiers e.g
  - `PI = 3.14`

Python Variables

- We can change variables by re-assigning them values
  - `my_name="Mary"`
- We can assign multiple variables in a single line e.g
  - `first_name,last_name,age = "Rashida", "Nagawa",30`
  - `x, y, z = 7`
- Unlike other languages, python doesn't explicitly define constants but you can conventionally create constants using upper case identifiers e.g
  - `PI = 3.14`

Python Literals

- Literal is a raw data given in a variable or constant.
- Literals in Python:
  - Numeric Literals
  - String Literals
  - Boolean Literals
  - None literal

Python Literals: Numeric Literals

- Python has three numeric literals: Integer, Float, and Complex

```
a = 0b10110 #Binary Literals
b = 100 #Decimal Literal
c = 0o310 #Octal Literal
d = 0x12c #Hexadecimal Literal

#Float Literal
float_1 = 10.5
float_2 = 1.5e2

#Complex Literal
x = 3.14j

print(a, b, c, d)
print(float_1, float_2)
print(x, x.imag, x.real)
print(type(a), type(d),type(float_2),type(x))
```

```
10 100 200 300
10.5 150.0
3.14j 3.14 0.0
<class 'int'> <class 'int'> <class 'float'> <class 'complex'>
```

Python Literals: String Literals

- A string literal is a sequence of characters surrounded by quotes.
- Python uses single, double, or triple quotes for a string.

```
strings = "This is Python"
char = "C"
multiline_str = """This is a multiline string
with more than one line code.
"""
unicode = u"\u00dcnic\u00f6de"
raw_str = r"raw \n string"
print(strings,type(strings))
print(char,type(char))
print(multiline_str)
print(unicode)
print(raw_str)
```

```
This is Python <class 'str'>
C <class 'str'>
This is a multiline string
with more than one line code.

Ünicöde
raw \n string
```

Python Literals: Boolean

- A Boolean literal can have any of the two values: True or False.
- Python uses single, double, or triple quotes for a string.

```
x = (1 == True)
y = (1 == False)
a = True + 4
b = False + 10

print("x is", x)
print("y is", y)
print("a:", a)
print("b:", b)
```

```
x is True
y is False
a: 5
b: 10
```

Python Literals: None

- We use it to None that the field has not been created
- Evaluates to False if tested

```
choice = input("What will you eat? (Chicken or None)\n")
if choice == "Chicken":
    choice = 'Chicken'
else:
    choice = None
if choice:
    print(r"Didn't know you love chicken")
else:
    print('You have no choice')
    print(choice)
    print(type(choice))
    choice = 'None'
    print(choice)
    print(type(choice))
```