Revision for Pythonic Adventures

NSC 1002: Workshop 9

1 Dec 2020

Geoff Vallis

Basic Concepts

- **Program**: Set of **statements** written in a programming language (code), executed by the computer exactly as written.
- **Statement**: An instruction, as part of a program, that performs a specific task (e.g.x = 0.0, sets the value of x to 0.0, ($x \leftarrow 0.0$) and print(x) prints the value of x)
- **Algorithm**: Is a set of step by step **instructions** that perform a designed task (e.g. root finding, numerical integration, etc.)
- **Pseudo code**: Generic (non-language specific) code written in designing programs, useful for useful for developing and communicating ideas
- **Plot**: In the sense of scientific computing, a plot is a graphical representation of numerical data. Matplotlib, histograms etc.

Data

- **(Data) Type**: Is a classification of data, e.g. float, int, string
- **Float**: A *data type* that represents real numbers (e.g. 3.14)
- **Integer**: A *data type* that represents integer numbers, (e.g. -1)
- **Boolean**: A *data type* that represents logical data (True or False)
- **String**: A *data type* that represents a group of characters e.g. “Hello World!”
- **Data structure**: Special arrangement of data that allows for ease of use, access and/or storage

Data Structure

- **List**: A **data structure**, surrounded by [ ] , are mutable and (usually) used for (homogeneous) data , (e.g.: `[-1, 0, 1, 2]` or `['A', 'B', 'C', 'D']`).
- **List element**: Each member of a **list**. For some list L, its first element is accessed by L[0], and its 3rd element by L[2] (For L=[2.0, 4.0, 8.0, 16.0], L[0] == 2.0, and L[2] == 8.0). (A convention used by Python and C)
- **Tuple**: *data structure* that is defined by a **,** but is often surrounded by parentheses, that is by ( ) , are (immutable) and (often) used for (heterogeneous) data, e.g. (117, 'John')
- **Array**: A *data structure* from the module numpy, that stores (homogeneous) data such as floats and integers, can be multi-dimensional. Useful for working with scientific data and (faster) than lists. Need numpy. The most used data type in *computational Python.*

Errors

- **Syntax error**: Error that arises from incorrect “language” or formatting in a program.
  E.g. `print('Hello)` is missing ' .
- **Type error**: Raised when a variable used is the incorrect data type
  e.g. X = 1.0 / 'Cat'
- **Other errors:** User errors, semantic errors etc. The code runs but gives the wrong answer! (Hardest to debug).

Program Flow

- **Control structure**: Statement or set of statements that tell a program when and how to perform certain actions e.g. **if**, statements **while** loops
- **Conditional**: A control structure that checks for boolean values of statements, then performs different actions based on them
  e.g. `if x > 0:  elif x == 0:  else:`
- **Relational operator**: Compares two objects (of the same type)
  e.g. `>, <, <=, >=, ==, !=`

Program Flow

- **Loop**: A *control structure* that repeats a block of code *for* a certain number of **iterations**, or *while* a condition is met.
  E.g. `while k > 0:`
- **Iteration**: Is a(n) individual repetition as part of a loop, or the process of looping
- **Iterator variable**: Variable that defines the current step of iteration within the loop.
  In the loop `for i in range(10):` the iterator variable is `i`
- **Indentation:** Commands within for loops or if statements are indented, normally by four spaces. The end of a loop is marked by the end of the indentation.

Code Re-use

- **Function**: Arrangement of code that accepts input, performs a task, and *often* returns output.
  (e.g `print()`, `numpy.sin()`, `lambda` functions...)

- **Defining a function**: Example function definition that takes two input arguments, and returns the sum of the first squared and the second times 3.0

```
def new_fcn(x, y):
    z = x**2+3.0*y
    return z
```

- **Calling a function**: Write an example of calling the function you've written, assigning its output to a variable:

```
x1 = 3.0
y1 = 2.0
t = new_fcn(x1, y1)
```

  As a result of this, $t = 15.0$

Debugging example 1

There are 3 errors in the following code, please correct them, and detail what the corrected code does

```
1  x = 15
2
3  if x < y:
4      print(x)
5  elif y < x
6      print(y)
7  else:
8      print('The two numbers are equal)
```

Missing declaration (initialisation) of y, (line 2) missing " : " (line 5), missing end quotation, line 8.

Debugging example 1

There are 3 errors in the following code, please correct them, and detail what the corrected code does

```
1  x = 15
2  y = 10
3  if x < y:
4      print(x)
5  elif y < x:
6      print(y)
7  else:
8      print('The two numbers are equal')
```

Missing declaration (initialisation) of y, (line 2) missing " : " (line 5), missing end quotation, line 8.

Debugging example 2

Identify what type of error will occur with this code and state how this error could be corrected.

```
1  def hello():
2      """
3      A function that prints out 'Hello' x number of times
4      where x is given by the user. Returns none.
5      """
6      # Read in the integer x from the user
7      x = int(input('How many times you would like Hello printed?'))
8      print("x" * x)
9
10 hello()
```

The error is a *semantic* error, the letter “x” is printed *x* number of times, rather than “Hello”.

Debugging example 2

Identify what type of error will occur with this code and state how this error could be corrected.

```
1  def hello():
2      """
3      A function that prints out 'Hello' x number of times
4      where x is given by the user. Returns none.
5      """
6      # Read in the integer x from the user
7      x = int(input('How many times you would like Hello printed?'))
8      print("Hello" * x)
9
10 hello()
```

The error is a *semantic* error, the letter "x" is printed *x* number of times, rather than "Hello".

Debugging example 3

```
1  divisible():
2      """
3      A function that takes in two integers from the user,
4      x and y, and calculates whether x is divisible by y.
5      Returns none.
6      """
7
8      # Read in the integer x from the user
9      x = input('Please enter your first number: ')
10
11     # Read in the integer y from the user
12     y = input('Please enter your second number: ')
13
14     if x % y = 0:
15         print('{} is divisible {}'.format(x, y)
16     else:
17         print('{} is not divisible {}'.format(x, y))
18
19 divisible()
```

- ☐ The function definition is incomplete (missing "**def**" on line 1)
- ☐ The input variables (x and y) are not converted to integers as indicated (lines 9 and 12)
- ☐ Assignment (=) has been used rather than check for equality (==) on line 14

Debugging example 3

```
1  def divisible():
2      """
3      A function that takes in two integers from the user,
4      x and y, and calculates whether x is divisible by y.
5      Returns none.
6      """
7
8      # Read in the integer x from the user
9      x = int(input('Please enter your first number: '))
10
11     # Read in the integer y from the user
12     y = int(input('Please enter your second number: '))
13
14     if x % y == 0:
15         print('{} is divisible {}'.format(x, y))
16     else:
17         print('{} is not divisible {}'.format(x, y))
18
19 divisible()
```

- ☐ The function definition is incomplete (missing "**def**" on line 1)
- ☐ The input variables (x and y) are not converted to integers as indicated (lines 9 and 12)
- ☐ Assignment (=) has been used rather than check for equality (==) on line 14

Debugging example 4

Please discuss what the following function does:

```
1  def temp(lst_in):
2      """Takes a list as input, returns it in reverse order."""
3      # Initialize a list
4      r = []
5
6      # Get original size
7      lsize = len(lst_in)
8
9      # Loop while output list is smaller than the input list
10     while lsize > len(r):
11
12         # lst_in[-1] refers to the last element in lst_in
13         element = lst_in[-1]
14
15         # Remove last element in the input list
16         lst_in = lst_in[:-1]
17
18         # Appends element from lst_in to the output list r
19         r.append(element)
20
21     # Finally return r
22     return r
```

Debugging example 5

```
def birthdays(x):
    """Print out the ages a group of x friends and
    the average age of the group. Returns None.
    """
    # Initialise the list that you will populate
    Ages = []

    # count how many friends have entered their data
    count = 1

    while count <= x:
        # Read in the ages from the user.
        d = int(input('Please enter your age '))

        # Append the new data to the list Ages
        Ages.append(d)
        count += 1

    print("\n The ages of your friends are:")
    # Iterate through list using indexing,
    # printing each element
    l = len(Ages)
    for n in range(l):
        print(Ages[n])

    Average = sum(Ages) / len(Ages)
    print('The average age is {}'.format(Average))
```

- Variable “cout” is not defined (should be *count*) on line 14
- `count` is not incremented within the loop
- Missing end quotation on line 22
- Variable “Age” is not defined on line 26, missing an “s”.

General questions?