UECM1703 Introduction to Scientific Computing
Topic 1: Basic Programming

Dr Liew How Hui

Oct 2020

About This Subject

Week 1: Basic Python (Liew HH); Lab 1: Chong Kar Ming

Week 2: Array (Liew HH); Lab 2: Chong Kar Ming

Week 3: Test (20%) during lecture (1.5 hours); Lab 3: Chong Kar Ming

Week 4: Graphics 1 (Lecture + Lab 4 + Assignment, Dr Goh YK)

Week 5: Graphics 2 (Lecture + Lab 5, Dr Goh YK)

Week 6: Modelling 1 (Lecture + Lab 6 + Quiz, Dr Jeeva)

Week 7: Modelling 2 (Lecture + Lab 7 + Quiz, Dr Jeeva)

About This Subject

After Week 7: Final Assessment (3 Hours)

- 2.5 Hours Working
- 0.5 Hour scan and submission
- Write in Word, Convert to PDF
- Submit PDF file

Outline

Background

Scientific Modelling: Use mathematical formulas to describe a scientific problem. For example, a forced pendulum ($ma = F$):

$$m\frac{d^2\theta}{dt^2} = \frac{-mg\sin\theta - \beta\frac{d\theta}{dt} + \tau\sin(\omega t)}{L}$$

Scientific Computing: Use numerical methods to approximate the solutions of the mathematical formulas and computer software to calculate the values.

$$\frac{\theta_{i+1} - 2\theta_i + \theta_{i-1}}{h^2} \approx -\frac{g}{L}\sin\theta_i - \frac{\beta}{mL}\frac{\theta_{i+1} - \theta_{i-1}}{2h} + \frac{\tau}{mL}\sin(\omega t_i)$$

Computer Software (cont)

Compiled Programming Languages:

- Ada (1983–)
- C ($\approx$ 1970–)
- C++ ($\approx$ 1980–)
- C# (2001–)
- D (2001–)
- Fortran (1957–)
- Haskell (1990–)
- Java (1995–)
- Kotlin (2011–)

Ref: `https://en.wikipedia.org/wiki/History_of_programming_languages`

Computer Software (cont)

Interpreted Programming Languages:

- Julia (2012–)
- LabVIEW (1986–)
- MATLAB (1984–)
- Python (1990–)
- Ruby (1995–)
- R (1993–)

Ref: `https://en.wikipedia.org/wiki/History_of_programming_languages`

Computer Software (cont)

Why Python?

Answer:

- "Easy" to learn
- Many online resources: just search "Python (language)" on the Internet.

How to work with Python?

Answer: Python Software Environments:

- Text Editor + Python + Shell (`https://www.youtube.com/watch?v=QKBcHuA3VJE`)
- Spyder: IDE (all in one)
- Jupyter Notebook: I don't recommend

Outline

Python (Scientific) Programming

My approach to (scientific) programming:

- Start from basic data types: Booleans, Integers, Floating-point numbers, Strings; & functions
- Then assignment, selective & loop statements
- Then containers: Tuple, List, Dictionary (Hash), Set, ...
- Then Numpy array and array operations (Week 2)
- Then graphics (Week 4& 5: Dr Goh Yong Kheng)
- Then modelling (Week 6& 7: Dr Jeeva)

Outline

Basic Data Types (cont)

Booleans

- Syntax: True, False
- and, or, not, &, |
- Applications: if statement, while loop

Basic Data Types

Integers

- Syntax: +1023, -1_0000 (underscore allow)
- Range: ..., -3, -2, -1, 0, 1, 2, 3, ... (can be arbitralily large)
- 0b111, 0o123, 0xABC, int(”123”, 7)
- +, -, *, /, //, **, %, abs()
- Bit arithmetic: &, |, ^, <<, >>
- >, >=, ==, ! =, <=, <

Basic Data Types (cont)

Floating-point Numbers

- Scientific notation: $\pm$ X.DDDDD $\times 10^{y}$
- Into computer: $\pm$ 1.MMMMM $\times 2^{E}$
- Syntax: -.1, +3.14, 1_0000.0 (underscore allow)
- +, -, *, /, **, abs()
- >, >=, ==, ! =, <=, <
- constants: math.e, math.pi, math.tau, math.inf, math.nan

Basic Data Types (cont)

(Unicode) Strings

- Syntax: ”a string”, ’a string’
- For IO and “categorical data”
- +, len(), ord(), chr(), str()

What is Unicode?

- A generalisation of ASCII
- https://en.wikipedia.org/wiki/Unicode

Basic Data Types (cont)

What is ASCII?

- https://en.wikipedia.org/wiki/ASCII
- $0101010_2 = 42 =$ '*'
- $0101011_2 = 43 =$ '+'
- $0110000_2 = 48 =$ '0'
- $1000001_2 = 65 =$ 'A'
- $1100001_2 = 97 =$ 'a'

Text Editors work with ASCII & Unicode

- Atom Editor
- VS Code
- Difficult to learn: Vim, Emacs

Functions

A **function** takes in **parameters** and returns a **value**.

- Defining our own function:

```
def f(x,y,z):
    return a value derive from x,y,z
```

Example:

```
def myaverage(x,y,z):
    return (x+y+z)/3.0
```

- Standard functions provided by Python are grouped into “modules”/“libraries”: E.g.

```
from math import *      # everything
from cmath import sqrt # specific
```

Functions (cont)

From `math` module:

- Exponential function e^x: `exp(x)`
- Logarithmic functions: `log(x)`, `log2(x)`, `log10(x)`
- Power function x^n: `pow(x,n)`, `sqrt(x)`
- Radian $\leftrightarrow$ Degree: `radians(degree)`, `degrees(radian)`
- Trigonometric & Hyperbolic functions: `sin(x)`, `cos(x)`, `tan(x)`, `sinh(x)`, `cosh(x)`, `tanh(x)`
- Inverse Trigonometric functions: `asin(x)`, `acos(x)`, `atan(x)`, `atan2(y,x)`

Functions (cont)

- Distance: `dist(x,y)`, `hypot(...)`
- Truncating functions: `ceil(x)`, `floor(x)`, `trunc(x)`
- "Check" functions: `isclose(a,b)`, `isfinite(x)`, `isinf(x)`, `isnan(x)`
- Combinatorics: `comb(n,k)`, `perm(n,k)`, `factorial(n)`, `gcd(x,y)`

UEEP2083 Computational Physics Final Exam Sept 2015, Q1(c)(i)

Solve the following nonlinear equation using Python

$$\sin(2x) = 2\cos(x). \qquad \text{(3 marks)}$$

Answer:

```
1 from math import sin, cos
2 def f(x): return sin(2*x) - 2*cos(x)
3 from scipy.optimize import fsolve
4 guess = 0.0
5 est_answer = fsolve(f, guess)
6 print("Est. x =", est_answer)
```

UEEP2083 Computational Physics Final Exam Sept 2015, Q1(c)(iii)

Create the Python commands to solve integral

$$\int_0^{10} (x^{x/2} + 0.9x)dx. \qquad \text{(2 marks)}$$

[Hint: `scipy.integrate.quad`]

Examples: Using standard functions

Write down the Python command to calculate $\sin 10° \sin 30° \sin 50° \sin 70°$.

Write down the Python command to calculate $\tan 6° \tan 42° \tan 66° \tan 78°$.

Write down Python command to calculate $\cos 42° \cos 18° - \sin 42° \sin 18°$. Are you able to find the value exact using trigonometric equality?

Final Exam Sept 2012, Q2(b)

Write a Python function that converts a temperature in Fahrenheit to a temperature in Celcius according to the following equation

$$[^\circ C] = \frac{5([^\circ F] - 32)}{9}.$$

Show an example of using the function `f2c` in the Python command window. (8 marks)

Example: 2019 Final Exam Q3(c)

In the late 1960's, book publishers realised that they needed a uniform way to identify all the different books that were being published throughout the world. In 1970 they came up with the International Standard Book Number system. Every book, including new editions of older books, was to be given a special number, called an ISBN, which is not given to any other book. The check digit x_{10} of a 10-digit ISBN $x_1 x_2 \cdots x_9 x_{10}$ is given by the formula:

$$\begin{aligned} x_{10} = (11 - (10x_1 + 9x_2 + 8x_3 \\ + 7x_4 + 6x_5 + 5x_6 + 4x_7 \\ + 3x_8 + 2x_9 \bmod 11)) \bmod 11. \end{aligned}$$

Example: 2019 Final Exam Q3(c)

Defining our own function (and using "assignment" statements)

If $x_{10} = 10$, it is represented as '`X`'. Write a Python **function** to implement `checkdigit10(x1,...,x9)` which takes an ISBN of the form $x_1 x_2 \cdots x_9$ as 9 parameters and returns a digit x_{10} as a character.

(5 marks)

Example: 2019 Final Exam Q3(c)

A sample implementation is shown below.

```
1  def check_digit_10(x1,x2,x3,x4,x5,x6,x7,x8,x9):
2      x10 = 10*x1+9*x2+8*x3+7*x4+6*x5+5*x6+4*x7+3*x8+2*x9
3      x10 = 11 - (x10 % 11)
4      return str(x10) if x10 != 10 else 'X'
5
6  print(check_digit_10(0,8,0,4,4,2,9,5,7))
7  print(check_digit_10(0,8,5,1,3,1,0,4,1))
8  print(check_digit_10(0,3,0,6,4,0,6,1,5))
9  print(check_digit_10(9,9,9,2,1,5,8,1,0))
```

- Correct definition of function . . . . . [1 mark]
- Proper use of for loop . . . . . [1 mark]
- Correct translation of formula to Python . [3 marks]

Outline

Programming Statement(s)

Ref: `https://en.wikipedia.org/wiki/Statement_(computer_science)`

In programming, a **statement** is a **syntactic unit** of an imperative programming language that expresses some **action** to be carried out. A program written in such a language is formed by a sequence of one or more statements. A statement may have internal components (e.g., expressions).

Programming Statement(s)

- Assignment statements: Storing ‘object’s
  - variable name mostly starts with a–z, A–Z, _, after that digits can be used
  - E.g. a0 = 1 + 2 + 3, MyVariable = 1 < 3 < 2
  - Special: +=, -=, *=, /= (Abbrev: x = x + expr)
  - Special: _ denotes previous / throw away value
- If statement : decision / selection
- For loop : going through a list
- While loop : indefinite loop
- Loop controls: break and continue

Selective Statements

The ability for a computer program to “select” or “choose” is crucial because it allows us to implement “decision”. In Python, “selection” is achieved by the various forms of the “if” selective statements. The following is a list of possible forms of the “if” statements:

- **If only**:

```
if condition:
    do_something_block
```

Selective Statements (cont)

- **If and else only**:

```
if condition:
    do_something_block
else:
    do_otherthing_block
```

- **One line (ternary operator) form**:

```
res = true_result if condition else false_result
```

Selective Statements (cont)

- **Multiple conditions**: A sample syntax is given below

```
if condition1:
    do_task1_block
elif condition2:
    do_task2_block
elif condition3:
    do_task3_block
else:
    do_default_task_block
```

Selective Statements (cont): Example [If and else only]

```
disc = b**2 - 4*a*c
if disc >= 0:
    x1 = (-b+sqrt(disc))/2/a
    x2 = (-b-sqrt(disc))/2/a
else:
    x1 = None
    x2 = None
```

Anything wrong with the above if statement?

Example: Sept 2015 FE, Q1(d)

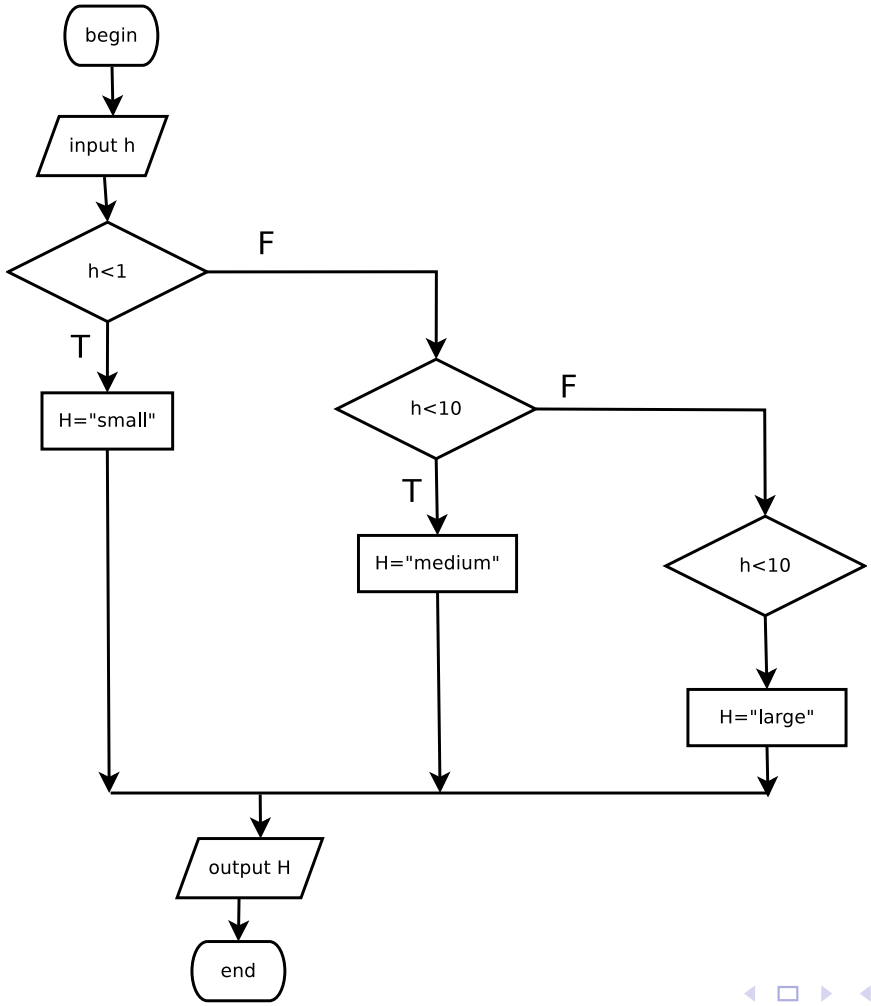
Find four errors in the following function, and identify each as a syntax error or an algorithmic error. Then correct the errors in the code.

```
1  function B = is_digit(Character)
2  % is_digit - Boolean function, return True if
3  %            C is a character digit '0' - '9'
4  if ischar(C) and C >= 48 and C<57
5      B = 1;
6  elseif
7      B = 0;
8  end
```

After correction, translate it to a **One line (ternary operator) form** Python statement.

Example: Sept 2012 FE, Q2(c)

Write a Python script that implements the following flowchart:



Example: Sept 2015 FE, Q5(b)

Hurricanes are categorised based on wind speeds. The following table shows the category number for hurricanes with varying wind ranges and what the storm surge is (in feet above normal).

Category	Wind speed	Storm surge
1	74 – 95	4 – 5
2	96 – 110	6 – 8
3	111 – 130	9 – 12
4	131 – 155	13 – 18
5	> 155	> 18

Write a script that will prompt the user for the wind speed, and will print the hurricane category number and typical storm surge. [**Multiple conditions**]

Example: Sept 2012 FE, Q3(a)

Consider the quadratic equation $ax^2 + bx + c = 0$. The solution to this equation is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

The term "$b^2 - 4ac$" is called the discriminate of the equation. The nature of the discriminate determines the number and type of the roots as follows:

discriminate value	number and type of roots
positive	2 distinct real roots
negative	2 complex roots
zero	1 repeated root

Example: Sept 2012 FE, Q3(a) cont

Write a Python program to solve for the roots of a quadratic equation. The program should:

- read the input values of a, b, and c;
- calculate the roots; and
- display the outputs, including a statement about the type of the roots, i.e. “There are 2 distinct real roots.”

Example: Sept 2012 FE, Q3(a) cont

The output should be displayed as below.

```
This program solves for the roots of a quadratic
  equation of the form ax^2 + bx + c = 0.
input the value of the coefficient "a": 1
input the value of the coefficient "b": 2
input the value of the coefficient "c": 3
the discriminate is -8.000000
the equation has two complex roots
x1 = -1.000000 + 1.414214 i
x2 = -1.000000 - 1.414214 i
```

Loops

Rationale:

When we want to solve a “repeating” problem, for example, if there are thirty quadratic equations with different sets (a, b, c), are we going to write a script with 30 lines as follows?

```
solve_quadratic_equation(1,2,3)
solve_quadratic_equation(4,5,6)
... another 27 lines ...
solve_quadratic_equation(5,-2,7)
```

Loops (cont)

Or is it possible two write a script with less lines such as?

```
go through a,b,c in (1,2,3), (4,5,6), ...
  another 27 pairs of (a,b,c), (5,-2,7) and
  perform solve_quadratic_equation(a,b,c)
```

The “go through all cases” statements or looping statements are called “loops”.

Loops (cont)

Python provides two kinds of “loops” as well as recursion to support looping statements.

- **while loop** statement:

```
while condition:
    block
```

- **for loop** statement:

```
for variable in somerange:
    block
```

What happens if we want to exit a loop or skip the rests and continue to the next value in a loop? We can use `break` and `continue`.

Loops: For statements

A ‘for’ statement with the variable ‘i’ going through the ‘range’ from 1 to 20 (< 21) incrementing by 2:

```
for i in range(1,21,2):
    print("the square of {i} is {isq}".format(
          i=i, isq=i**2))
```

In mathematics, we have $\sum_{i=1}^{n} f(i)$, how to express in Python?

```
thesum = 0
for i in range(1,n+1): thesum += f(i)
```

Example: Sept 2016 FE, Q3(a)

Consider the alternating series

$$S = \sum_{n=1}^{k} (-1)^{n+1} \frac{1}{n \ln(n+1)}.$$

Write a Python expression to calculate S for $k = 100$.

Example: Sept 2014 FE, Q2(c)

Write a Python code to calculate the following summation

$$3(2+1)+4(3+2+1)+5(4+3+2+1)+$$
$$6(5+\cdots+1)+\cdots+1000(999+\cdots+1).$$

(10 marks)

Example: Propositional Logic

Generate the truth table of the statement $\sim s \rightarrow (p \wedge (q \vee r))$ where p, q, r, s are atomic statements.

Example: Probability and Statistics

Use Python's `scipy.stats.norm.cdf` function to generate the following cumulative normal distribution table. [Compare this to `https://home.ubalt.edu/ntsbarsh/Business-stat/StatistialTables.pdf`]

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.00	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.10	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.20	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.30	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.40	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.50	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.60	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.70	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.80	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.90	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.00	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621

Example: Number Theory

Study the following Python script which generates a multiplication table for simple finite field and write down the output:

```
1  prime_number = 5
2  print("  |",end="")
3  for i in range(prime_number):
4      print(f"{i:5d}",end="")
5  print("\n" + "-"*(3+5*prime_number))
6  for i in range(prime_number):
7      print(f"{i} |", end="")
8      for j in range(prime_number):
9          v = i*j % prime_number
0          print(f"{v:5d}",end="")
1      print()
```

Example: 2019 Final Exam Q3(a)

Demonstrate the working of the following Python program script by stepping through it and write down the output.

```
1  P=7
2  print("   |",end="")
3  for i in range(P):
4      print(f"{i:5d}",end="")
5  print("\n" + "-"*(3+5*P))
6  for i in range(P):
7      print(f"{i} |", end="")
8      for j in range(P):
9          v = i*j % P
0          print(f"{v:5d}",end="")
1      print()
```

(8 marks)

Example: 2019 Final Exam Q3(a)

Answer: The following table will be generated:

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Students need to perform modulus calculations with 7.

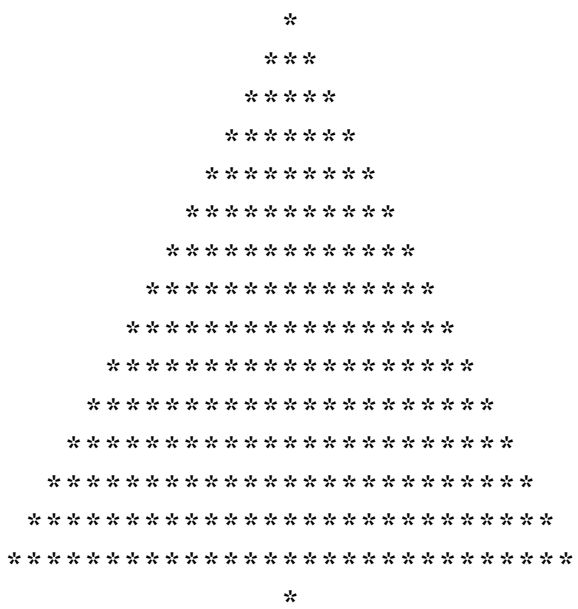
. . . . . [8 marks]

Example: ASCII Art

In the 1970s when laser printer is super-expensive and the most commonly available printers were line printers (i.e. texts are being printed line by line). In the 1980s, the computer terminals were 50×70 grid and plotting sine, cosine, tangent, etc. on a text terminal screen.

Example: ASCII Art (cont)

Construct a Python program to generate the ASCII "Christmas tree" on the left and the ASCII sine function (30×15 screen) on the right.



```
-1              0              1
--------------+--------------
                *
                        *
                                *
                                *
                                *
                        *
                *
        *
    *
*
*
  *
          *
```

Example: Logo Graphics

What does the following Python script do?

```
1  import turtle as tt
2
3  # https://www.youtube.com/watch?v=b6AcYxIxXMA
4  # Hayley Denbraver - Recursion, Fractals, and the Python Turtle Module
5  hayley_turtle = tt.Turtle()
6  hayley_turtle.color("blue")
7  hayley_turtle.pensize(4)
8  hayley_turtle.shape("turtle")
9  hayley_turtle.speed(7)
0
1  wn = tt.Screen()
2  side_length = 200
3  num_of_trig = 15
4  angle = 360//num_of_trig
5  for triangle in range(num_of_trig):
6      for i in range(3):
7          hayley_turtle.forward(side_length)
8          hayley_turtle.right(120)
9      hayley_turtle.right(angle)
0  wn.exitonclick()
```

While Loop

In contrast to “for loop”, we seldom use “while loop”. However, when we don’t know when to stop, a “while loop” statement augmented with `break` statement is preferred and illustrative examples are given below.

Example: Sept 2012 FE, Q1(b)

Write a Python script to generate a sequence of random number and determine the number of attempt it takes to obtain the first random number that is in between 0.5 and 0.55. (9 marks)

Sample Answer:

```
import random
#random.seed(2020)
count = 0
while True:
    anum = random.random()
    count += 1
    if 0.5 < anum < 0.55:
        print("Number of attempt taken =",count)
        break  # exit while loop
```

Example: Sept 2014 FE, Q2(a)

Consider the following MATLAB function file:

```
function [A,B] = question2a(x,y,z)
W = -17; A = -5; B = -5;
while W<=0
    if (y<x) | (-z<=x)
        A = A+2*abs(x-y);
        B = B+x-y+3*z;
        W = W+B;
    elseif (x<-y) & (z<10*x)
        A = A-x+y;
        B = B+5*y-2*z+x;
        W = W+2*A+B;
    else
        A = A+abs(x-y+z);
        B = B+A;
        W = W+8;
    end
end
```

Example: Sept 2014 FE, Q2(a) cont

(i) Write down and explain the values of `A` and `B` for the following MATLAB command.

```
>> [A, B] = question2a(-4,-1,-2)
```

(ii) Rewrite the MATLAB function as a Python function.

Example: Sept 2015 FE, Q3(c)

Consider the following MATLAB code.

```
1  function index = mycode(vec,key)
2      Len = length(vec);
3      i = 1;
4      while i < Len && vec(i) ~= key
5          i = i+1;
6      end
7      if vec(i) == key
8          index = i;
9      else
10         index = 0;
11     end
12 end
```

Example: Sept 2015 FE, Q3(c)

1. Write down the value for `C1` and `C2` below.

```
>> values = [85 70 100 95 80 91];
>> C1 = mycode(values, 70);
>> C2 = mycode(values, 93);
```

(4 marks)

2. Write a Python function which achieves similar functionality to MATLAB `mycode` (beware that Python index starts from 0, not 1).

Example: Sept 2013 FE, Q5(b)

Write a Python script to determine the greatest value of n that can be used in the sum $2 + 4 + 6 + 8 + \cdots + 2n$ and get a value of less than 200. (12 marks)

Example: Sept 2017, Q3(c)

Write a Python script to calculate the following summation not exceeding 100. Script must give total sum and the last element as output.

$$\frac{4}{5} + \frac{5}{6} + \frac{6}{7} + \frac{7}{8} + \frac{8}{9} + \cdots . \qquad \text{(6 marks)}$$

Outline

Container Data Types

Python provides the *derived data structures*, called "collections" in many programming language, called "containers" in C++, to store data.

Despite being slow, "collections" are easy to use. The four main "collections" in Python are:

1. *Tuple*.
2. *List*.
3. *Set*.
4. *Dictionary*.

Container: Tuple

A tuple is constructed using curved brackets and has virtually no methods associated with it.

Example: `(1,1.0,"Helo")`.

Example: `(x, y, z) = (1, 2, 5)` or `x, y, z = 1, 2, 5` (the brackets can be ignored in assignment statement)

Application:

- Simple assignment: a,b,c = 1,2,3
- Encoding error: (result, error)
- Construct array of certain shape: np.zeros((2,2,3))

Container: List

A list is normally constructed using square bracket: `[1, 2, 3]` or `list()` for programming construct. The important operations associated with it are `append`, `clear`, `copy`, `insert`, `sort`.

A Python "list" is normally used as an "ordered queue". For example, if we keyed in an integer sequence "8 18 28 48" as our lucky numbers, then we can store in Python's list as follows.

```
aString = input("Enter a list of integers (separated by a space): ")
anIntList = [int(i) for i in aString.split(" ")]
```

Example: Horner's method

A polynomial evaluation method can be calculated using `https://en.wikipedia.org/wiki/Horner's_method`

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + xa_n)))).$$

Implement the Horner's method as a Python function `horner(coeffs,x)` which takes in the coefficients $a_0, a_1, \cdots, a_n$ of the polynomial $p(x)$ as `coeffs` and a value `x` and then returns the value of the polynomial $p(x)$ at `x`.

Example: Horner's method (cont)

$$p(x) = 1 + 4x + 9x^2 + 8x^3 = 1 + x(4 + x(9 + 8x))$$

Let coeffs = [1,4,9,8]

Horner's method steps:

- y = 8
- y = 9 + x*y $= 9 + 8x$
- y = 4 + x*y $= 4 + x(9+8x)$
- y = 1 + x*y $= 1 + x(4 + x(9+8x))$

Example: Horner's method (cont)

Using for loop:

- n = len(coeffs)
- y = coeffs[n-1]
- for i in range(n-2,-1,-1):
  - y = coeffs[i] + x*y
- return y

Example: 2019 Final Exam Q3(b)

The *Horner's method* is a polynomial evaluation method expressed by

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + xa_n)))).$$

By using this expression, implement the Horner's method as a Python function `horner(coeffs,x)` which takes in two parameters, i.e. the coefficients $a_0, a_1, \cdots, a_n$ of the polynomial $p(x)$ as `coeffs` and a value `x` and returns the value of the polynomial $p(x)$ at `x`. (7 marks)

Example: 2019 Final Exam Q3(b)

The marking for the implementation is as follows:

```
1  def horner(coeffs,x):                    # [1 mark]
2      deg = len(coeffs)                    # [1 mark]
3      y = coeffs[deg-1]                    # [1 mark]
4      for j in range(N-2,-1,-1):           # [1 mark]
5          y = coeffs[j] + x*y              # [2 marks]
6      return y                             # [1 mark]
```

Container: List (cont)

A list is powerful because it can represent any discrete structure theoretically (limited by computer storage in practice). The following example illustrates how a list can be used to represent a matrix. However, it is inconvenient to perform matrix multiplication with a list.

```
A = [[1,2],[3,4]]              # represents a 2x2 matrix A as a list of list
B = [[8,6,7,9],[3,8,5,6]]      # represents a 2x4 matrix B
# The matrix multiplication B x A is
C = [
  [A[0][0]*B[0][0]+A[0][1]*B[1][0], A[0][0]*B[0][1]+A[0][1]*B[1][1],
   A[0][0]*B[0][2]+A[0][1]*B[1][2], A[0][0]*B[0][3]+A[0][1]*B[1][3]],
  [A[1][0]*B[0][0]+A[1][1]*B[1][0], A[1][0]*B[0][1]+A[1][1]*B[1][1],
   A[1][0]*B[0][2]+A[1][1]*B[1][2], A[1][0]*B[0][3]+A[1][1]*B[1][3]]
]
```

Container: List (cont)

We can compare this to the matrix multiplication formula found on the website `https://en.wikipedia.org/wiki/Matrix_multiplication` except that Python's index start from 0 instead of 1.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad i = 1, ..., m; \quad j = 1, ..., p.$$

From this example, we can see that Python's "list" structure is not friendly for the linear algebra arithmetic.

Final Exam Oct 2018, Q1(b), CO1

The dot product of two vectors $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ and $\boldsymbol{y} = (y_1, y_2, \cdots, y_n)$, $\boldsymbol{x} \cdot \boldsymbol{y}$, is defined as

$$\boldsymbol{x} \cdot \boldsymbol{y} = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

The angle θ between two arrays is defined by the following relation

$$\cos\theta = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} \times \sqrt{\boldsymbol{y} \cdot \boldsymbol{y}}}.$$

Implement a **Python function** `theta` to calculate the angle θ (in **degree**) if you are given two arrays `a=[a1,a2,a3,a4]` and `b=[b1,b2,b3,b4]`. You **must** write down the proper import statements. If you use the Numpy module, you must prefix the Numpy functions

Containers (cont)

Conclusion: List is super-powerful but slow.

Application:

- Store arbitrary items: [”string”, 1, 3.14, True]
- Programming: len(x), x[index], x.append(e), x.pop(), x.reverse(), sorted(x), x.sort()
- Construct Numpy array
- Construct tree?

Containers: List (cont)

Maths:

$$\{n^2 \ : \ n \in \{1, 2, 3, \cdots, 10\}\}$$
$$=\{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$$

List comprehension:

```
[n**2 for n in range(1,11)]
```

In general,

```
[f(x) for x in S if predicate(x)]
```

Example: ISBN-13

In the late 1960's, book publishers realised that they needed a uniform way to identify all the different books that were being published throughout the world. In 1970 they came up with the *International Standard Book Number* system. Every book, including new editions of older books, was to be given a special number, called an ISBN, which is not given to any other book. ISBN is changed from a 10-digit system to 13-digit system since 1 January 2007. The check digit of the ISBN-13 is given by the formula

$$x_{13} = \begin{cases} r, & r < 10, \\ 0, & r = 10 \end{cases}$$

Example: ISBN-13 (cont)

where $r = \Big(10 - \Big(x_1 + 3x_2 + x_3 + 3x_4 + x_5 + 3x_6 + x_7 + 3x_8 + x_9 + 3x_{10} + x_{11} + 3x_{12}\Big) \bmod 10\Big)$ (Ref: `https://en.wikipedia.org/wiki/International_Standard_Book_Number`).

Provide an implementation of the ISBN-13 check digit as a Python function `check(isbn)` which takes in a list of digits and return a check digit.

Find the check digit for `978-0-306-40615-?`.

Example: 2019 Final Exam Q3(c)

In the late 1960's, book publishers realised that they needed a uniform way to identify all the different books that were being published throughout the world. In 1970 they came up with the International Standard Book Number system. Every book, including new editions of older books, was to be given a special number, called an ISBN, which is not given to any other book. The check digit x_{10} of a 10-digit ISBN $x_1 x_2 \cdots x_9 x_{10}$ is given by the formula:

$$\begin{aligned} x_{10} = (11 - (10x_1 + 9x_2 + 8x_3 \\ + 7x_4 + 6x_5 + 5x_6 + 4x_7 \\ + 3x_8 + 2x_9 \bmod 11)) \bmod 11. \end{aligned}$$

Example: 2019 Final Exam Q3(c)

If $x_{10} = 10$, it is represented as '`X`'. Write a Python **program script** to implement the function `checkdigit10(isbn)` which takes an ISBN of the form $x_1 x_2 \cdots x_9$ as a string of length 9 and returns a digit x_{10} as a string of length 1. (5 marks)

Example: 2019 Final Exam Q3(c)

A sample implementation is shown below.

```
1  def check_digit_10(isbn):
2      isbn = list(isbn.replace('-','').replace('?',''))
3      assert len(isbn) == 9
4      x10 = 0
5      for i in range(len(isbn)):
6          x10 += (10-i)*int(isbn[i])
7      x10 = (11-x10) % 11
8      return str(x10) if x10 != 10 else 'X'
9
0  print(check_digit_10('0-8044-2957'))
1  print(check_digit_10('0-85131-041'))
```

- Correct definition of function . . . . . [1 mark]
- Proper use of for loop . . . . . [1 mark]
- Correct translation of formula to Python . [3 marks]

Example: Sept 2015 FE, Q2(c)

1. Write a Python function (named `swap`) that swaps the elements in a list. (3 marks)
2. Write a Python function (named `sort`) that sort the elements in a list by using the `swap` function in (c)(i). (5 marks)

Example: Sept 2015 FE, Q2(b)

- Consider the following MATLAB code. Write down the value for `find([2,15,30,79],4)`, after the code is executed and explain what does the function `find` do?

```
1  function v = find(A, Len)
2  v = 0;
3  for Index = 2:Len
4      if A(Index) < A(Index-1)
5          v = A(Index-1);
6          break
7      elseif A(Index) > A(Index-1)
8          v = A(Index);
9          break
10     end
11 end
```

- Translate the MATLAB code above to Python code.

Container: Set

A set is similar to list but does not allow duplicate elements and is normally constructed from some list: `set([1,2,1,3])` or `{1,2,1,3}`. It has some computer operations and some set theory operations: `add`, `update`, `discard`, `pop`, `clear`, `copy`, `difference`, `intersection`, `isdisjoint`, `issubset`, `issuperset`, `symmetric_difference`, `union`.

It works 'like' finite set in maths.

Container: Set (cont)

A Python set is useful in data processing when we deal with data table. For example, one lecturer is in charge of test 1 and another lecture is in charge of test 2, then we have a possible scenario as follows.

```
lecturer1 = ['Programme', 'Name', 'Id', 'Test1Q1',
             'Test1Q2', 'Test1Q3']
lecturer2 = ['Id', 'Name', 'Test2Q1', 'Test2Q2a',
             'Test2Q2b']
# The common columns are
set(lecturer1).intersection(lecturer2)
# The different columns are
set(lecturer1).symmetric_difference(lecturer2)
```

Container: Dictionary

A **dictionary**, AKA, “associated array”, “hash” or “hash map” in other programming languages. In Python, it is constructed using curly brackets `{'a' : 123, 'bb' : 467}` (or `dict()` for programming construct.)

Application:

- Key-value pair: {key : value}, mydict[”word”] = 1
- Basic idea for NoSQL database
- Count items, e.g. mydict[”word”] += 1
- Construct data frame
- Configuration: E.g. `np.printoptions(*args, **kwargs)`

Outline

Advanced Programming Statements

Scripting and Debugging:

- Script: A sequence of statements in one or many Python programming text file
- Debug: Find what is wrong in script(s).

Recursive function:

- A function which calls itself.

Object-oriented programming statements:

- Defining classes, inheritance, object initialisation, ...
- try ... except ... finally ...
- with statement

Scripts

- Writing functions into a script
- Import functions from a script
- Regarding __main__

Example: Oct 2018 FE, Q2(a)

The area of a triangle ABC can be calculated by the Heron's formula

$$|ABC| = \sqrt{s(s-a)(s-b)(s-c)}, \quad s = \frac{a+b+c}{2}$$

when the lengths of the three sides, $a = BC$, $b = AC$, $c = AB$ of the triangle ABC are given. The cosine rules of the triangle ABC are given below

$$a^2 = b^2 + c^2 - 2bc\cos A,$$
$$b^2 = a^2 + c^2 - 2ac\cos B,$$
$$c^2 = a^2 + b^2 - 2ab\cos C.$$

Example: Oct 2018 FE, Q2(a) cont

Given a triangle PQR with lengths $PQ = 4.5$cm, $PR = 3.5$cm and $QR = 7$cm. Write a **program script** to find and **print** the area of the triangle PQR and all the three angles P, Q and R in **degree**. (10 marks)

Debugging

- print!
- Logging:
- type(), dir(), help()

Recursion

An alternative to using the looping statements is the use of *recursion*, which stands for “the act of a function calling itself to do things” as illustrated in the following examples. This is a useful technique for handling mathematical functions but can be expensive in memory usage and speed.

Example: Sept 2015 FE, Q5(a)

Famous Fibonacci sequence as a past year exam question:

1. Find four errors in the following function, and identify each as syntactic error or an algorithmic error. Then correct the error in the code.

```
function fib(N)
%    fib(N) - return the Nth Fibonacci number.
switch N
    case {1}
        Out = 1
    else
        Out = feval(@fib,N-1) + feval(@fib, N-1);
end
```

2. Rewrite the MATLAB function in part (i) as Python function.

Simple Example

Consider a variation of the recurrence relation found in 2017 West Australia Applications Exam (`https://senior-secondary.scsa.wa.edu.au/__data/assets/pdf_file/0006/458997/Mathematics_Applications_Calc_Free_2017.PDF`):

$$T_{n+1} = 4T_n - 3, \quad T_1 = 2.$$

Implement this recurrence relation as a Python function `T(n)`. In your implementation, return `None` if `n` is less than 1. **Demonstrate** the workings of the function by writing a Python script to display `T(1)`, `T(2)`, $\cdots$, `T(10)`. Write down the value of `T(4)`.

Final Exam Sept 2015, Q2(d)

The Taylor series of a cosine function at $x = 0$ is

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \cdots .$$

Write a recursive function that computes and approximation of cosine.

Sample Answer:

```
def mycos(x,n=8):
    from math import factorial
    if n==0: return 1.0
    else: return mycos(x,n-1) + (-1)**n*x**(2*n)/factorial(2*n)
```

Others: https://stackoverflow.com/questions/27234243/recursive-algorithm-for-cos-taylor-series-expansion-c

Final Exam Sept 2015, Q2(d) cont

More complicated answer is based on:

$$\cos x = 1 - \frac{x^2}{2!}\left[1 - \frac{x^2}{4\times 3}\left[1 - \frac{x^2}{6\times 5}\left[1 - \cdots\right]\right]\right].$$

Therefore

```
def mycos(x,n=16):
    def cos_aux(n0, x):
        if n0 > n: return 1.0
        return 1-x*x/(2*n0)/(2*n0-1) * cos_aux(n0+1, x)
    return cos_aux(1,x)
```

Example: Sierpinski Triangle

```
1  # https://stackoverflow.com/questions/25772750/sierpinski-triangle-recurs
2  import turtle
3  def draw_sierpinski(length,depth):
4      if depth==0:
5          for i in range(0,3):
6              t.fd(length); t.left(120) # draw triangle
7      else:
8          draw_sierpinski(length/2,depth-1)
9          t.fd(length/2)
0          draw_sierpinski(length/2,depth-1)
1          t.bk(length/2)
2          t.left(60)
3          t.fd(length/2)
4          t.right(60)
5          draw_sierpinski(length/2,depth-1)
6          t.left(60)
7          t.bk(length/2)
8          t.right(60)
9  window = turtle.Screen()
0  t = turtle.Turtle()
1  draw_sierpinski(100,2)
2  window.exitonclick()
```