COMP1002

Computational Thinking and Problem Solving

Lecture 2
Number representation, function and procedure, and pseudo-code

Lecture 2

- Number Representation
  - Number System
  - Number Conversion
  - Computer Memory
  - Information Representation
- Function & Procedure
- Pseudo-code

Number System

- How do we count?

  1 2 3 4 5 6 7 8 9 10 11 12…

- The Decimal System

  $123 = 1 \times 100 + 2 \times 10 + 3 \times 1$

  $= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

- Each digit represents a value as determined by its position
  - Unit for last digit, ten for second last digit, hundred for third last digit and so on (starting from the right)

Number System

- Computer counts from 1 to 2
  - Precisely, from 0 to 1
- Binary System

  $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

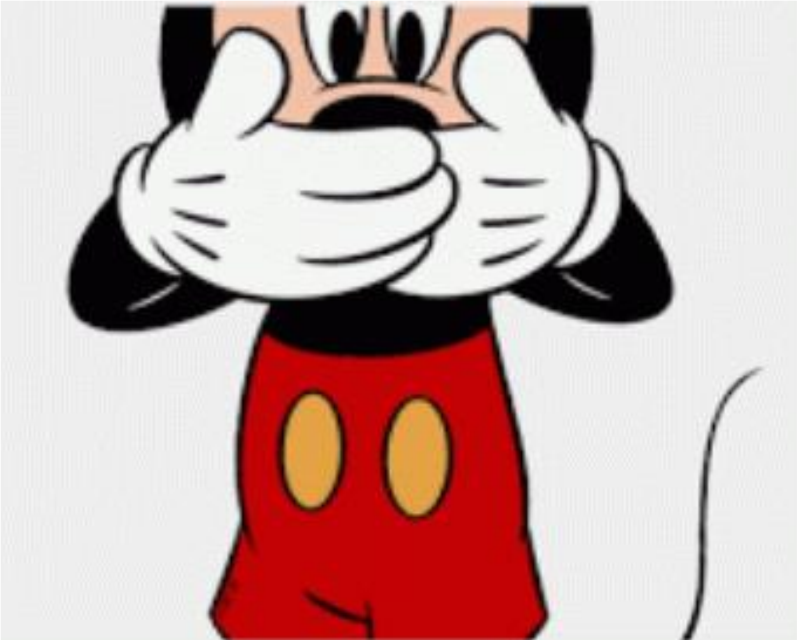
  $= 8 + 0 + 2 + 1 = 11$
- Each binary digit (bit) represents a value as determined by its position

Exercise

- What are the decimal values of the followings?
  - 10101
  - 110110
  - 10100101
  - 111010000000110
- Try NOT using a calculator to make sure you really understand AND do quickly

Number System

› How about Mickey Mouse?



Number Representation

- Doing the reverse, the decimal number 11 is represented as 1011 in binary system (understood by computers)
- We know that 11 in decimal is represented as 1011,
  - how do we represent decimal 19? And decimal 50?

Number Conversion

- Binary to Decimal for 1011

  $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

  $= 8 + 0 + 2 + 1 = 11$

- Decimal to Binary for 11

  $11 = 8 + 3 \ (\because \ 8 < 11 < 16)$

  $= 8 + 2 + 1$

  $= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1$

  $= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

- Try converting 50 to a binary number

Number Conversion

- The division approach to find the binary
- 50 ÷ 2 = 25 … The reminder is 0
- 25 ÷ 2 = 12 … The reminder is 1
- 12 ÷ 2 = 6 … The reminder is 0
- 6 ÷ 2 = 3 … The reminder is 0
- 3 ÷ 2 = 1 … The reminder is 1
- 1 ÷ 2 = 0 … The reminder is 1

- So, it is 110010

Alternative Systems

- Octal
  - Mickey's system
- Example, if we have 29702_{10}
  - 29702 ÷ 8 = 3712 … 6
  - 3712 ÷ 8 = 464 … 0
  - 464 ÷ 8 = 58 … 0
  - 58 ÷ 8 = 7 … 2

Alternative Systems

- Hexadecimal
  - Commonly used in computers (for better readability for human)
  - Can treat each digit as 4 bits
  - Example
    - 29702 ÷ 16 = 1856 … 6
    - 1856 ÷ 16 = 116 … 0
    - 116 ÷ 16 = 7 … 4
- How to represent a hex digit which is > 9?

  A=10, B=11, C=12, D=13, E=14, F=15
- Binary Value of 29702: 111010000000110

Home Exercises

- Convert the following decimal into binary, octal, hexadecimal:
  - 51966
  - 64206
  - 712173
  - 14600926

- What do you find?

Computer Memory

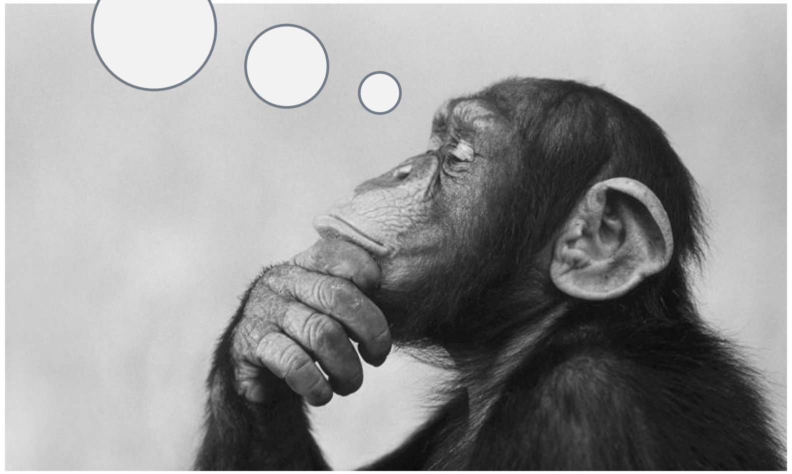
- All data need to be stored before being manipulated by a computer
- Data are stored in computer memory
  - Memory is physical, so it is not unlimited
  - There must be a size for memory storage
  - How large is the memory size?
    - Memory of size 256 MB, 1 GB, 4 GB
  - Normally 8 bits is used to represent a data unit, which is called a *byte*

Computer Memory

- How large a number can a byte represent?
  - 8 bits: smallest is 00000000_2 and largest is 11111111_2
  - Between 0 to 255
- Byte is the unit of storage and memory structure
  - Usually, 2 bytes or 4 bytes are grouped together as a storage unit, called a *word*
  - A *word* could be 16 bits or 32 bits, or even 64 bits (often called a long word for 32/64 bits)
- How large can 16 bits represent? 32 bits?
  - 0 to 65535 vs 0 to 4294967295

Number Representation

Since a number is represented by a sequence of bits, how large is a number in a program?



Number Representation

› How many decimal digits for 123456789 * 987654321?

› Can it be represented by a 32-bit number?

› Let us try it in Python

```
>>> 123456789 * 987654321
121932631112635269
```

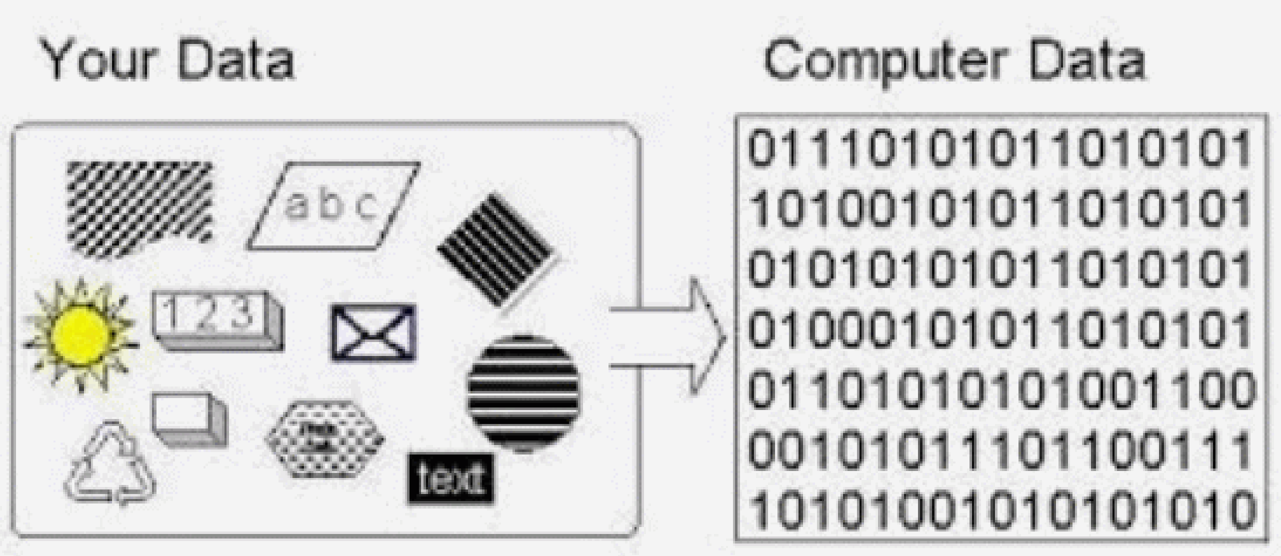
› Verify by multiplying yourself

Number Representation

- Let us try yet bigger numbers
  123456789 * 123456789 * 123456789
- Python gives you 1881676371789154860897069 (25 digits)
- Technically unlimited size for integers in Python
  - They can be represented and computed as long as memory is available, but that could be slow

Information Representation

- Note that computers only understand 0 and 1
  - Operations in a computer require the representation of all information as binary numbers, or sequence of bits.
- How do we represent these?
  - Negative numbers
  - Real numbers
  - Text
  - Image
  - Audio
  - Video



Information Representation

- Negative numbers could have +/- sign represented as 1/0, as the first sign bit
  - Sign-and-magnitude representation
    - a leading sign bit 0 indicates positive and 1 indicates negative
  - Two's complement representation
    - split the range 00000000 to 11111111 into two consecutive halves, 0bbbbbbb for positive and 1bbbbbbb for negative, and 00000000 for zero linking the two

Information Representation

- Decimals can be represented in fixed point representation
  - Extend binary representation with the “decimal point” separating the integer part and fractional part
  - For example, $101.11_2 = 5.75_{10}$
- Real numbers can be represented in floating point representation
  - The number to be represented is based on the 3 components – sign, exponent and mantissa
  - Further reading: The IEEE 754 standard

Information Representation

- Text is represented as a sequence of characters
  - Each character is normally a byte
    - It can represent 0 to 255, i.e., 256 characters
  - Use a standard representation for each English character, digit, punctuation symbol, and some others.
  - The most common standard is called ASCII (American Standard Code for Information Interchange).
    - It represents 128 characters using 7 bits
  - Other standards also exist

ASCII code table

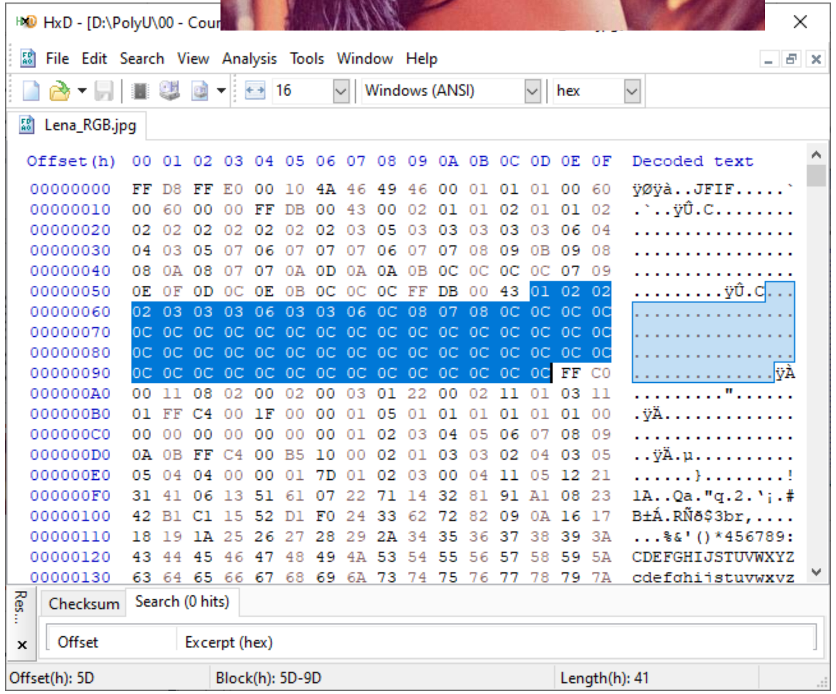
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Properties of ASCII code

- Space (32) comes before all the printable characters
- Numbers (48 – 57) come before (less than) letters
- Uppercase letters (65 – 90) always come before than lowercase letter (97 – 122)

Information Representation

› Multimedia data like images, audios, videos are represented according to some standard encoding methods
  – Common format: GIF, PNG, JPEG, MP3, MP4



What is a Function?

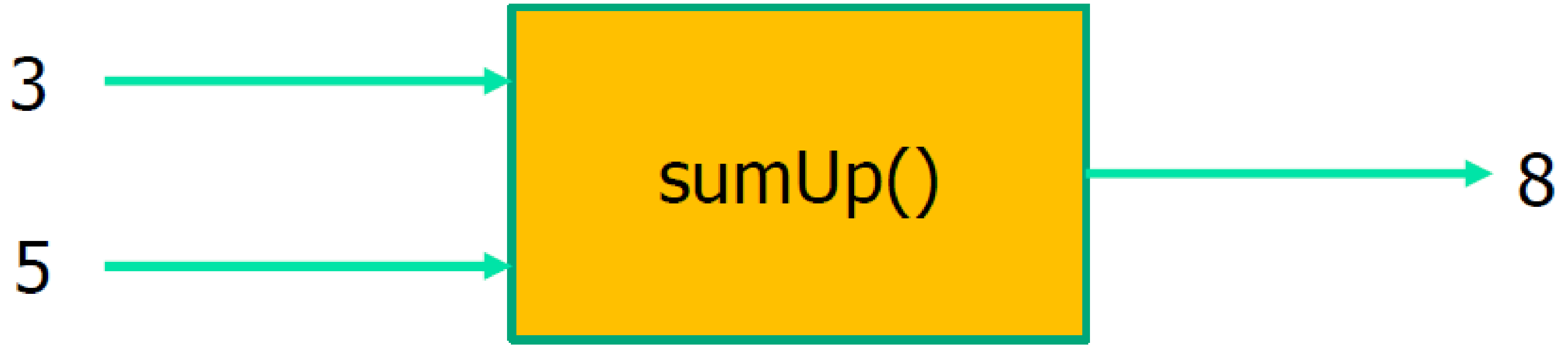
- In mathematics

  $f(x) = 2x^4 + 3x^3 - 6x^2 + 5x - 8$

- In programming

  ```
  answer = sumUp(firstnumber, secondnumber)
  ```

- A function performs a certain task usually with some inputs and provides the result as an output

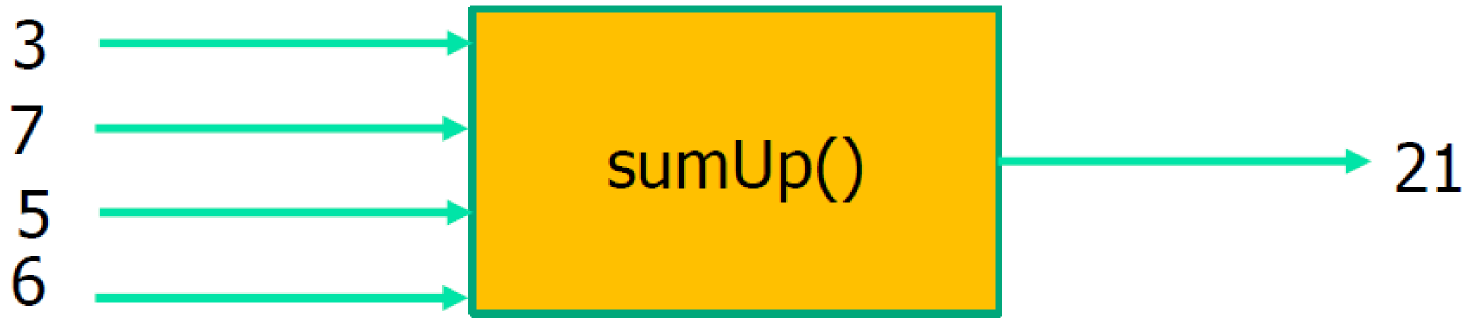


What is a Function?

› The inputs and outputs of a function must be specified clearly
  – Example: `sumUp(x, y)`
  – Input: x and y are integers
  – Output: the sum of x and y

› Usually we do not need to know how it is implemented (we just use it)
  – Treat it like a black box

› Maybe more inputs, generally only one output



Procedure

- A function returning no value is often called a procedure
  - A black box returning nothing
  - Is there any use if there is no output?
- Besides output, a program is designed to perform computation, as reflected by changes in memory content, or external storage like file/database
  - The collection of memory content and the current position of program execution is called the program (or system) state
  - A procedure will lead to a change in the state
  - Example: a procedure may sort a list of numbers stored in the memory, without giving any output. The list printed out before and after executing this procedure will be different

Pseudo-code

- Pseudo means false
- Pseudo-code (or pseudocode) refers to high level program code being expressed in English-like style
- They reflect lines of thought, or logical steps
- They can be of higher level than computer program statements

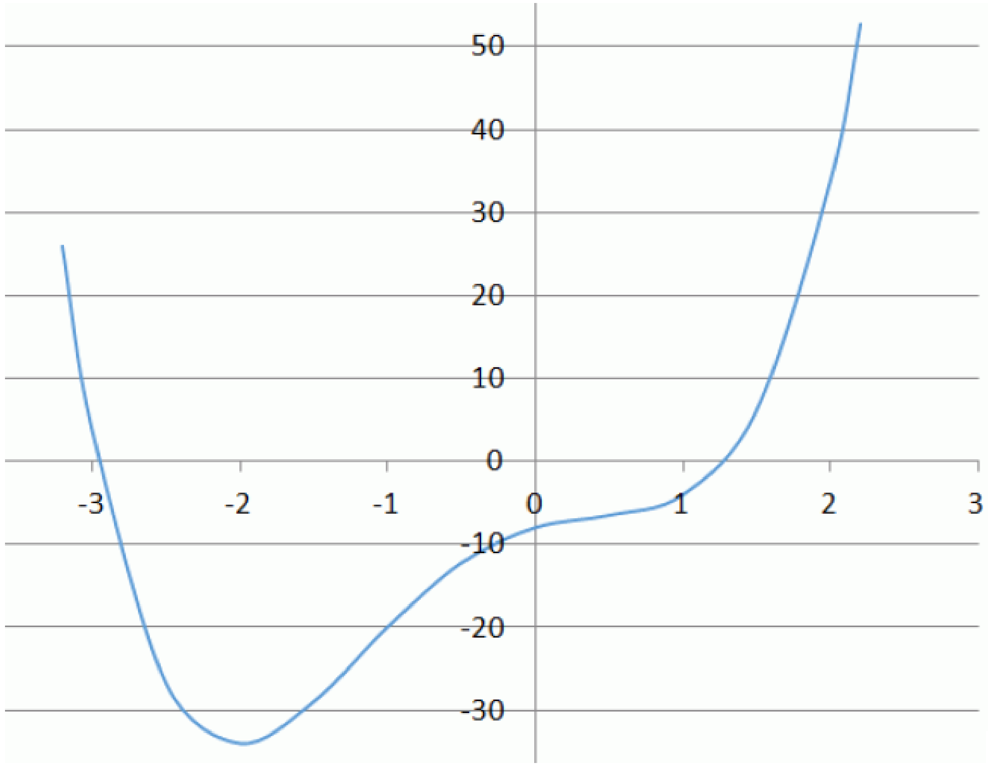
Pseudo-code

Repeat
    Generate first digit 0 to 9 using a 10-faced die
    Generate second digit 0 to 9 using a 10-faced die
    If number already exists, try again
    If number does not exist, output to screen or file
Until all 100 numbers are generated

Pseudo-code

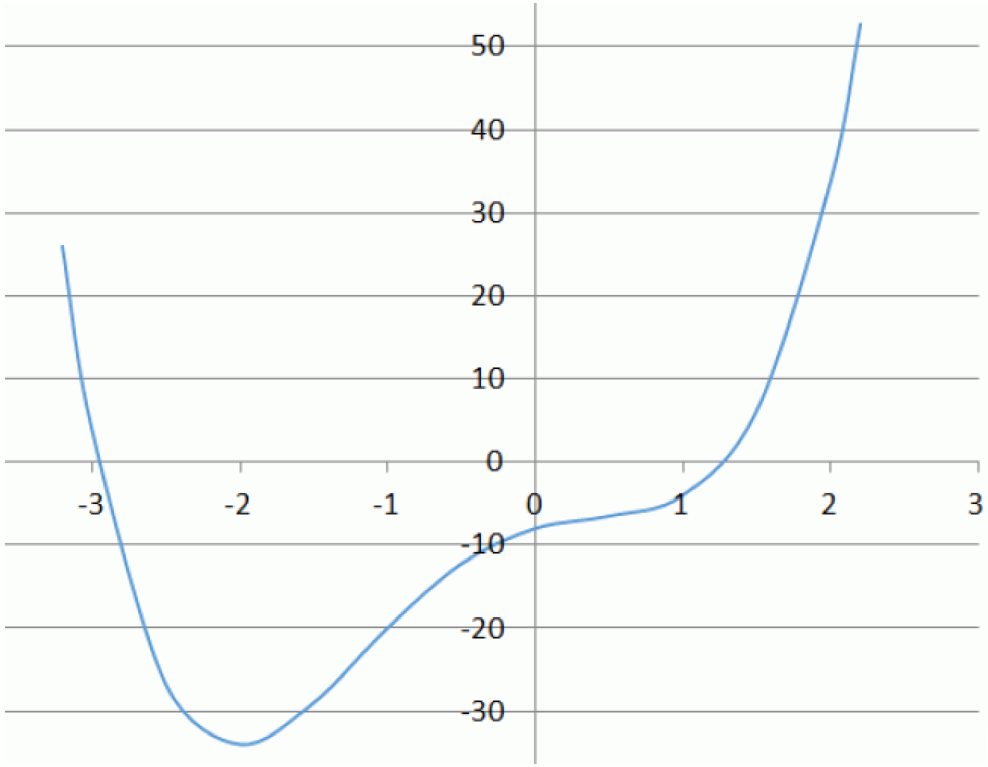
› The following illustrates how pseudo-code is used in problem solving

› Solving $f(x) = 2x^4+3x^3-6x^2+5x-8 = 0$



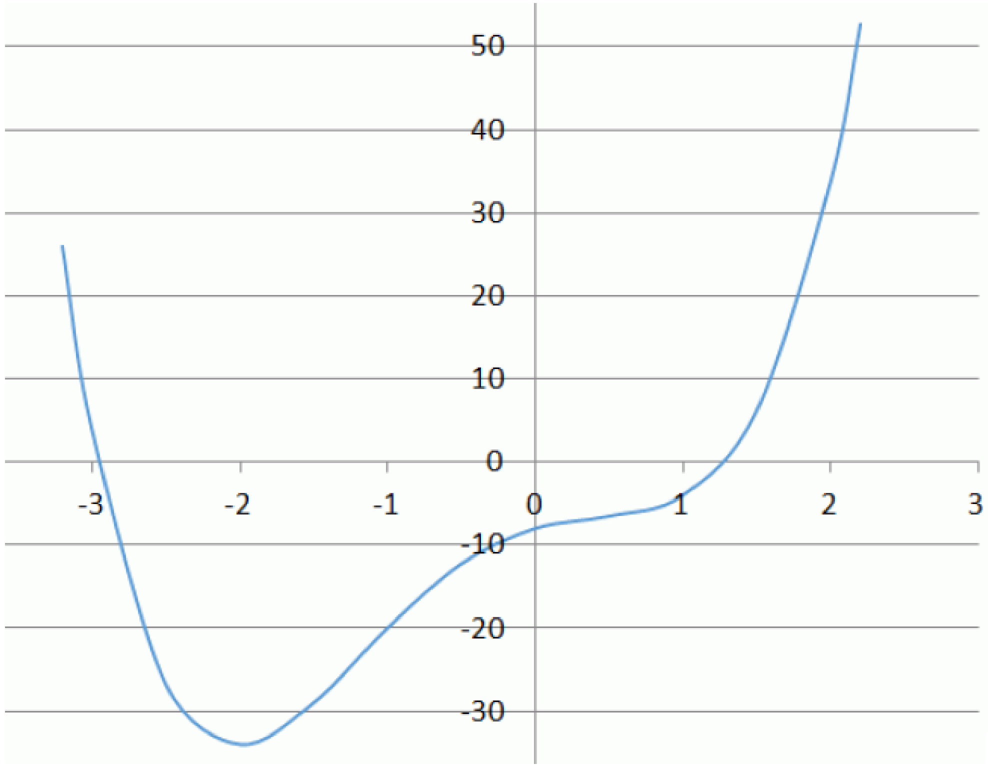
Pseudo-code

- It is obvious that a root exists between -3 and -2 and another between 1 and 2
- We may plot the graph with high accuracy and measure out the values of the roots
- We may try to compute $f(x_1)$ by trying various x_1 between -3 and -2 until $f(x_1)$ is almost zero (similar for x_2 between 1 and 2 so that $f(x_2)$ is almost zero)



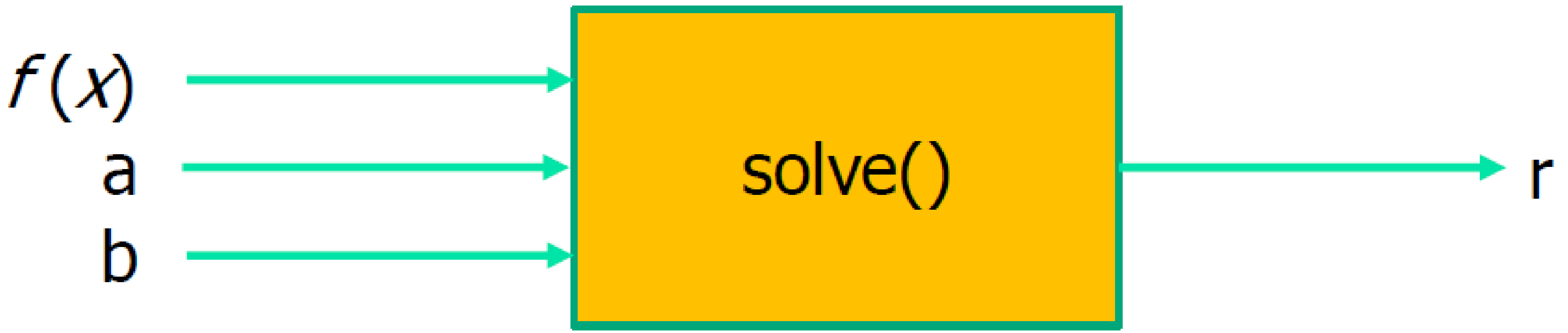
Pseudo-code

- Theorem
  - For a function f(x), which is continuous between a and b and f(a) and f(b) are of opposite sign, then there exists a root between a and b.
  - We will just “search” for this root between a and b



Pseudo-code

- To solve this problem, we first define the input and output
  - Input: mathematical function f(x), a and b
  - Output: a root r between a and b such that f(r) is technically zero



Pseudo-code

› Then we develop the pseudo-code with a step-by-step approach:

```
check that f(a) and f(b) are of different sign
if they are of same sign
    return "cannot proceed"
set left = a, right = b          # root between left and right
repeat
    get middle point mid = (left + right) / 2
    if f(mid) is technically zero, return mid as root
    if f(left) and f(mid) are of opposite sign then
        set right = mid # root between left and mid
    else
        set left = mid # root between mid and right
until root is found
```

Pseudo-code

- Critical thinking
  - Is the program correct?
    - If not, where is the problem?
  - Why do we want to say technically zero, rather than equal to zero?
    - How can we implement this “technically zero” checking?
  - Can we ask the program to find its own range, a and b, instead of being provided as input?

Pseudo-code

- Another example
  - Consider a simple task to add up a list of numbers L, where L = [1, 5, 8, 2, 7, 9]
    - Input: a list of numbers, L
    - Output: the sum S of those numbers in L
  - Possible pseudo-code:

```
set S = 0
for each number n in list L
      add n to S
return the sum S
```

Pseudo-code

- Exercise
  - What if we want the multiplication of the numbers instead?

Pseudo-code

› Yet anther example
  – Consider a task to look up the meaning of "computation" in a dictionary
    › Input: a term to look up, t
    › Output: explanation E of the term t in dictionary
  – Possible pseudo-code:

```
set p = 1
repeat
    turn to page p and look for t
    if t is found in page p, return corresponding explanation E
    if t is not found, increment p by 1
until t is found
```

Pseudo-code

› A better solution, since it is a dictionary

› Improved pseudo-code:

```
set start = 1 and end = last page
repeat
set mid = (start + end) / 2
    turn to page mid and look for t
    if t is found in page mid, return corresponding explanation E
    if t is before first word in page mid then
        set end = mid            # first part
    else
        set start = mid          # second part
until t is found or end <= start
if t is not found return “not found”
```

Pseudo-code

- Some considerations
  - "look for t"
    - Are you looking up t in page p or page mid from top to bottom?
    - You would need to give more details about this step if you are taking the start-from-middle checking approach
  - Pseudo-code could be of different levels of details
    - Normally we start from higher level and gradually refining towards lower level
    - Program code is at lowest level of details
    - Dumb computers could execute program code at the lowest level to achieve the designated goal

Summary

› Number Representation
  – Number System
  – Number Conversion
  – Computer Memory
  – Information Representation

› Function & Procedure

› Pseudo-code