CEng 240 – Spring 2021
Week 3

Sinan Kalkan

Dive into Python [Part 1]

Disclaimer: Figures without reference are from either from "Introduction to programming concepts with case studies in Python" or "Programming with Python for Engineers", which are both co-authored by me.

Previously on CENG240!

What does 'algorithm' mean?

- "A procedure or formula for solving a problem"
- "A set of instructions to be followed to solve a problem"
- "an effective method expressed as a finite list of well-defined instructions for calculating a function"
- "step-by-step procedure for calculations"

Previously on CENG240!

Describing algorithms

Option 1: Use pseudo-code descriptions.

Algorithm. Calculate the average of numbers provided by the user.

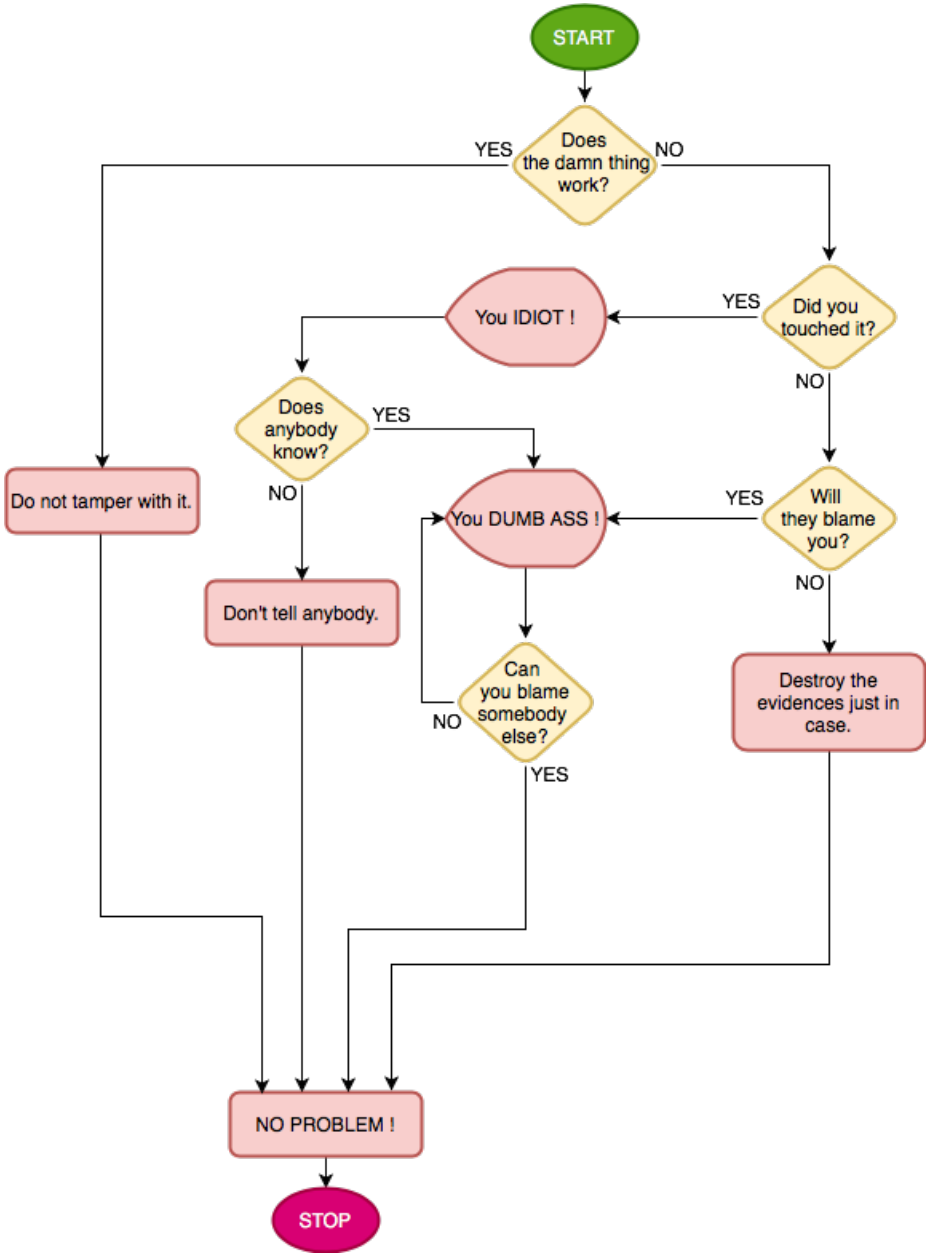
Input: N -- the count of numbers
Output: The average of N numbers to be provided

Step 1: Get how many numbers will be provided and store that in N
Step 2: Create a variable named Result with initial value 0
Step 3: Execute the following step N times:
Step 4: Get the next number and add it to Result
Step 5: Divide Result by N to obtain the average

Describing algorithms

Previously on CENG240!

Option 2: Use flow-charts.



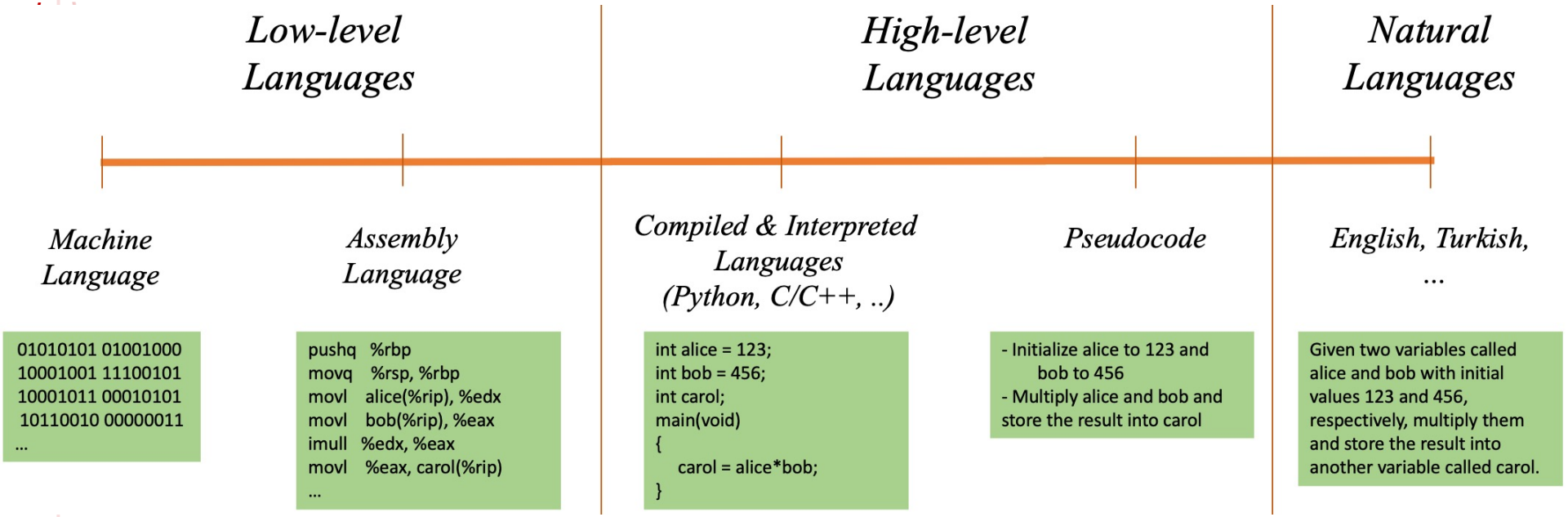
Previously on CENG240!

Comparing Algorithms

- Rougly count the main number of steps in terms of n, the 'size' of the problem.
- Example: Guess my number!
  - Random guessing
  - Sweeping from beginning
  - Middle guessing

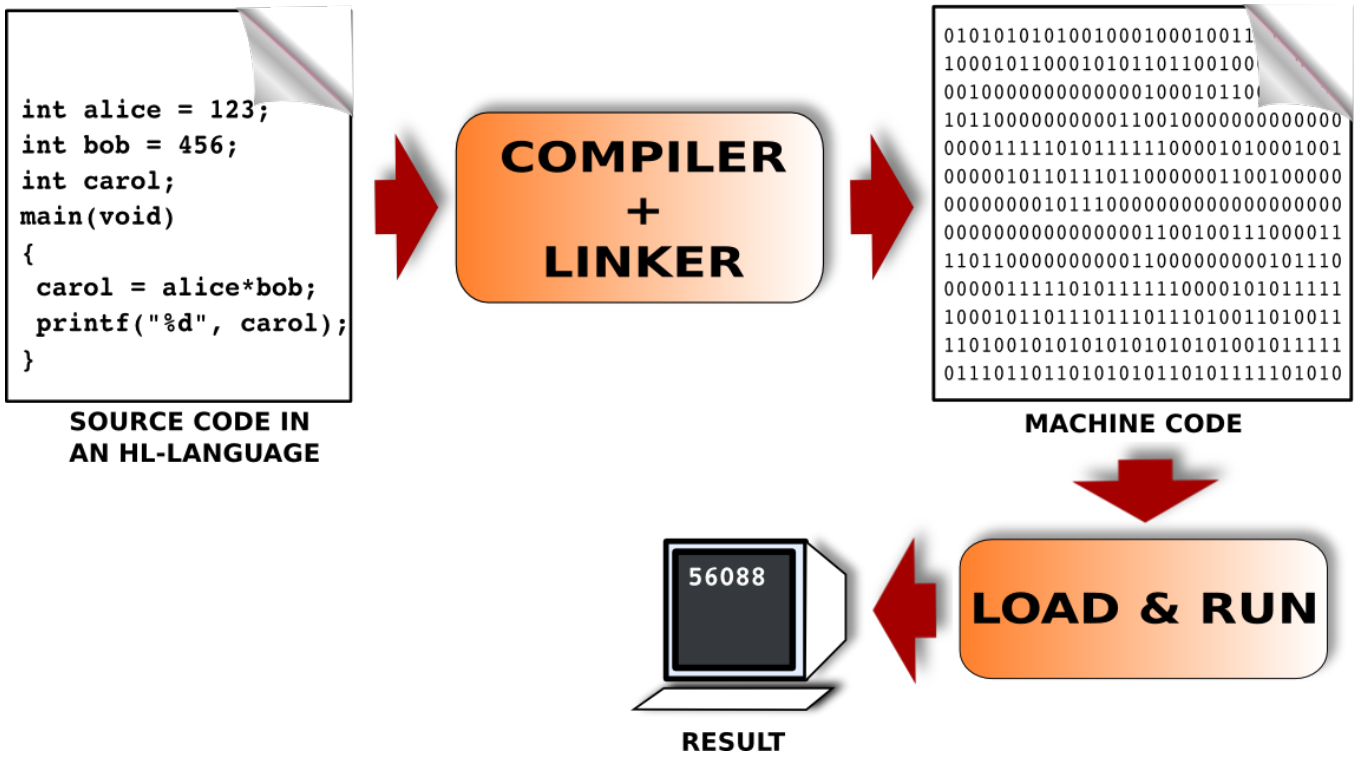
The World of Programming Languages

Previously on CENG240!



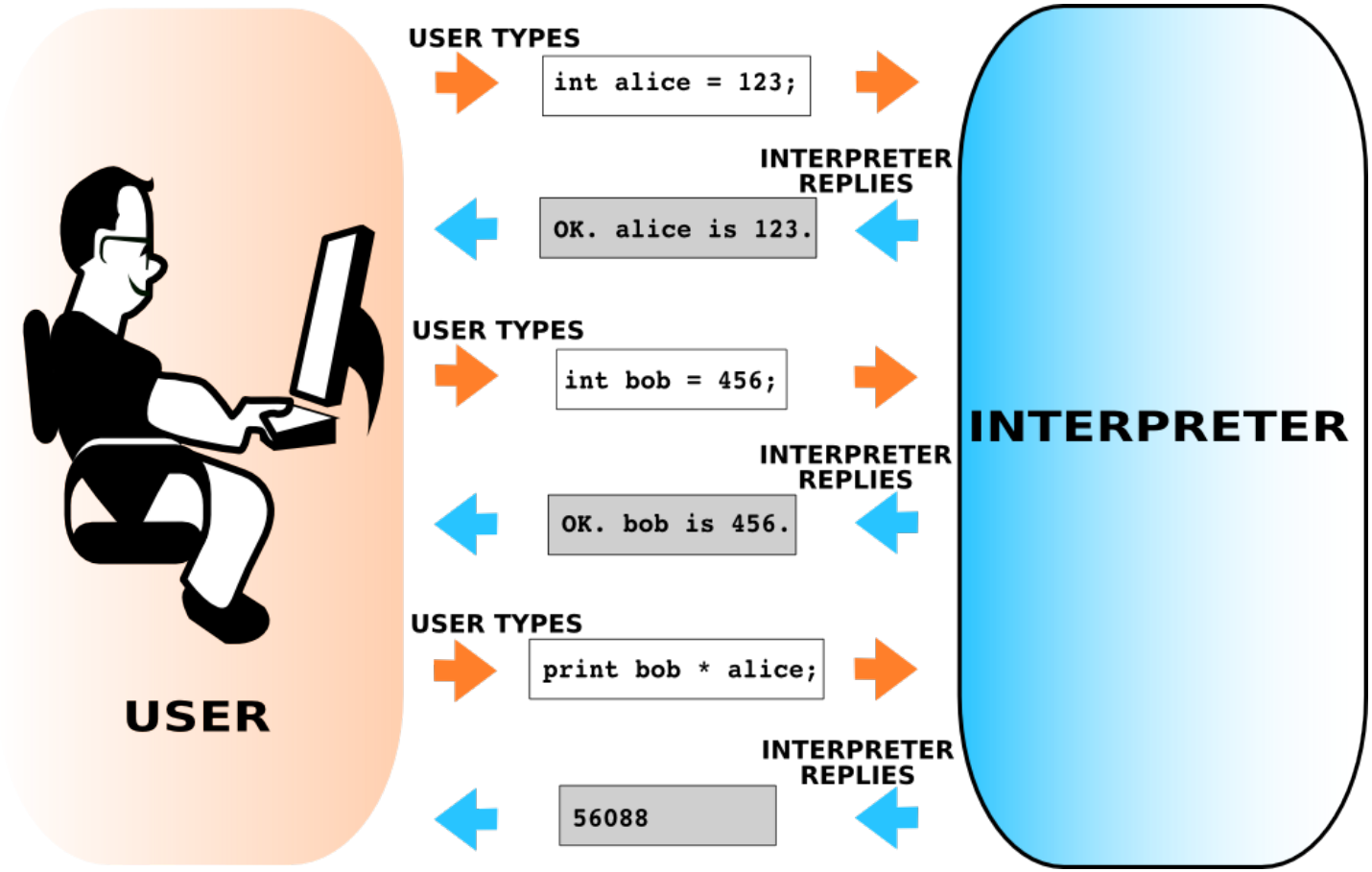
Previously on CENG240!

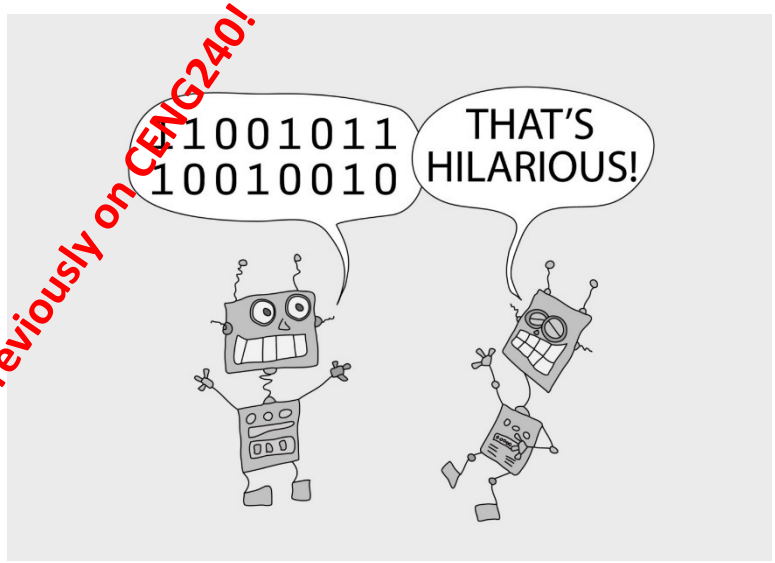
Interpreter vs. Compiler

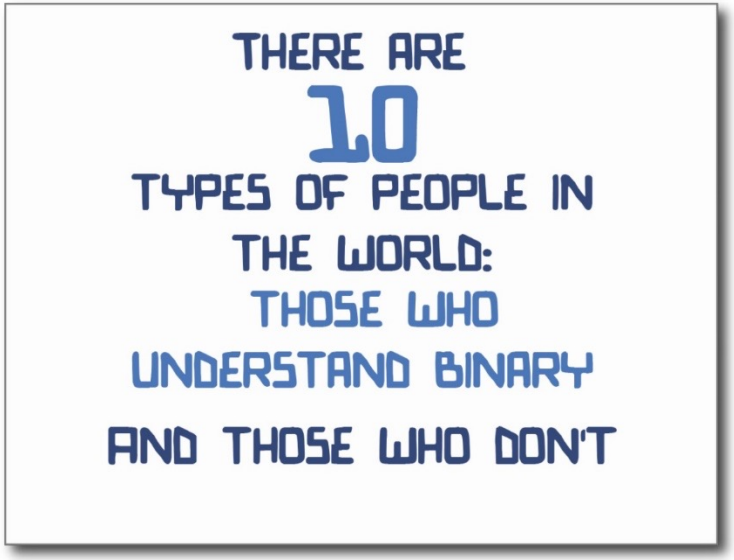


Previously on CENG240!

Interpreter vs. Compiler







Two's complement representation of integers, IEEE floating-point representation, Information loss with Floating Points, representation of characters, text and Boolean.

REPRESENTATION OF DATA IN COMPUTERS (CH3)

Binary Representation of Numeric Information (continued)

Previously on CENG 240!

- Two's complement instead of sign-magnitude representation
  - Positive numbers have a leading 0.
    - 5 => 0101
  - The representation for negative numbers is found by subtracting the absolute value from 2^N for an N-bit system:
    - -5 => $2^4 - 5 = 16 - 5 = (11)_{10} => (1011)_2$
- Advantages:
  - 0 has a single representation: +0 = 0000, -0 = 0000
  - Arithmetic works fine without checking the sign bit:
    - 1011 (-5) + 0110 (6) = 0001 (1)
    - 1011 (-5) + 0011 (3) = 1110 (-2)

Previously on CENG240!

Binary Representation of Real Numbers

Conversion of the digits after the dot into binary:

- 1st Way:
  - 0.375 → 0x½ + 1x¼ + 1x1/8 → 011
- 2nd Way:

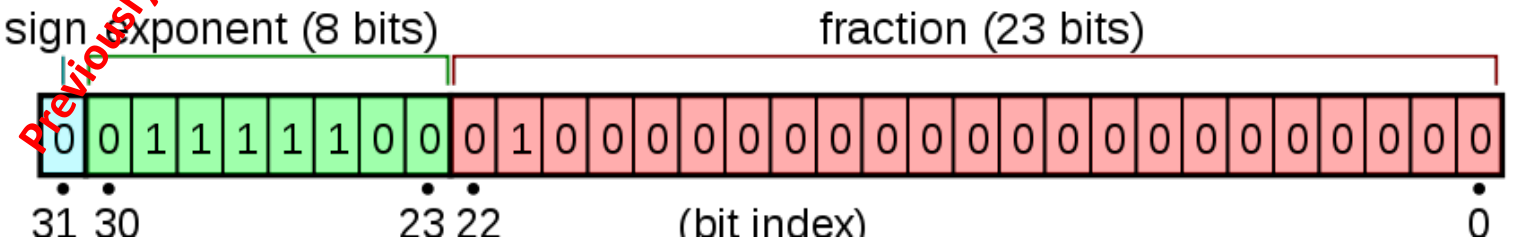
	Fraction		Multiplier		Whole		Fraction
Step 1	0.375	×	2	=	0	.	75
Step 2	0.75	×	2	=	1	.	5
Step 3	0.5	×	2	=	1	.	0

The result:

.	0	1	1

Continue until fraction is zero

IEEE 32bit Floating-Point Number Representation



$$= (-1)^{\text{sign}}(1.b_{-1}b_{-2}...b_{-23})_2 \times 2^{e-127}$$

- M x 2^E $(2-2^{-23})\times 2^{127}$
- Exponent (E): 8 bits
  - Add 127 to the exponent value before storing it
  - **E can be 0 to 255 with 127 representing the real zero.**
- Fraction (M - Mantissa): 23 bits
- $2^{128} = 1.70141183 \times 10^{38}$

IEEE 32bit Floating-Point Number Representation

- Now consider 4.1:
  - 4 => $(100)_2$
  - 0.1 =>
    - x 2 = 0.2 = 0 + 0.2
    - x 2 = 0.4 = 0 + 0.4
    - x 2 = 0.8 = 0 + 0.8
    - x 2 = 1.6 = 1 + 0.6
    - x 2 = 1.2 = 1 + 0.2
    - x 2 = 0.4 = 0 + 0.4
    - x 2 = 0.8 = 0 + 0.8
    - …….

- So,
  - Representing a fraction which is a multiple of $1/2^n$ is lossless.
  - Representing a fraction which is not a multiple of $1/2^n$ leads to precision loss.

Representing Real Numbers:

Information Loss

Previously on CENG240!

```
>>> 0.9375 - 0.9
0.037499999999999998
```

```
>>> 2000.0041 - 2000.0871
-0.0829999999998563

>>> 2.0041 - 2.0871
-0.08299999999999974
```

```
>>> sin(PI)
1.2246467991473532e-16
>>> cos(PI)
-1.0
```

```
>>> A = 1234.567
>>> B = 45.67834
>>> C = 0.0004
>>> AB = A + B
>>> BC = B + C
>>> print (AB+C)
1280.2457399999998

>>> print (A+BC)
1280.2457400000001
```

Previously on CENG240!

Binary Representation of Textual Information (cont'd)

ASCII
7 bits long

Decimal	Binary	Val.
48	00110000	0
49	00110001	1
50	00110010	2
51	00110011	3
52	00110100	4
53	00110101	5
54	00110110	6
55	00110111	7
56	00111000	8
57	00111001	9
58	00111010	:
59	00111011	;
60	00111100	<
61	00111101	=
62	00111110	>
63	00111111	?
64	01000000	@
65	01000001	A
66	01000010	B

Hex.	Unicode	Charac.
0x30	0x0030	0
0x31	0x0031	1
0x32	0x0032	2
0x33	0x0033	3
0x34	0x0034	4
0x35	0x0035	5
0x36	0x0036	6
0x37	0x0037	7
0x38	0x0038	8
0x39	0x0039	9
0x3A	0x003A	:
0x3B	0x003B	;
0x3C	0x003C	<
0x3D	0x003D	=
0x3E	0x003E	>
0x3F	0x003F	?
0x40	0x0040	@
0x41	0x0041	A
0x42	0x0042	B

Unicode
16 bits long

Partial listings only!

This Week

- Dive into Python [Part 1/2]
  - Basic and Container Data in Python:
    - int, float, complex, bool
    - string, list, tuple, dict, set
  - Operators and Expressions: Arithmetic operators, Expression, Comparison operators, Logic connectives

Administrative Notes

- Quiz 2 announced!
- Labs starting next week.
  - Demo this weekend
- Midterm: 1 June, Tuesday, 17:40

int, float, complex, bool

string, list, tuple, dict, set

BASIC AND CONTAINER DATA

What is data?

- **Data**: Information to be processed to solve a problem.
- Identify the data for the following example problems:
  - Find all wheat growing areas in a terrestrial satellite image.
  - Given the homework, lab and examination grades of a class, calculate the letter grades.
  - Alter the amplitude of a sound recording for various frequencies.
  - Extrapolate China's population for the year 2040 based on the change in the population growth rate up to this time.
  - Compute the launch date and the trajectory for a space probe so that it will pass by the outermost planets in the closest proximity.
  - Compute the layout of the internals of a CPU so that the total wiring distance is minimized.
  - Find the cheapest flight plan from A to B, for given intervals for arrival and departure dates.
  - Simulate a war between two land forces, given (i) the attack and the defense plans, (ii) the inventories and (iii) other attributes of both forces.

What is data?

- CPU can only understand two types of data:
  - Integers,
  - Floating points.
- The following are not directly understandable by a CPU:
  - Characters (‘a’, ‘A’, ‘2’, …)
  - Strings (“apple”, “banana”, …)
  - Complex Numbers
  - Matrices
  - Vectors
- But, programming languages can implement these data types.

Basic Data in Python
Numerical Types

- Integers:
  - `int`
  - Unlimited size
- Floating point numbers:
  - `float`
  - IEEE754 standard (32bit, 64bit)
- Complex numbers
  - `complex`
  - `3+4j`

Basic Data in Python

Numerical Types

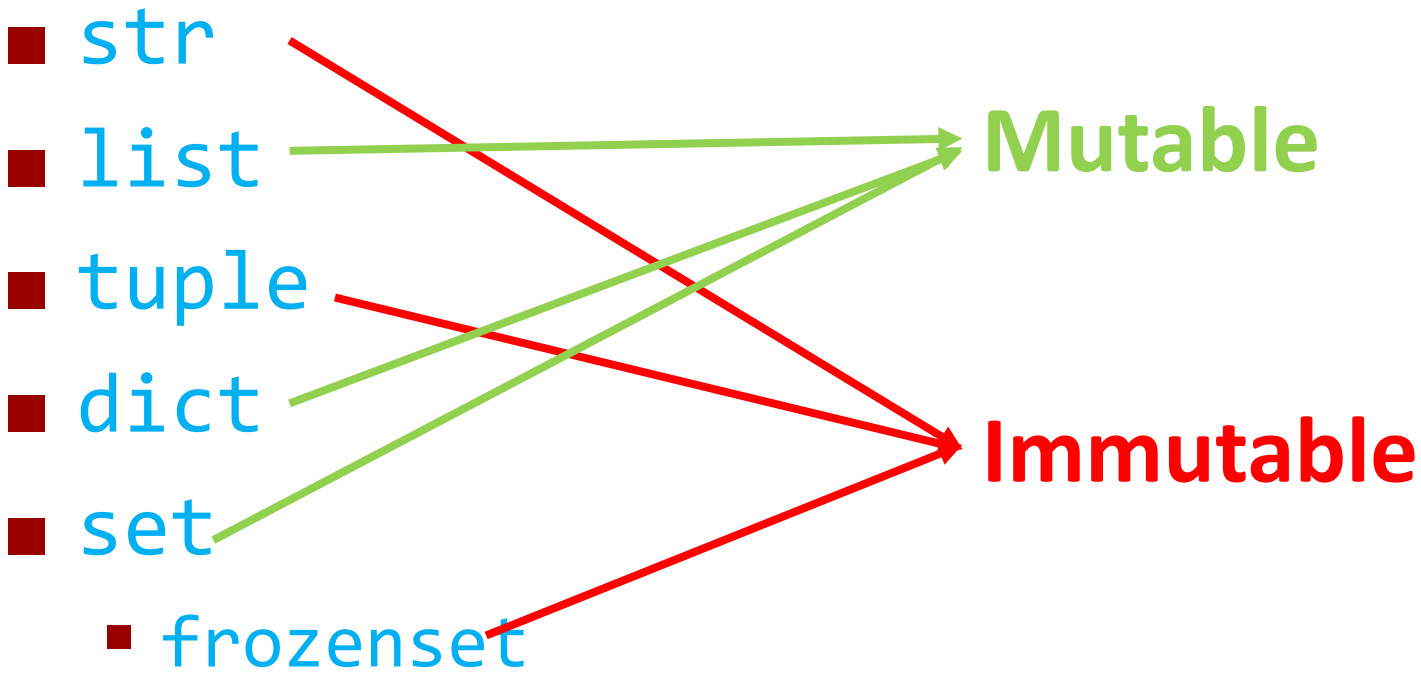
- Useful operations with numerical types
  - `type(<data>)` function
  - `abs(<number>)`
  - `pow(<number1>, <number2>)`
  - `round(<float-number>)`
  - `sin(), cos(), log()` from `math` library

Basic Data in Python

Boolean Type

- `bool` type
  - Can take `True` or `False`
- Useful operations with `bool` type
  - `and, or, not`

Container Data in Python

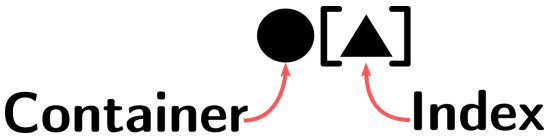


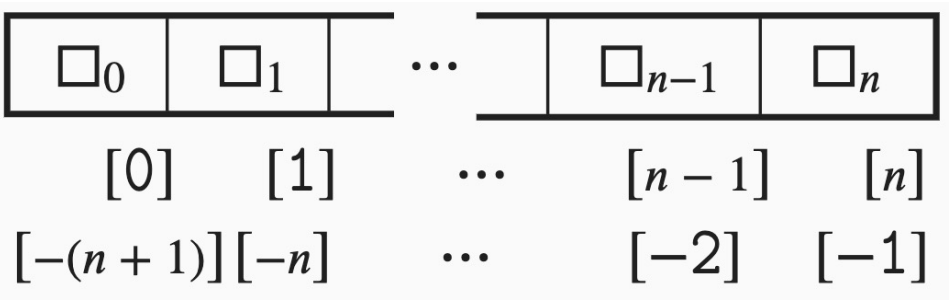
- Mutability vs. immutability

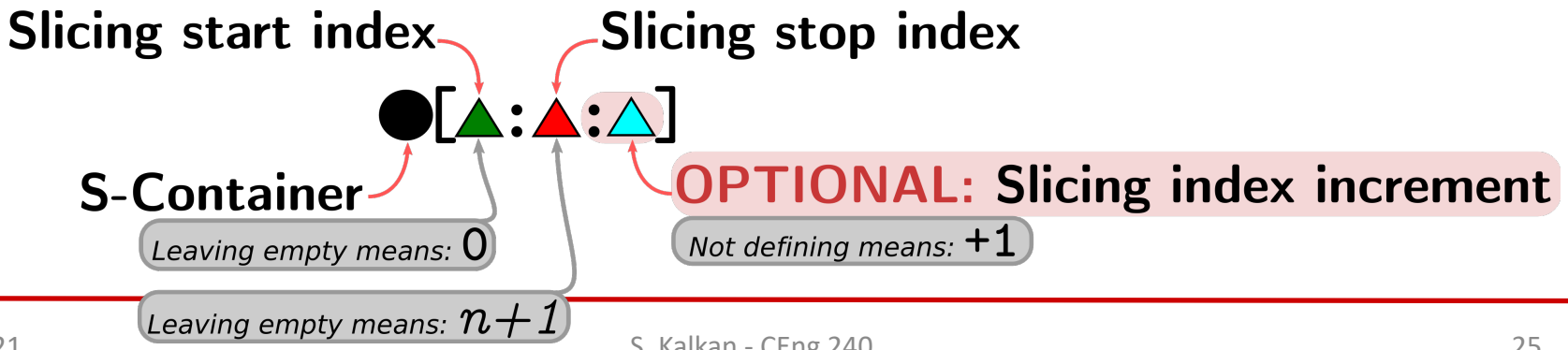
Container Data in Python

Accessing Elements of Sequences

- Positive indexing
- Negative indexing
- Slicing







Container Data in Python

Useful Operations

- len()

```
>>> len("Five")
4
```

- Concatenation (+)

```
>>> "Hell" + "o"
'Hello'
```

- Repetition

```
>>> "Yes No " * 3
'Yes No Yes No Yes No '
```

- Membership

```
<item> in <Container>
```

or

```
<item> not in <Container>
```

Container Data in Python

String

- Writing strings
  - single quote
  - double quote
  - triple quote
- Special characters
- Unicode support in v3

Container Data in Python

String

- Examples with strings

- Strings are immutable

Container Data in Python

String: Useful Operations

- `str()`
- `len()`
- Concatenation, repetition, membership
- Evaluate a string: `eval()` function

- Deletion from / insertion into a string
  - Not possible

Container Data in Python

List and Tuple

- Lists: mutable
  - `[10, 20, 30]`
  - `["ali", 20, "veli", 30]`
  - `[10, [20, [30]], 40]`
- Tuples: immutable
  - `(10, 20, 30)`
  - `("ali", 20, "veli", 30)`
  - `(10, (20, (30)), 40)`

Container Data in Python

List: Useful Operations

- Deletion

```
>>> L = [111,222,333,444,555,666]
>>> L[1:5] = []
>>> print(L)
[111, 666]
```

```
>>> L = [111,222,333,444,555,666]
>>> del L[1:5]
>>> print(L)
[111, 666]
```

Container Data in Python

List: Useful Operations

- Insertion

```
>>> L = [111,222,333,444,555,666]
>>> L[2:2] = [888,999]
>>> print(L)
[111, 222, 888, 999, 333, 444, 555, 666]
```

```
>>> L = [111,222,333,444,555]
>>> L.append(666)
>>> print(L)
[111, 222, 333, 444, 555, 666]
>>> L.extend([777, 888])
[111, 222, 333, 444, 555, 666, 777, 888]
```

```
>>> L = [111,222,333,444,555,666]
>>> L.insert(2, 999)
>>> print(L)
[111, 222, 999, 333, 444, 555, 666]
```

$\bullet.f(\square)$ has the conceptual meaning of $f(\bullet, \square)$

Container Data in Python

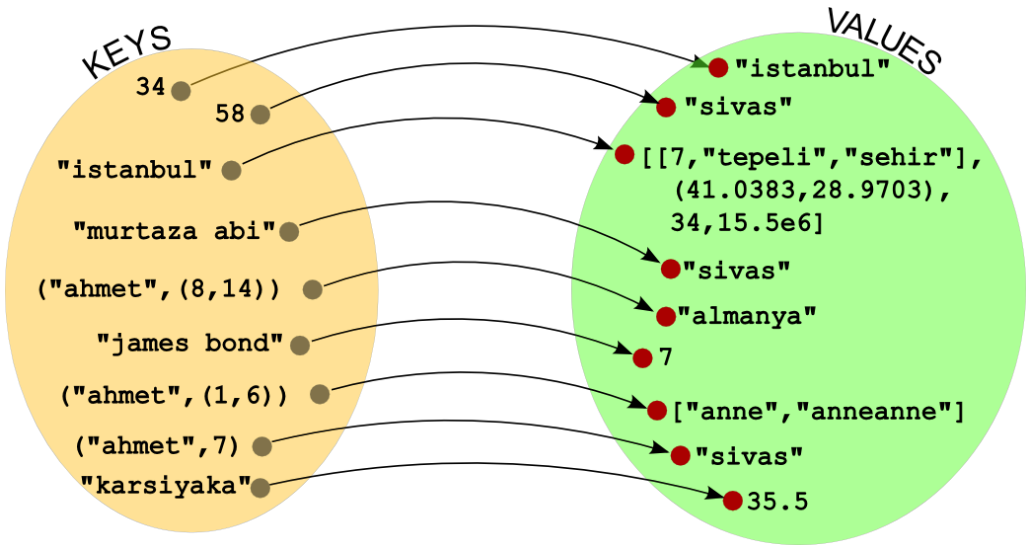
List: Useful Operations

- `list()` and `tuple()` functions
- Concatenation, repetition
- Membership

Container Data in Python

Dictionary

- `dict`: mutable
- Add/delete elements
- KeyError



Container Data in Python

Dictionary: Useful Operations

- `len()`
- Membership
- `values()`
- `keys()`

Container Data in Python

Set

- `set`: mutable

```
a = {1,2,3,4}
b = {4,3,4,1,2,1,1,1}
print (a == b)
a.add(9)
a.remove(1)
print(a)
```

Container Data in Python

Frozen Set

- `frozenset`: immutable

```
>>> s = frozenset({1, 2, 3})
>>> print(s)
frozenset({1, 2, 3})
```

Container Data in Python

Set: Useful Operations

- `len(<set>)`
- Membership
- Set operations:
  - Subset: `S1 <= S2`
  - Superset: `S1 >= S2`
  - Union: `S1 | S2`
  - Intersection: `S1 & S2`
  - Set difference: `S1-S2`

- Only with the '`set`' type:
  - `S.add(element)`
  - `S.remove(element)`
  - `S.pop()`

Operators and Expressions

ACTION

Action

- Purposes of actions
  - Creating/modifying data
  - Interaction with the environment
- Types of actions
  - Expressions
  - Statements

Expressions

- Expression: A mathematical operation that has a resulting value
  - `a = 10 + 20 * 3`
- Operators can be unary or binary

Expression Evaluation

Precedence and Associativity

Operator	Precedence	Associativity
`[]`	1.	Left-to-right
`**`	2.	Right-to-left
`*, /, //, %`	3.	Left-to-right
`+, -`	4.	Left-to-right
`<, <=, >, >=, ==, !=, in, not in`	5.	Special
`not`	6.	Unary
`and`	7.	Left-to-right (with short-cut)
`or`	8.	Left-to-right (with short-cut)

Expression Evaluation
and, or evaluation

- A shortcut is taken if possible

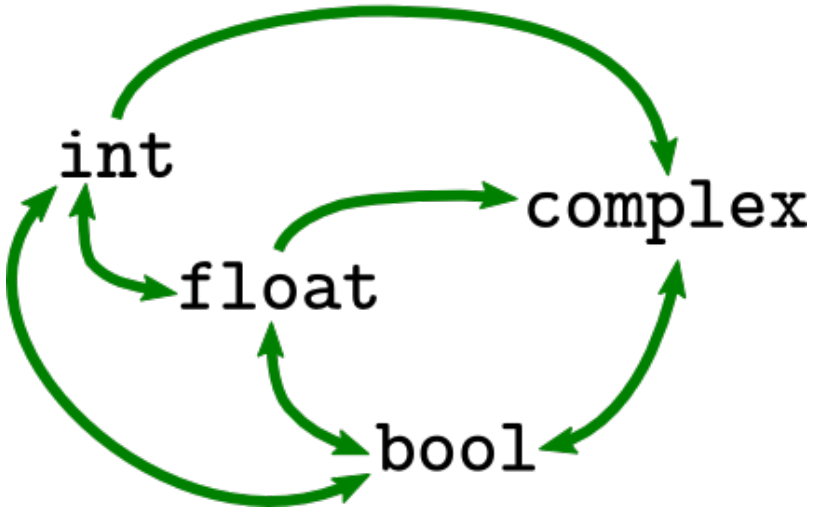
$$\square_1 \text{ and } \square_2 \text{ and } \square_3 \text{ and } \cdots \text{ and } \square_n$$

$$\square_1 \text{ or } \square_2 \text{ or } \square_3 \text{ or } \cdots \text{ or } \square_n$$

Expression Evaluation

Type Conversion

- Implicit
  - `3 + 4.5`
  - `3 + True`
- Explicit
  - `int(), float(), bool()`
  - `str(), ...`



Statements

- Basic statements
  - `del L[3]`
  - `a = 20`
  - `pass, del, return, yield, raise, break, continue, import, future, global, nonlocal.`
- Compound statements
  - Conditional statement
  - Repetition statements

Statements: Assignment

- Simple assignment
  - `a = 10`
- Multiple assignment
  - `a = b = c = 10`
  - `a, b = 10, 20`
- Compound assignment
  - `a += 10`
  - `a *= 20`
- Swapping values
  - `a, b = b, a`

`id()` function

Operators in Python

- Arithmetic operators
- Logic operators
- Container operators
- Comparison operators

Table 4.3.1 Arithmetic, Logic, Container and Comparison operators in Python.

Operator	Operation	Result Type
`[]`	Indexing	Any data type
`**`	Exponentiation	Numeric
`*`	Multiplication or Repetition	Numeric or container
`/`	Division	Numeric (floating point)
`//`	Integer Division	Numeric (integer)
`+`	Addition or concatenation	Numeric or container
`-`	Subtraction	Numeric
`<`	Less than	Boolean
`<=`	Less than or equal to	Boolean
`>`	Greater than	Boolean
`>=`	Greater than or equal to	Boolean
`==`	is equal to	Boolean
`!=`	is not equal to	Boolean
`in`	is a member	Boolean
`not in`	is not a member	Boolean
`not`	logical negation	Boolean
`and`	logical and	Boolean
`or`	logical or	Boolean

Final Words: Important Concepts

- Data
  - Basic Data Types
  - Container Types
  - Accessing elements of a container type (indexing, negative indexing, slicing).
  - Mutable vs. immutable types
- Actions
  - Expressions, statements
  - Expression evaluation: Operators, precedence, associativity
  - Basic statements
  - Assignment

THAT'S ALL FOLKS!
STAY HEALTHY