EGR 1330 Computational Thinking with Data Science

Computational Thinking and Programming Principles

Outline

- Concepts of Computational thinking
- Variables, data types in Python programming
- Operators and precedence in Python programming

Objective

- Understand the concept of computational thinking
- Use of Python variables, data types, and some operators in programming.

Computational thinking

Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer. (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016).

CT vs Programming

- Programming: Write code in a specific programming language to address a certain problem.
- Computational thinking: It is not just about programming. It is an approach to solve problems using concepts and ideas from computer science.

Computational Thinking Approach

- Breaking down a problem into smaller parts
- Looking for patterns in the problems
- Figuring out what information is needed
- Developing a step-by-step solution

Pillars of Computational Thinking

1. Decomposition

2. Pattern Recognition

4. Algorithms

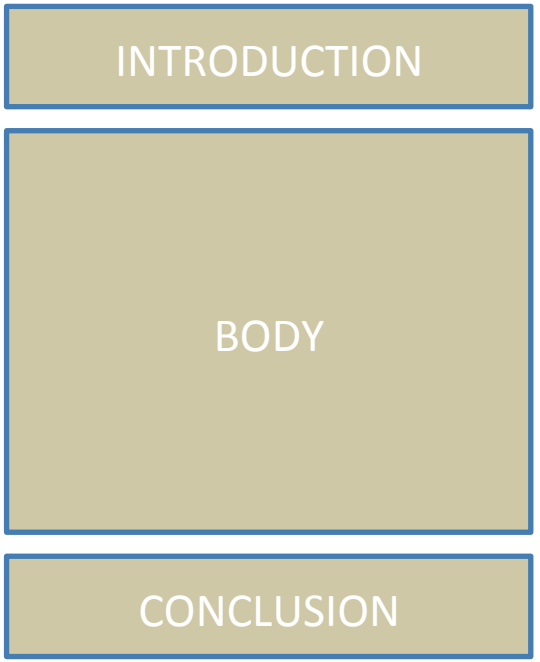
3. Abstraction

Decomposition

Decomposition is the process of taking a complex problem and breaking it into more manageable sub-problems.

Examples:

- ✓ Writing a paper:
  - Introduction
  - Body
  - conclusion.
- ✓ Wide-viewed (Panorama) image:
  - Taking multiple overlapped photos
  - Stitch them

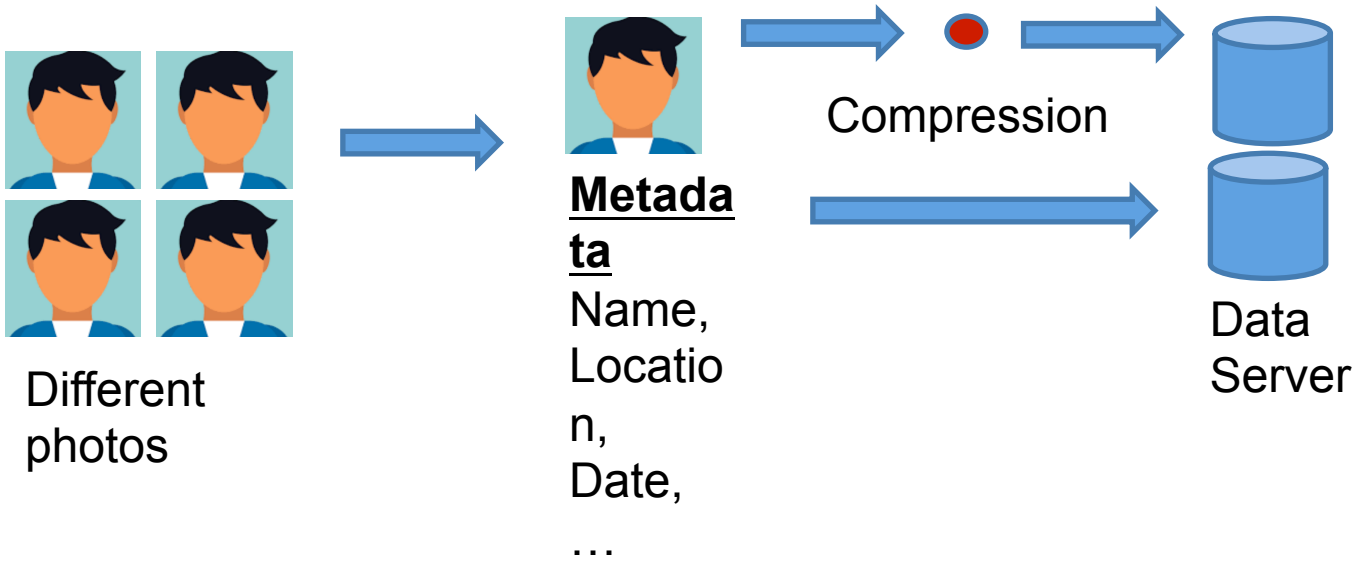


Pattern Recognition

- Find similarities, or shared characteristics of the problems
- Complex problem becomes easier to solve. Use same solution each occurrence of the pattern.

Example:

- Upload an album of photos to Facebook: same pattern.



Data Representation and Abstraction

- Determine important characteristics of the problem and filter out ones that are not important.
- Use these characteristics to create a representation of what we are trying to solve.

Students in a University

Important	NOT important
Name	Favorite color
Billing address	Food preferences
Phone number	Shoe size
Student Id	….
…	

Books in an online bookstore

Important	NOT important
title	Cover color
ISBN	Author's hometown
authors	Complete content
category	….
…	

Algorithms

- Step-by-step instructions of how to solve a problem
- Identifies what is to be done, and the order in which they should be done

Making a cup of tea

1. Fill electric kettle
2. Boil it
3. Pour hot water in a cup
4. Put teabag in the cup
5. Steep for 3 minutes
6. Remove the teabag

Case Study: Scheduling a meeting

Decomposition: Two steps process

- ✓ Schedule an earlier meeting request
- ✓ Remove conflicting request with the one just scheduled.

Pattern:

- ✓ Look at unscheduled requests and pick the best one

Abstraction:

- ✓ Each request is represented with proposed start time, end time, and student name for the meeting.

Algorithm:

- ✓ Meeting request at earliest time
- ✓ Schedule it
- ✓ Look at other requests one by one and remove it if overlapped with the scheduled ones.

Program: It is the realization of an algorithm using a syntax that the computer can understand.

Algorithm: is to manipulate the data

Data: is stored in memory.

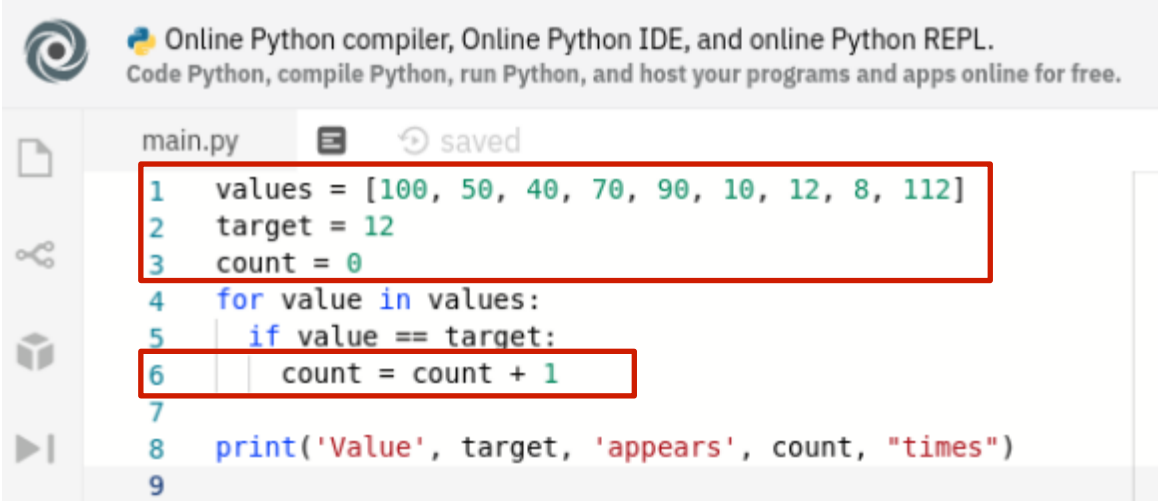
Program accesses the data in memory using its address.

Variable is used to have friendly name for accessing data memory address. We don't need to know where exactly the memory address is.

Variables

- A meaningful name for a piece of data stored in memory
- Value can change during execution of the program

- **Define** with a name and initial value
- **Assign** new value using equal sign ("=")



Variables

- Should be meaningful
- May consist of letters, digits, and underscores but may not start with a digit or a special character.
- Should not include uppercase letters (Python convention).

Valid variables?

a) exchange_rate = 0.1

b) my_name = “John”

c) is_student = True

d) name = my_name

e) my_name = “Brian”

f) $_class = “python”

g) 1class = “programming”

Variables

- Should be meaningful
- May consist of letters, digits, and underscores but may not start with a digit or a special character.
- Should not include uppercase letters (Python convention).

Valid variables?

a) exchange_rate = 0.1

b) my_name = “John”

c) is_student = True

d) name = my_name

e) my_name = “Brian”

~~f) $_class = “python”~~

~~g) 1lass = “programming”~~

Keywords

- Reserved words: Internally defined keywords in Python
- You cannot use reserved words as variable names

True	class	return	is	finally
False	if	for	lambda	continue
None	del	from	while	nonlocal
and	def	global	not	with
as	elif	try	or	yield
break	else	import	pass	
assert	except	in	raise	

Data types

- Numeric data types:
  - Integer: Whole number that can be positive, negative or zero

    E.g.: 2045, 767, 0, -87, -435
  - Float: Floating point number that has a decimal place

    E.g.: 1433.3, 140.75, -25.187, -100.1

Data types

- String: Collection of one or more characters that are enclosed within single or double quotes

  E.g.: 'Hello World!', "Computational Thinking with Data Science"

- Boolean: Data type that takes one of the two possible values - True or False

- Advanced data types:
  - Array
  - List
  - Tuple
  - Set
  - Dictionary

Valid variables?

a) exchange_rate = 0.1

b) my_name = "John"

c) is_student = True

d) name = my_name

e) my_name = "Brian"

~~f) $_class = "python"~~

~~g) 1lass = "programming"~~

Comments for Code

- Comments: Useful information to help the readers understand the source code, i.e., helps in understanding the logic behind the Python code
- Comments: Starts with a hashtag symbol (#)

  E.g.: # This is the workshop on Computational Thinking with Data Science

```
1
2   # print from 1 to 10
3   for i in range(1, 11):
4       print("value of i=", i)
```

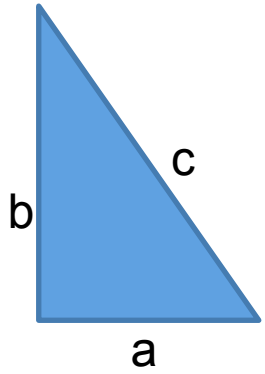
Operators on Variables

num1 = 15
num2 = 2

Operation	Operator	Result
sum = num1 + num2	+	17
diff = num1 – num2	-	13
product = num1 * num2	*	30
quotient = num1 / num2	/	7.5
integer_quotient = num1 // num2	//	7
power = num1 ** num2	**	225
modulus = num1 % num2	%	1

Precedence of Operators

Operator	Description
()	Parentheses (grouping)
**	Exponentiation
*, /, %	Multiplication, division, remainder
+, -	Addition, subtraction



c = $\sqrt{\square}a2+b2$ Which one is correct?

1) c = a*a + b*b**0.5
2) c = (a*a + b*b)**0.5
3) c = ((a*a) + (b*b))**0.5

Comparison operators

Python code	Meaning
a == b	Equal to
a != b	Not equal to
a > b	Greater than
a >= b	Greater than or equal to
a < b	Less than
a <=b	Less than or equal to

Input and Output

- Python provides two in-built functions for input and output operations
  - input( ): This function takes the input and evaluates the expression

    Python automatically detects the data type entered
  - print( ): This function prints the output

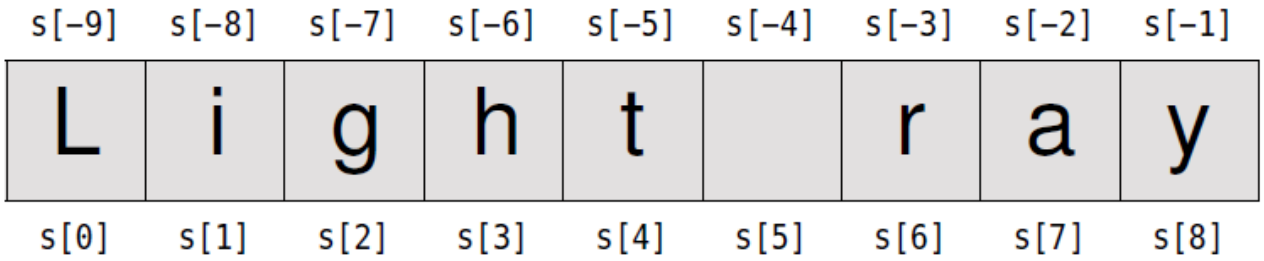
    Multiple expression can be passed with each of them separated by a comma

    Converts the expressions into a string before writing to the screen

    Note: Functions can also be written by the user (user-defined functions)

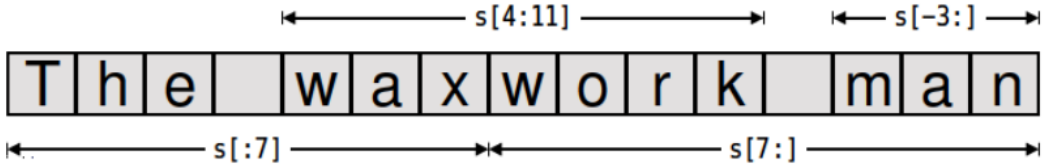
String

- string holds a sequence of characters
- string is immutable
- Indexing:



- Comparison: <, >, <=, >=, ==, !=
- Slicing:

```
seq[start]
seq[start:end]
seq[start:end:step]
```




```
>>> s = s[:12] + "wo" + s[12:]
>>> s
'The waxwork woman'
```

Examples

1) Write a program that requests the user to enter two numbers and prints the sum, product, difference and quotient of the two numbers.

2) Write a program that reads in two integers and determines and prints whether the first is a multiple of the second. (Hint: Use the modulus operator.)

3) State the order of evaluation of the operators in each of the following Python statements and show the value of **x** after each statement is performed.

```
a) x = 7 + 3 * 6 / 2 - 1
b) x = 2 % 2 + 2 * 2 - 2 / 2
c) x = ( 3 * 9 * ( 3 + ( 9 * 3 / ( 3 ) ) ) )
```

4) Add more examples for string indexing