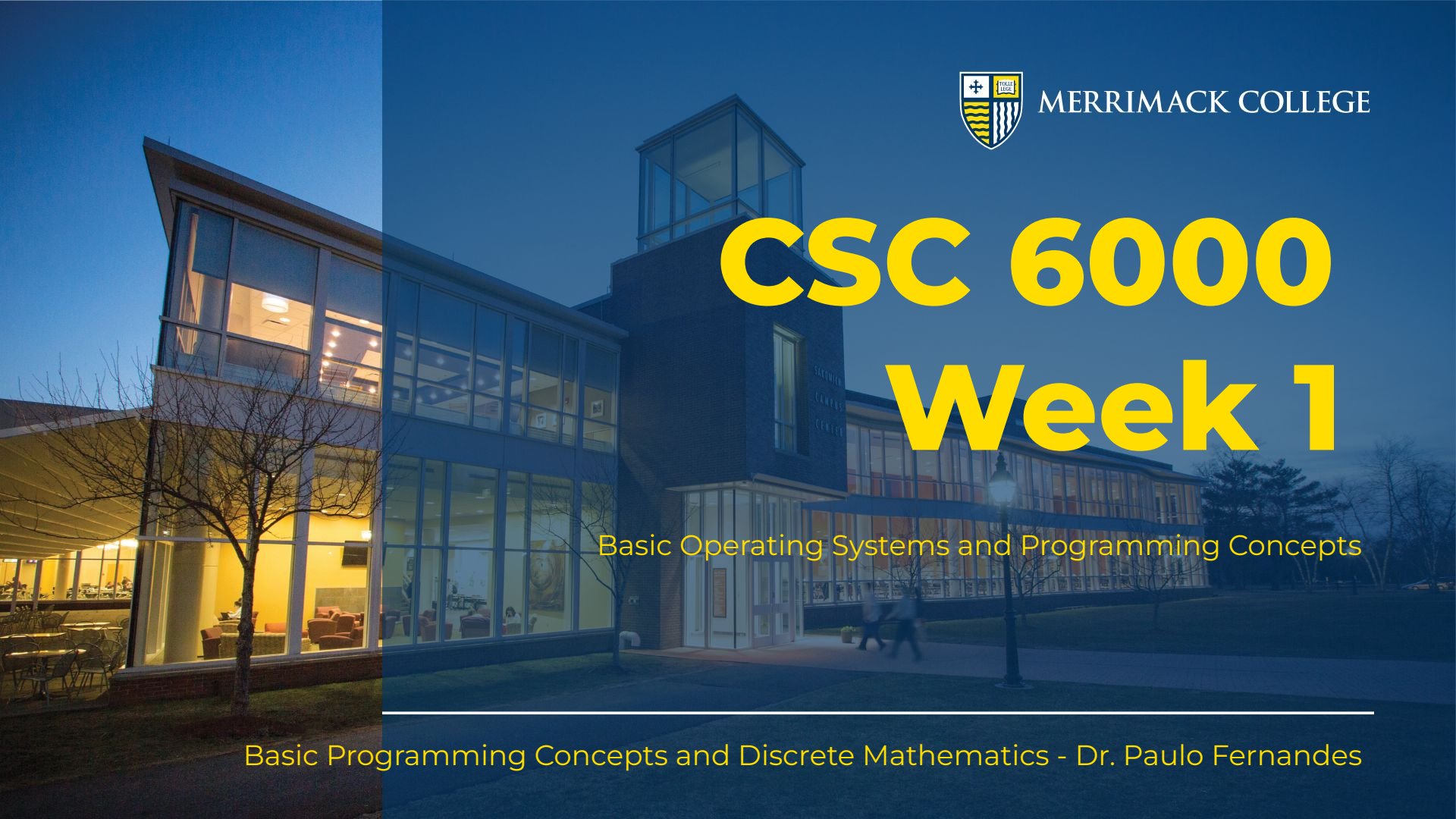


MERRIMACK COLLEGE

CSC 6000 Week 1

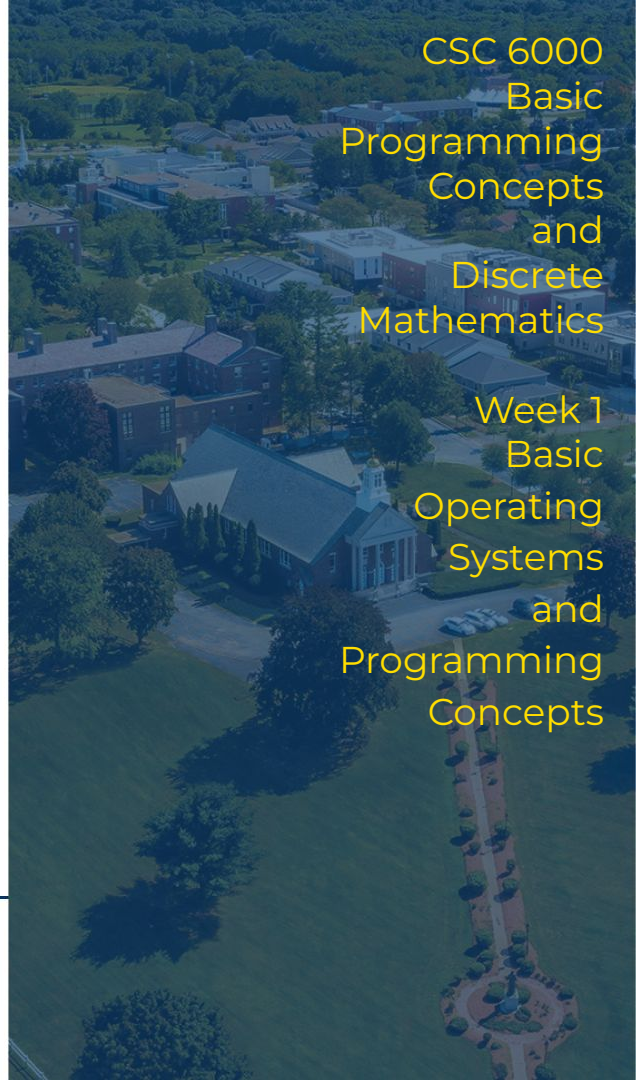
Basic Operating Systems and Programming Concepts

Basic Programming Concepts and Discrete Mathematics - Dr. Paulo Fernandes

Presentation Agenda

Week 1

- Course Introduction
  a. Course format
  b. Evaluation
  c. Timeline
- Basic OS & Programming
  a. What is a Computer?
  b. File Structure and File Formats
  c. Installing Python, Hello World!
  d. Variables, Operations, Basic I/O
- This Week's tasks



Course Introduction

This and the next seven weeks ahead

Welcome!

This course is entitled CSC 6000 Basic Programming Concepts and Discrete Mathematics.

This is the first course for the Master in CS and its goal is to get you ready to start a solid formation in Computer Science by giving you the basic knowledge on dealing with computers and programming, as well as the basic notions on discrete mathematics.

- Course format
- Timeline
- Evaluation

Course Format

Success Coach and Graduate Advising
- ecs-grad-advising@merrimack.edu

This course is fully online, all weeks will have:
- Live sessions every Thursday 6:30pm to 8:30pm
- Office Hours every next Monday 8:45pm to 9:45pm

Every week but the last, we will have:
- Discussion/In-class exercises tasks due by next Monday
- Quizzes may be taken from Saturday until next Thursday
- Coding assignments must be turned in by next Thursday

Last week has:
- Final exam must be done by Sunday

Student Tutors available!

Visit the Hub!

Send me an email if you are planning to attend office hours!

Email subjects have to start with CSC6000

Communication
- fernandesp@merrimack.edu

Course Format

- Live sessions every Thursday 6:30pm to 8:30pm
  - Meeting ID: 922 0224 6996 Passcode: CSC6000
- Office Hours every Next Monday 8:45pm to 9:45pm
  - Meeting ID: 922 0224 6996 Passcode: CSC6000
- In-class exercises tasks due by next Monday
  - Simple exercises, and/or posts on the discussion board
- Quizzes may be taken from Saturday to next Thursday
  - Ten multiple choice questions, open book, open notes, and untimed
- Coding assignments to be turned in by next Thursday
  - Programming tasks
- Final exam is due by last Sunday of classes
  - 50 questions to be done in 75 minutes, open book, open notes

Attendance is optional, but strongly recommended!

Timeline

This is a very fast-paced course (better get used to that).

Try to check out the topics before the live session.

Any changes to the schedule will be announced in class and on Canvas.

Unit	Week	Topic	Project	Quiz	Discussion	Exam
1	1	Basic operating system and programming concepts	P#1	Q#1	D#1	
2	2	Number theory and binary logarithms	P#2	Q#2	D#2	
3	3	Arithmetic and geometric progressions, summations	P#3	Q#3	D#3	
4	4	Combinatorics and permutations	P#4	Q#4	D#4	
4	5	Combinatorics and binomial coefficients	P#5	Q#5	D#5	
5	6	Linear algebra: scalars, vectors, matrices, and tensors	P#6	Q#6	D#6	
6	7	Basic probabilities and their computation	P#7	Q#7	D#7	
6	8	Basic statistics and their computation		Q#8	D#8	Final Exam (all units)

Don't let tasks pile up!

Evaluation

The evaluated tasks are weighted as such (103%):

Activity	Published grades	Percentage
Projects (7)	Evaluated from 0 to 100 each	42.0%
Quizzes (8)	Evaluated from 0 to 10 each	32.0%
Discussions (8)	Evaluated from 0 to 100 each	4.0%
Final Exam	Evaluated from 0 to 50	25.0%

All tasks are due by their stated deadline. The late penalties reduce 10% of the evaluation, plus 2% for each full day of delay.

Final deadline on Friday of the course's last week

Evaluation

Because this is a Pass/Fail course, you get a passing grade with a numeric grade equal to or superior to 70%.

For each course of the MSCS program, the scale will be:

A	A-	B+	B	B-	C+	C	C-	F
95 and up	90 to 94.9	87 to 89.9	83 to 86.9	80 to 82.9	77 to 79.9	73 to 76.9	70 to 72.9	69.9 and low

There are no grade D+, D, or D- in graduate courses. C- is not a passing grade for MSCS. An average grade of B is required to continue in the Masters of Science in Computer Science, lower than that puts you in probation.

Final grades will be published by the first Monday after the course's end.

Basic OS and Programming

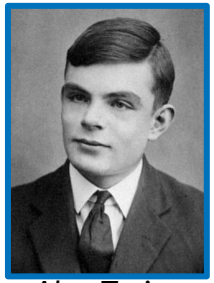
All computer systems are essentially the same, but with a different look and feel. They differ in the way they are perceived, their user interface. Yet nowadays, there is a convergence of technologies making computer systems more similar than ever before.

Computer systems are any form of device that provides digital computation.

- **What is a Computer?**
- File Structure and File Formats
- Installing Python, Hello World!
- Variables, Operations, Basic I/O

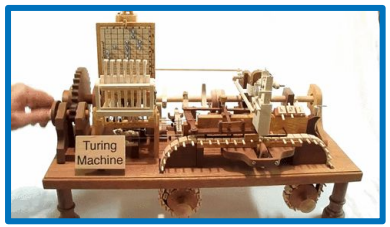
What does a computer do?

When the idea of computers was envisioned by Alan Turing back in the first half of the 20th century, it was called ***Universal Machines***. Unlike the simplest machines, these new kind of machines, later called ***Computers***, were able to take decisions according to different inputs in order to produce a meaningful output.



Alan Turing
1912 - 1928 - 1954

Everything that we do today with computers is based on this idea formulated by Alan Turing almost 100 years ago.



A computer is a device that, given an input, **takes some decisions** and delivers the appropriate output.

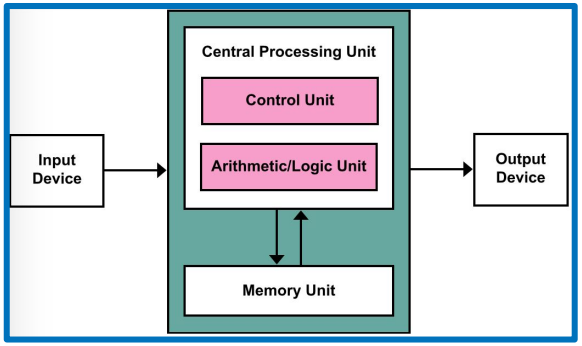
The kinds of inputs are usually motivated by a user, and the outputs are directed to a user as well.

Video: Alan Turing - Celebrating the life of a genius.
Wikipedia: Computer.

What structure does a computer have?

Computers became structured as the first computer scientists (and often they were not called as such) defined that the computer instructions would be stored in a memory. The first one to propose that was the physicist John Von Neumann, who defined the Von Neumann Architecture.

John Von Neumann
1903 - 1940 - 1957



According to this a computer is composed of:

- Instructions and data stored in a memory unit;
- A Central Processing Unit that fetches commands and executes them.

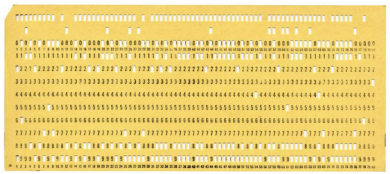
All computers have this basic architecture.

Video: Von Neumann Architecture.

How to use a computer?

The Von Neumann architecture remains the same, but what we call “memory” has evolved considerably. Roughly, we can say that there are two levels of memory:

- Primary memory (normally counted in terms of 4 Gb, 8 Gb, 16 Gb, or more) once called RAM - Random Access Memory;
- Auxiliary memory (normally counted in 256 Gb, 512 Gb, 1 Tb, or more) once called HDD - Hard Drive Disk (now, more likely SSD - Solid State Drive).



Actually, there are many memory levels more in modern machines (cache hierarchy, external drives, the cloud, etc), but from a user’s point of view they are called Unified Memory (for the primary) and Storage(for the auxiliary).

Video: The History of Computer Storage.

Basic OS and Programming

All computer systems are essentially the same, but with a different look and feel. They differ in the way they are perceived, their user interface. Yet nowadays, there is a convergence of technologies making computer systems more similar than ever before.

Computer systems are any form of device that provides digital computation.

- What is a Computer?
- **File Structure and File Formats**
- Installing Python, Hello World!
- Variables, Operations, Basic I/O

File Structure

The organization in the auxiliary memory, the computer storage, is organized in the file system tree structure:

- The file structure starts with drives that usually represents physical devices or partitions of devices;
- The drives are composed of files and folders;
- The folders can have folders and files within it;
- The files can hold data to be accessed by programs.

Most accessing of the file structure is done using the graphical user interface (GUI) of an Operating System (OS) where folders (also called directories) and files can be visualized. The file system can also be directly accessed by a command line interface (terminal).

Videos: Windows 11 File Explorer and MacOS Finder.

File Formats

There are several file formats commonly encountered in different operating systems. Here is a brief list of useful ones to know:

- Textual files: .txt
- Image files: .jpg, .gif, .png, .bmp, ...
- Video files: .avi, .mpg, .mp4, .mov, ...
- Portable document format files: .pdf
- Comma separated values files: .csv
- Tab separated values files: .tsv
- Compressed (zipped) files: .zip or .rar
- Files for specific apps:
  - .docx, .xlsx, .pages, .numbers, ...
  - .py, .cpp, .java, .html, .css, .js, ...

You can configure your graphical interface to omit file types. Usually, it is a bad idea.

Videos: Zipping Files in Windows 11 and MacOS.

Virtual Drives

A common design choice in Operating Systems is the use of virtual drives, i.e., drives that are not physical, but just a format that follows a given standard and which the OS treats as a connected device.
The more common use of such virtual disks is done using the standard ISO-9660 defined by the International Standards Organization. Such formats are known as an ***ISO image***, and they are commonly employed to replace a CD or DVD disk, or even a flash drive. In both modern Mac OS and Windows the ISO images are automatically handled by the OS by either double clicking an ISO image file, or right-clicking and choosing “mount”. In Mac OS, it is also frequent to find a DMG format (Apple Disk iMaGe) with similar behavior.

Creating a disk image in Mac OS or in Windows.

Basic OS and Programming

All computer systems are essentially the same, but with a different look and feel. They differ in the way they are perceived, their user interface. Yet nowadays, there is a convergence of technologies making computer systems more similar than ever before.

Computer systems are any form of device that provides digital computation.

- What is a Computer?
- File Structure and File Formats
- **Installing Python, Hello World!**
- Variables, Operations, Basic I/O

Python Language

In this course, and in several others in the MSCS Program as well, we will use the Python programming language. This language was created by Guido Van Rossum, a big fan (like me!) of the English comedy group Monty Python, hence the name of the language.

Python was a language designed from the beginning to offer the user easy code generation, with few restriction rules, simple commands, and a large number of built-in functions to make code development fast.

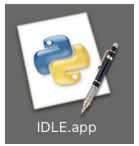
By 2008, the Python language was subject to a major revision that nearly created a completely different language. It was called Python version 3. We, and mostly everyone else, will ignore versions previous to Python 3.0.

Wikipedia: Python (programming language).

Installing Python Language

To install Python in your machine you need to download it from the official webpage (python.org). Just go there, choose Download and pick the appropriate version for your OS (likely the website will figure out which one is the right for you). Any version 3.X should do, but prefer versions 3.7 or higher.

Once you download it, you should have an installer (.exe for Windows, or .pkg for MacOS). Usually, double click it and follow the instructions. In general, it is pretty safe to install from official sources. This one certainly is.

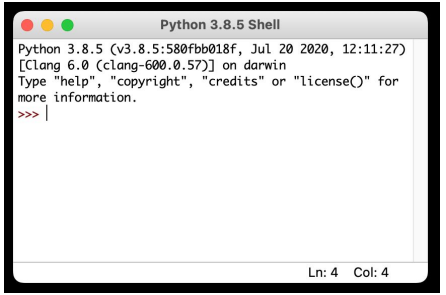


This will install for you the Python interpreter and a basic IDE - Integrated Development Environment - called IDLE.

If you are struggling to install in Windows or MacOS.

Your very first Python program … or not …

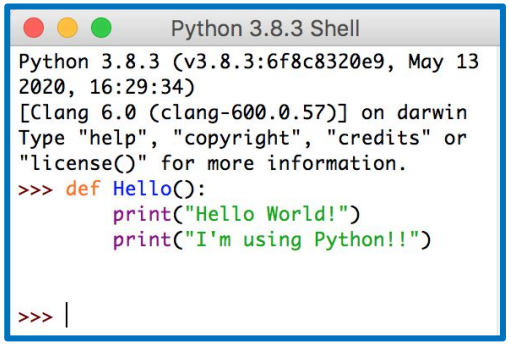
Open IDLE and you should see a window like this one here —---------->



In this window type in:

- ***print("Hello Simpler World!")***

But, this is just your first Python command. Now type in the code to the right —---------->



Now, type:

- ***Hello()***

Yes, now you have run your very first Python program!

If you want to use another IDE than IDLE, make sure your code runs also on IDLE (I will grade on IDLE).

Basic OS and Programming

All computer systems are essentially the same, but with a different look and feel. They differ in the way they are perceived, their user interface. Yet nowadays, there is a convergence of technologies making computer systems more similar than ever before.

Computer systems are any form of device that provides digital computation.

- What is a Computer?
- File Structure and File Formats
- Installing Python, Hello World!
- **Variables, Operations, Basic I/O**

Python Variables

Inside a Python program, a **variable** is something capable of holding information.

All variables have an unique identifier that is used to refer to it.

Python variables can hold several types of data, the more basic ones are:

- A number, either an Integer or a Real (float) (Complex too - but not now);
  - Integers are positive, negative, or zero whole numbers;
  - Reals are positive, negative, or zero numbers with a decimal point in it;
- A string, a sequence of characters representing, for example, a letter, a symbol, a word, or a sentence:
  - Strings are delimited by single or double quotation marks:
    - For example: ***"Monday"***, ***'the first day of the week'***, or ***"US$ 56.12"***;
- A boolean value, either ***True*** or ***False***.

Video: Basic Data Types in Python.

Python Identifiers

Inside a Python program a **variable** is referred by an **identifier**.

Identifiers cannot be reserved keywords, but we will see that soon.

Python identifiers:

- Must start with a letter (lower or uppercase) or in specific cases start with an underscore symbol (_);
- Can have digits or underscore symbols in it;
- Must be unique - variables in a function cannot have the same identifier;
- In an identifier, like everything else in Python, upper and a lowercase letters are not the same - the variables ***time*** and ***Time*** are not the same;
- Valid identifiers examples are ***i***, ***acc***, ***Time5***, ***fifthTime***, ***A_34b***;
- Invalid identifiers examples are ***3A***, ***us$***, ***rate%***, ***rate-percent***, ***i(f)***.

Text: Identifiers in Python.

Python Identifiers

Inside a Python program a **variable** is referred by an **identifier**.

Identifiers are composed of letters, digits, and the symbol "_", but they cannot start with digits, as we saw.

Python identifiers cannot be any of the 35 reserved keywords:

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

Text: Python Keywords.

Python Assignments

Inside a Python program a **variable** is something capable of holding information. **Literals**, however, are raw information.

Literals are raw information as a number or a string. For example:

- **45**, **-2**, **0**, **-3.0**, **3.1459**, **6.02E+23**
- **"This is a long string that doesn't have a problem with apostrophes"**, **'short'**, **"45"**, **'here comes a tab\tNow a line break\n... and a few dots'**

Python can assign values to variables using the assign operator "**=**":

- **a = 56**
- **b, c = "thirteen", 13**
- **a = 56 // 6**
- **b, c = "twenty"+" "+"two", 13+a**

Text: Python literals.

Python Operations

Go to Python Interpreter and type in the equation to see the answer.

According to the type of a variable (or literal constant), some operations are allowed.

- **Numbers** (Integers or Reals) accept the following arithmetic operators in this precedence order:
  - Parenthesis **( )**
  - Exponent ******
  - Negation unary **-**
  - Product/Division/Modulo *** / // %**
  - Sum/Subtraction **+ -**

(54 + 4) / 2 ** 3 ?

(54 + 4) // 2 ** 3 ?

(54 + 4) % 2 ** 3 ?

45 - 3 + 12 - 8 ?

32 * 4 / 8 + 8 ?

34 + -3 * 5 ?

Video: Parenthesis and precedence in Python.

Python Operations

Go to Python Interpreter and type in the equation to see the answer.

According to the type of a variable (or literal constant), some operations are allowed.

- **Strings** accept the following operators in this precedence order:
  - Indexing **[ ]**
  - Slicing **[:]**
  - Repetition *****
  - Concatenation **+**

```
"dog" + "cat" * 2          ?

"dog" * 2 + "cat"          ?

d = "dog"
d[1]                       ?

d = "dog"
d[-1]                      ?

s = "abcdefghijkl"
s[3:]                      ?
s[:-4]                     ?
s[2:8:2]                   ?
```


Video: String Slicing with Python.

Python Operations

Go to Python Interpreter and type in the equation to see the answer.

According to the type of a variable (or literal constant), some operations are allowed.

- **Booleans** accept the following logical operators in this precedence order:
  - Negation **not** (in logic ~)
  - Conjunction **and** (in logic ∧)
  - Disjunction **or** (in logic ∨)

Use parenthesis!

not (not a or not b) ??

```
a = True
b = False
a and a      ?
a and b      ?
b and a      ?
b and b      ?

a or a       ?
a or b       ?
b or a       ?
b or b       ?

not a        ?
not b        ?
```


Video: Truth Tables.

Python Operations

Go to Python Interpreter and type in the equation to see the answer.

According to the type of a variable (or literal constant), some operations are allowed.

- **Comparison** operators (lower precedence than arithmetic) result in **True** or **False**:
  - Equal **==**
  - Different **!=**
  - Greater/Smaller than **>** **<**
  - Greater/Smaller than or equal **>=** **<=**
  - Belonging/Not belonging **in** **not in**

```
24 > 32 / 2                  ?

3 * 4 >= 10                  ?

3 * 4 != 12                  ?

3 * 4 == 12 % 6              ?

"a" in "cat"                 ?

d = "dog"
d[2] < "f"                   ?
d[0]+"a"+d[-1] <= d          ?
```

MERRIMACK COLLEGE

Video: Python Comparison Operators.

Python Basic Output

Go to Python Interpreter and type in the command to see the answer.

The built-in **print** function prints whatever variable, literal or combination (expression) you pass along.

- Unless informed otherwise, the print function receives a list of expressions and prints it out in the default output device converting everything to strings and putting a blank space between the expressions values and a break line at the end, for example:
  - **print("The sum of", 3, "and", 4, "is:", 3+4)**
- There are other uses of *print*, like writing into files, but we'll see that later.
- You can change the separator and end using the options **sep=** and **end=**.

Text: Python print() Function.

Python Basic Input

Go to Python Interpreter and type in the commands to see the answer.

The built-in **input** function prints a prompt string, then receives a string from the user.

- Unless informed otherwise, the input function stops the code execution and waits for the user to input from the default input device a string, for example:
  - **name = input("Enter your name: ")**
- If you want something else than a string you have to convert the input string to the new type. For example, when getting a number, you might use the built-in function **eval**:
  - **age = eval(input("Enter your age: "))**

Text: Python eval() Function.

Text: Python input() Function.

This Week's tasks

- Post discussion D#1
- Coding Project P#1
- Quiz Q#1

Tasks

- Post in the discussion what was your previous experience with Python, and which of the topics seen today was more surprising to you.
- Coding Project #1, compute the year of birth.
- Quiz #1 about this week topics.

Post Discussion - Week 1 - D#1

Post in the discussion what was your previous experience with Python, and which of the topic seen today was more surprising to you.

- You might know Python very well, or know another programming language, or even not know programming at all. However, there is always something new that we learn, even with basic stuff as shown in this class.
- Which among the topics shown today was more surprising to you?
  - Use personal opinions, but justify your opinion with technical reasons.

Your task:

- Post your discussion in the message board until this Monday;
- Reply to posts of your colleagues in the message board until next Thursday.

This task counts towards the Discussions grade.

First Coding Project - Week 1 - P#1

- Write a Python program that:
  - Asks the user their name, their age, and the current year;
  - Then, your program should print out a message like:
    - "**Dear <name>, you were born either in <year1> or <year2>**"
  - For example, if the user enters: **Arnold Schwarzenegger**, **75**, and **2023**, your program should prints out:
    - "**Dear Arnold Schwarzenegger, you were born either in 1947 or 1948**"

Your task:

- Go to Canvas, and submit your Python file (.py) within the deadline:
  - The deadline for this assignment is Next Thursday.

This assignment counts towards the Projects grade.

First Quiz - Week 1 - Q#1

- The first quiz in this course covers the topics of Week 1.
- The quiz will be available this Saturday, and is composed of 10 questions.
- The quiz should be taken on Canvas (Module 1), and it is not timed.
  - You can take as long as you want to answer it.
- The quiz is open book, open notes, and you can even use any language interpreter to answer it.
- However, the quiz is evaluated and you are allowed to submit it only once.

Your task:

- Go to Canvas, answer the quiz and submit it within the deadline:
  - The deadline for the quiz is Next Thursday.

This quiz counts towards the Quizzes grade.

Welcome to CSC 6000

- Read the syllabus
- Post discussion D#1 by Monday, reply to colleagues by next Thursday;
- Take quiz Q#1 (available Saturday) by next Thursday;
- Develop coding project P#1 by next Thursday.

Next Week - Number Theory and Binary Logarithms

MERRIMACK COLLEGE