Introduction to Computing — Lab 5 Worksheet

Number Systems: do it by hand, then make the loop agree with you

City of Ilagan College · Session 5 · Saturday 12 September 2026 · ONE SHEET PER STUDENT

Your name: _____ **Group:** _____

Your folder *(lastname then firstname, all lowercase, nothing but letters and digits)*:

_____

Your two addresses, written out in full:

Program: `g____.cic.rmlsystems.net/_____/`

Page: `g____.cic.rmlsystems.net/_____/page.html`

Where you upload, which is the only thing that changed this week:

`app/people/_____/`

> **Hand this in at the end.** Two of the twenty marks are on this sheet, and one of those two is the single line you write in Part 5 and the single line you write in Part 6. They are one sentence each and people still leave them blank.

PART 1 — By hand, before any keyboard

No calculator. No phone. No Python. Two of you work each question separately, then compare answers. Two people arriving at the same answer independently is the cheapest checking method there is, and you will use it for the rest of your life.

1.1 The one everybody does together: 13 into binary

Fill in the divisions. Keep going until the "whole part" column reaches 0.

Divide	Whole part	Remainder
13 ÷ 2		
___ ÷ 2		
___ ÷ 2		
___ ÷ 2		

Read the remainder column BOTTOM TO TOP. 13 in binary is: _____

Check it by going back the other way. Multiply and add:

`___ × 8  +  ___ × 4  +  ___ × 2  +  ___ × 1  =  _____`

Did you get 13 back? ☐ Yes ☐ No — if no, do the divisions again before you go on.

1.2 Your group's Truth Set Card

Copy your eight answers here **from the card**, and get the card signed.

#	The question	Our answer, worked by hand	Who worked it
1	Binary → decimal		
2	Decimal → binary (8 bits)		
3	Decimal → hex		
4	Hex → decimal		
5	Binary → hex		
6	Decimal → **octal**		
7	Hex → binary (8 bits)		
8	Binary addition (show the carries)		

Instructor signature: _____ Time: _____

☐ First try ☐ Sent back once ☐ Sent back twice

1.3 — Which question caused the most argument in your group, and who turned out to be right?

> **This card is now the authority.** For the rest of the afternoon, when the computer and this card disagree, that is not the computer winning. It is a question.

PART 2 — Warm-up at the desk

Open PowerShell (`Win` key, type `powershell`, Enter) and start the Python shell by typing the command on its own:

```
python
```

You get the `>>>` prompt. Type each of these there, one line at a time, and write down **exactly** what comes back. Do not guess and do not copy from your neighbour — you are checking that the machine does what your paper said it would. `exit()` leaves the shell when you are done.

If `python` is not the right command on your machine, if the Microsoft Store opened instead, or if Python says `SyntaxError: invalid character` on a line you copied, see **S00 — PowerShell and Python, The One-Page Sheet.**

You type	What came back
`print(13 // 2)`	
`print(13 % 2)`	
`print(2 ** 8)`	
`print(13 / 2)`	

2.1 — `13 // 2` and `13 / 2` gave you two different things. Write down the difference in your own words, and say which one the paper method uses:

Now the counter. Type this and write down what happens:

```
count = 1
while count <= 5:
    print(count)
    count += 1
```

What it printed: _____

2.2 — Now delete the line `count += 1` and run it again. Describe what happened, and how you stopped it:

2.3 — Python's own converters. Run each and write down what came back, including any label Python puts on the front:

You type	What came back
`print(bin(13))`	
`print(oct(13))`	
`print(hex(13))`	
`print(f"{13:b}")`	
`print(f"{13:o}")`	
`print(f"{13:X}")`	
`print(int("1101", 2))`	
`print(int("D", 16))`	

2.4 — `bin(13)` and `f"{13:b}"` do not print the same thing. What is the difference, and which one would you put in a table on a web page?

PART 3 — The group build: `/convert`

Our group's route is live at: `g____.cic.rmlsystems.net/convert`

The range our group chose: from _____ to _____

Why we chose it, in our own words:

How many rows that is: _____ How many `<tr>` lines the leader typed: _____

The rotation — who drove, and when

The keyboard changes hands every ten minutes, and **the person who has not driven yet goes next.**

Minutes	Who was driving
0–10	
10–20	
20–30	
30–40	
40–45	

3.1 — Check three rows of the group's table against Part 1's method, by hand. Pick rows nobody has looked at yet.

Row	The page says	Our hand working	Agree?
decimal _____	bin _____ oct _____ hex _____		☐
decimal _____	bin _____ oct _____ hex _____		☐
decimal _____	bin _____ oct _____ hex _____		☐

3.2 — Change `LAST` from 15 to 63 and reload. How many characters did the leader change, and how many rows appeared?

Characters changed: _____ Rows now: _____

Then put it back to whatever your group decided.

PART 4 — Your own page

MY_NUMBER is: _____ Because it is: _____

(the day of the month you were born, or the last two digits of your student number — it has to be 1 or more, and it should not be the same as the person beside you)

4.1 Work it out by hand FIRST. The page has to agree with this box, not the other way round.

Divide	Whole part	Remainder
_____ ÷ 2		
_____ ÷ 2		
_____ ÷ 2		
_____ ÷ 2		
_____ ÷ 2		
_____ ÷ 2		

My number in **binary**, remainders read bottom to top: _____

My number in **octal** *(divide by 8 instead, same method)*: _____

My number in **hexadecimal** *(divide by 16; remainders 10–15 are written A–F)*: _____

4.2 Now the page

- ☐ `index.py` uploaded to `app/people/<myfolder>/` — **not** `public_html`
- ☐ My address loads with no traceback
- ☐ The name, group and town on it are mine
- ☐ `MY_NUMBER` and `WHY` are mine
- ☐ The working table has one row per division and ends at 0
- ☐ The sentence under it says my loop and Python agree
- ☐ The sixteen-row table starts at my number

4.3 — Did the page's binary match the box in 4.1? ☐ Yes ☐ No

If no: which was wrong, the paper or the page, and how did you find out?

4.4 — Look at the last row of your working table. The "whole part" column is 0 there. Explain, in one sentence, why that is the row where the loop stops:

4.5 — Someone in your group has the same birthday as you. Their page and your page will show the same number. What else on the two pages has to be different, and why is that not cheating?

> **Eight identical pages is eight zeros.** Your number, your reason, your town, your words.

PART 5 — The HTML twin

Open your two addresses side by side:

`.../<yourfolder>/` and `.../<yourfolder>/page.html`

Now type the same content into `index.html` **by hand**. Every number in it you work out yourself.

Stop when you are told to stop. You are not expected to finish.

How many of the sixteen rows did you finish? _____

How many minutes did you spend on them? _____

How many lines of Python produced all sixteen? _____ *(count the lines inside `sixteen_rows()`, from `rows = ""` to `number += 1`)*

5.1 — The count, side by side

	Python side	HTML side
Lines you typed		
Values you worked out by hand		
Rows that appeared		

5.2 — THE ONE LINE. What was annoying about the HTML side? One sentence, your own words.

5.3 — Now change your birthday number to a different number. How many places did you edit in `index.py`? How many in `index.html`?

`index.py`: _____ `index.html`: _____

5.4 — Add the honest sentence to the bottom of `index.html` saying which row you stopped at. ☐ Done

PART 6 — The AI task

Tool used: ☐ ChatGPT ☐ Gemini ☐ Claude

The prompt, exactly:

> *"Write Python that converts a decimal number into a binary string without using bin(). Show me the whole thing."*

Copy what it gave you here *(or staple the printout to this sheet)*:

_____

_____

_____

_____

_____

_____

6.1 — Go through it line by line and tick everything it used that you have not been taught:

☐ `bin()` ☐ a list `[ ]` ☐ `.append()` ☐ `for` ☐ `if` ☐ `reversed()` ☐ `"".join()` ☐ recursion (the function calls itself) ☐ `divmod()` ☐ **a function with something in the brackets** — `def to_binary(n):` — a value handed in from outside. You have only ever written `def page():`, with empty brackets. This one is **Session 8, 3 October.** Write that date next to the tick. ☐ something else: _____

6.2 — Rewrite it using only a variable, `while`, `//`, `%`, `+` and `+=`. No function, no brackets, no list. Just lines that run down the page, the way your `my_binary()` loop already works. **And make it do one thing the AI's version did not do: count the divisions.** Write it here first, then type it into a new file called `binary.py`, save it (Notepad: **Save as type: All Files**), and run it in PowerShell with `python .\binary.py`:

```
NUMBER = 27

answer = _______        (an empty string, and it is not the word "empty")

digits = _______

left   = _______

while ____________________________________________

    answer = ____________________________________

    digits ____________________________________

    left   ____________________________________
```

```
print(f"_____")

print(f"Binary digits: _____")

print(f"Python says: {NUMBER:b}")
```

Test yours. Change `NUMBER`, save the file, then run `python .\binary.py` in PowerShell and read what comes back. Four times. Python's own answer, to check yours against, is two lines at the `>>>` shell:

```
NUMBER = 27
print(f"{NUMBER:b}")
```

Set `NUMBER` to	Your loop prints	Digits	`f"{NUMBER:b}"` says	Same?
`27`				☐
`1`				☐
`255`				☐
`0`				☐

One of those four rows is not the same on both sides. Which one, and what did your loop print there?

6.3 — THE ONE LINE. What did the AI use that you have not been taught, and why is your version still correct?

6.4 — Your version gets `0` wrong. The AI's does not, because the AI wrote a guard. What is that guard, and why can you not write it yet?

6.5 — The AI handed its number in through the brackets, so it can convert any number without the file being edited. You have to change `NUMBER`, save, and run the file again every time. Write down the one date that fixes this for you:

Session _____, _____ 2026.

6.6 — AI usage log. Everything you asked any tool this afternoon, including while debugging.

Tool	What you asked it	What you kept, changed or threw away

Tool	What you asked it	What you kept, changed or threw away

Declaration. *I used the tools listed above. I tested everything they gave me against work I did myself, and I can explain every line on my page.*

Signed: _____

PART 7 — When it broke

Everybody breaks something today. Write down at least one.

What the last line of the traceback said	What you had actually done	How you fixed it

7.1 — Your page had a traceback on it for a while. What were the other seven people in your group seeing on their pages during that time?

PART 8 — Think about it

8.1 — Sixteen rows came out of a loop you typed once. Where else have you seen a computer do something sixteen times that a person would have done by hand?

8.2 — Your loop and `f"{NUMBER:b}"` gave the same answer. One of them shows the method and one hides it. Which would you rather have been taught first, and why?

8.3 — A `while` loop that never stops is not an error. Python does not complain about it. Why is that more dangerous than a crash?

8.4 — Name one thing you now check every time, that you would not have thought to check this morning.

8.5 — Your folder moved this week, from `public_html` into `app/people`, and your address did not change at all. In your own words, one sentence each: what did the server do when somebody asked for your page LAST Saturday, and what does it do TODAY?

Last Saturday: _____

Today: _____

And one more sentence: why could today's sixteen-row table not have been published from the old folder?

Back page — the cheat sheet

Direction	Method
Decimal → binary	Divide by 2 over and over. Keep the remainders. Read them **bottom to top**.
Decimal → octal	Same, dividing by 8. Remainders are 0–7 only.
Decimal → hex	Same, dividing by 16. Remainders 10–15 are written A B C D E F.
Binary → decimal	Write 1 2 4 8 16 32 64 128 above the bits, from the right. Add the worths above a 1.
Binary → hex	Group four bits **from the right**. Each group is one hex digit. Pad the left group with zeros.
Binary → octal	Group **three** bits from the right. Each group is one octal digit.
Hex → binary	Every hex digit becomes exactly four bits, leading zeros kept.

Powers of two: 1 · 2 · 4 · 8 · 16 · 32 · 64 · 128 · 256 · 512 · 1024

Hex digits: 0 1 2 3 4 5 6 7 8 9 A(10) B(11) C(12) D(13) E(14) F(15)

Octal digits: 0 1 2 3 4 5 6 7 — and there is no 8 and no 9. If you write one, you have made a mistake.

Binary addition: 0+0=0 · 0+1=1 · 1+0=1 · **1+1=10 (write 0, carry 1)** · 1+1+1=11 (write 1, carry 1)

This week's Python

Line	What it does
`13 // 2`	Whole division. Throws the remainder away. Gives `6`.
`13 % 2`	The remainder only. Gives `1`.
`2 ** 8`	To the power of. Gives `256`.
`count = 0`	A counter starts at nothing, before the loop.
`count += 1`	Add 1 to `count` and keep it in `count`.

Line	What it does
`rows = ""`	An empty piece of text, ready to be built up.
`rows += "abc"`	Stick `"abc"` onto the end of `rows`.
`number = 13`	The box the next line is written against.
`while number > 0:`	Do the indented lines again and again, for as long as `number` is above 0.
`bin(13)`	Gives `0b1101` — with a label on the front.
`oct(13)`	Gives `0o15`.
`hex(13)`	Gives `0xd`.
`f"{13:b}"`	Gives `1101` — no label. This is the one for a table.
`f"{13:o}"`	Gives `15`.
`f"{13:X}"`	Gives `D`.
`int("1101", 2)`	Read that text as a base-2 number. Gives `13`.

Always check your answer by converting it back. It costs ten seconds and catches nearly everything.