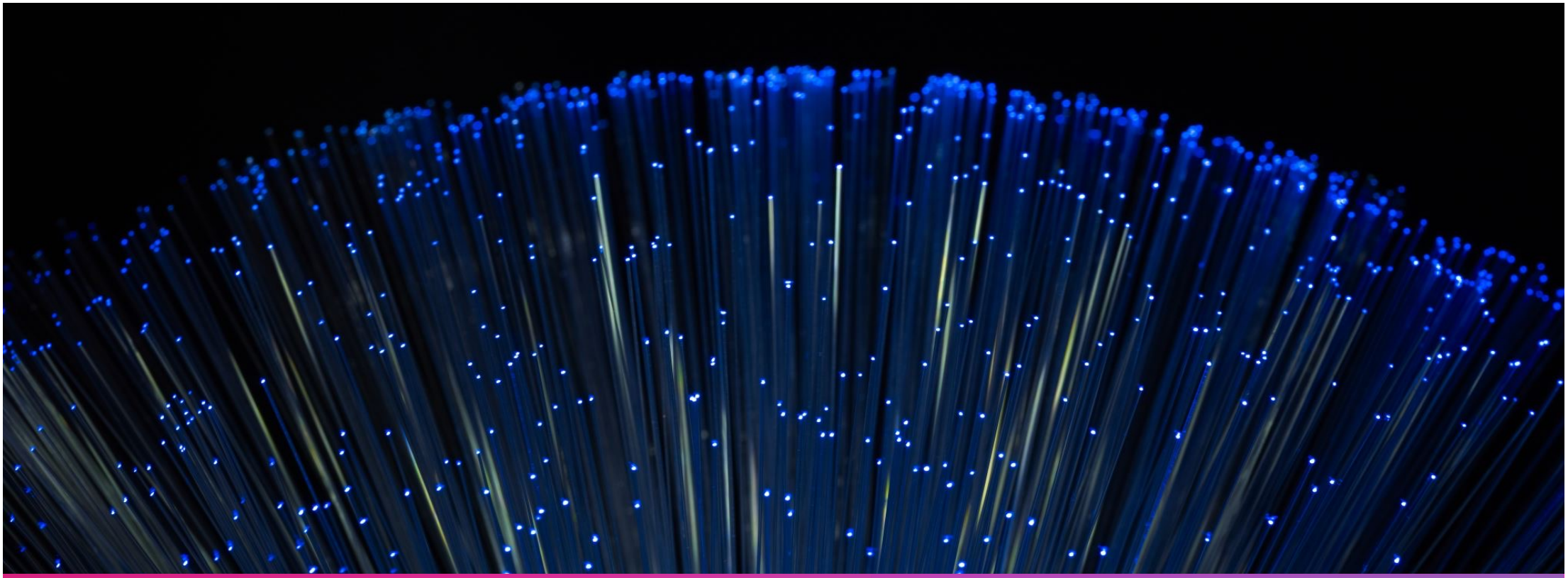

WEEK 2: CONDITIONAL STRUCTURES

BOOLEANS

DEFINITION

- The boolean data type represent the two truth values of logic and Boolean algebra
- A boolean variable has one of two possible values
- Those values are True and False in python, CAPITAL T, CAPITAL F

COMPARISON OPERATORS

DEFINITION

- A comparison operator is an operator, just like +, –, *, /, // and %
- Unlike the previous operators, a comparison operator does not return a number
- A comparison operator returns a boolean value: True or False
- What is the boolean value of:
  - 1 == 2
  - 2 < 3
  - ‘blue’ == ’BLUE’

COMMON COMPARISON OPERATORS

- == tests if two values are equal – not to be confused with the assignment operation =
- != tests if two values are not equal
- < and > test if a value is lower than (resp. greater than) another
- <= and >= test if a value is lower or equal (resp. greater or equal) to another

SIDENOTES

- The operands of a comparison operator can be of various types:
  - the == and != operators can test equality between numbers, strings, lists (more on that later), boolean values etc.
  - >, <, <=, >= are mostly used to test inequality between numbers (int or float). They can be used with complex structures, but we will not study that this year
  - The + operator is not only for adding numbers. You can also **concatenate** strings with it:
    - “abc” + “def” results in “abcdef”

BOOLEAN OPERATORS

DEFINITION

- A boolean operator is an operator that has (a) boolean value(s) as its operand(s)
- The “and” operator returns true iff. both its operands are True
- The “or” operator returns true when at least one of its operand is true
- the “not” operator returns the oposite of its only operand

TRUTH TABLES

NOT

x	x'
0	1
1	0

AND

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

OR

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

COMPLEX BOOLEAN EXPRESSIONS

- Boolean operators can be composed like arithmetics operators
- They all have the same priority. We use parentheses to explicit priorities
- What are the values of:
  - ((1 == 2 or 2 >= 2) and (1 > 0 and not('blue' == 'blue'))
  - (((1 == 1 or not(4 == 4)) or 4 >= 2) and not(7 == 5))

IF STATEMENTS

BASIC NOTIONS

- An if statement enables us to execute code conditionnaly.
- An if statement is made of:
  - the if keyword
  - followed by a boolean value or expression
- Examples:
  - if my_boolean_var: # my_boolean_var = True
  - if x <= 5:

ELSE AND ELIF

- if allows us to execute some code if a condition is True
- elif allows us to execute some code if the condition for “if” was false, and another condition is true
- else allows us to execute some code if all the conditions in the preceding “if” and “elif”s were false.

EXAMPLE

```
x = -10.0 # Initial value of x

if x> 0.0:
     print ('The initial value of x is greater than 0')
     x -= 20.0
elif x <0.0:
     print ('The initial value of x is less than 0')
     x += 21.0
else:
     print ('The initial value of x is neither greater than 0 nor less than 0, it must be 0!')
     x *= 2.5

# Print the new value of x
print ("The new value of x is:", x)
```

if statement

elif statement

else statement (no boolean expr.)

2 lines executed if statement true
Beware the indentation!

INDENTATION SIDENOTE

Describe the following piece of code mathematically as a piecewise linear function of x:

```
if x < 5:
        x += 2
x –= 2
print(x)
```