COMP10001

2023 Sem 1

Foundations of Computing

Week 4

Tutorial/Workshop ○ ○ ○ ○

Today's Tutorial

1. Logical operators
2. Indexing and slicing
3. Functions – print vs return

Boolean data types

Stores a truth value of **True or False**

Other data types can be **converted** to Boolean

Numbers:

0 = False, others = True

Strings:

"" = False, non-empty = True

Logical Operators

Combines Boolean values to return a single truth value

e.g.

```
>>"Hello" and 3
True
```

Relational Operators

Compares two values to produce a truth result

e.g.

```
>>(2*2) == 4
True
```

In groups, fill in the below table.

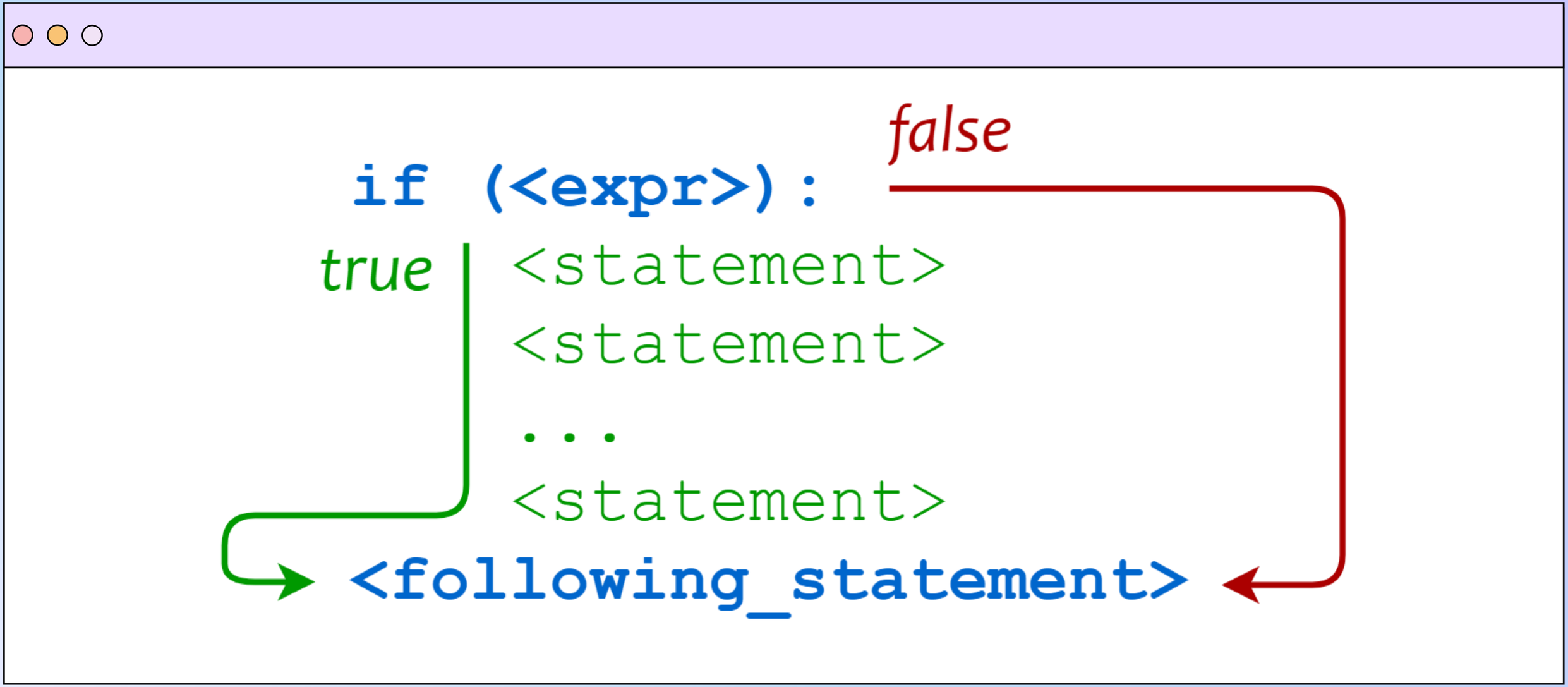
2. For each of the following, identify whether it is: (a) a Boolean value; (b) a relational operator; or (c) a logical operator.

`==`		`>`		`False`	
`!=`		`and`		`<=`	
`or`		`>=`		`not`	
`True`		`<`			

Completed table

==	Relational	>	Relational	False	Boolean
!=	Relational	and	Logical	<=	Relational
or	Logical	>=	Relational	not	Logical
True	Boolean	<	Relational		

If statements



If statements

Do not always need an "else" for every if; however do need an if before an else.

```
if ____:
elif ____:
else:
```

```
if ____:
if ____:
else:
```

```
if ____:
if ____:
if ____:
```

```
if ____:
else:
if ____:
```

Exercises 1-4

Exercise 1 answer

1. Evaluate the following truth expressions:

(a) `True or False`

A: *True*

(b) `True and False`

A: *False*

(c) `False and not False or True`

A: *True*

(d) `False and (not False or True)`

A: *False*

Exercise 2 answer

2. For each of the following if statements, give an example of a value for `var` which will trigger it and one which will not.

   (a) `if 10 > var >= 5:`

   A: *A number 5 or greater and lower than 10 will trigger this if statement. A number outside this range will not. A value other than a number will cause a TypeError.*

   (b) `if var[0] == "A"and var[-1] == "e":`

   A: *A string which begins with "A" and ends with "e", for example `"Apple"`, `"Antelope"` or `"Ae"` (with that capitalisation). A list or tuple with `"A"` and `"e"` as the first and last items will also pass. Other strings and sequences will not trigger the conditional, and any sequence which does not have at least one one item will give an IndexError. A non-sequence type will produce a TypeError.*

   (c) `if var in ("VIC", "NSW", "ACT"):`

   A: *Only strings "VIC", "NSW" and "ACT" will trigger this condition. Any other value of `var` will result in a False result.*

   (d) `if var:`

   A: *This condition will convert `var` into a boolean value, so if it is non-zero/non-empty, the if statement will be triggered.*

Exercise 3 answer

3. What's wrong with this code? How can you fix it?

```
letter = input("Enter a letter: ")
if letter == 'a' or 'e' or 'i' or 'o' or 'u':
    print("vowel")
else:
    print("consonant")
```

A: *Logical operators separate conditions, so* `letter == 'a'` *will be evaluated completely separately from* `'e'` *and the rest of the conditions. Strings with one character will always evaluate to True so the logical statement* `letter == 'a'or True or True or True or True` *will always evaluate to True which is undesired. This can be fixed by writing out* `letter ==` *for every condition; or by using the* `in` *operator as* `if letter in 'aeiou':` *or better,* `if letter in ('a', 'e', 'i', 'o', 'u'):`.

Exercise 4 answer

4. What's wrong with this code? How can you fix it?

```
eggs == 3
if eggs = 5:
    print("spam")
else:
    print("not␣spam")
```

A: *This programmer has confused the assignment (=) and equality (==) operators. This can be fixed by swapping them (`eggs = 3`, `if eggs == 5:`).*

Sequences

A data type

Stores a **series of objects**

String is a series of **characters** e.g. "Hello"

Tuple is a series of **anything** e.g. ("hi", 3, 2.7)

List is also a series of **anything** e.g. ["hi", 3, 2.7]

We will explore difference between these later

Indexing

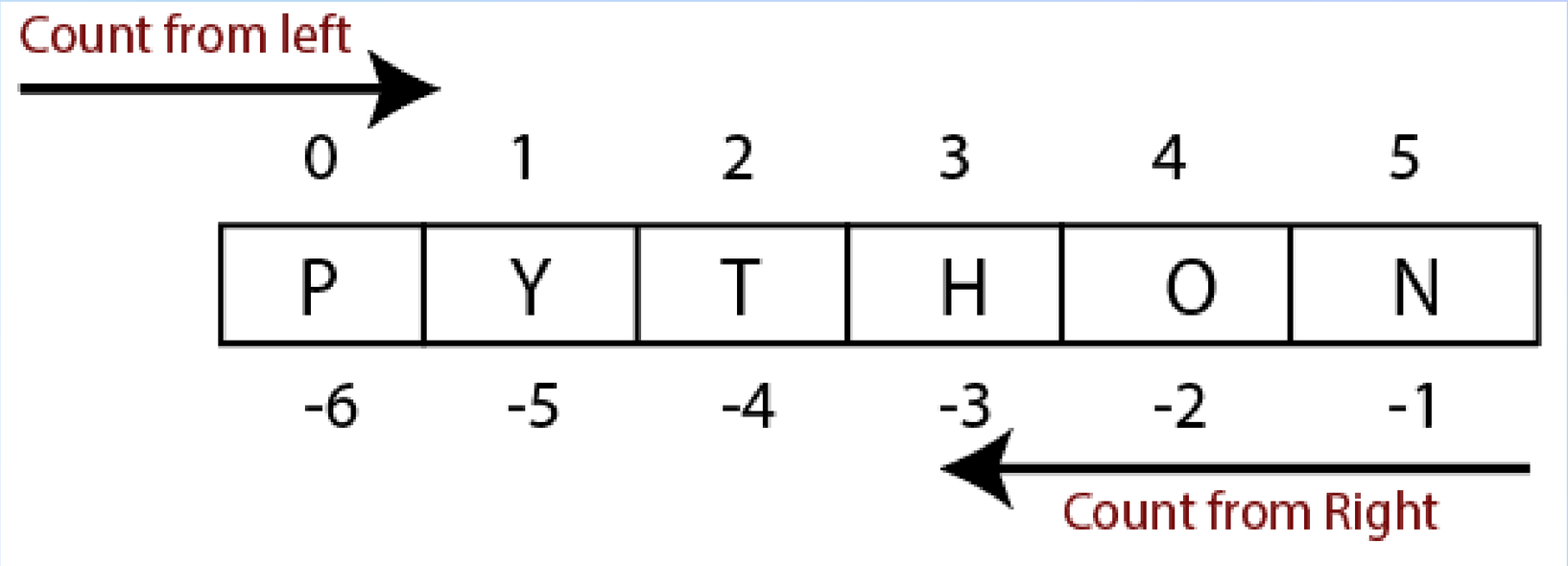
Returns a single element in a sequence

```
e.g. s[2] = 'T'
```

Slicing

Returns a section of a sequence

```
e.g. s[1:4] = 'YTH"
e.g. s[-3:-1] = "HO"
```



Exercise 5

Exercise 5 answer

5. Evaluate the following given the assignment `s = "python"`

(a) `s[1]`

A: `'y'`
(Indexing starts from 0)

(b) `s[-1]`

A: `'n'`
(Negative indexing starts from -1)

(c) `s[1:3] + s[3:5]`

A: `'ytho'`

(d) `s[10]`

A: `IndexError: string index out of range`

(e) `s[10:]`

A: `''` *(Empty String)*

(f) `s[-4:-2]`

A: `'th'` *(Note that even with negative indices, the index of the left-most character is first)*

(g) `s[:-4]`

A: `'py'`

(h) `s[::2]`

A: `'pto'`
(skips every second letter)

(i) `s[::-1]`

A: `'nohtyp'` *(backwards)*

A: *Note that both d and e ask for a portion of the string which doesn't exist, but while d (indexing) gave an error, e (slicing) returned an empty sequence.*

Functions

Block of **reusable code**

Call it by writing the name and then brackets. Even if no arguments, **still need brackets**

We can define our own functions using "def"

```
def my_function(parameter):
    // do something
    return result
```

Return is optional

Why use a function?

Benefits:

- cleaner code (shorter main method)

- reusable (in other programs, throughout 1 program)

- easier to change (less duplication)

Exercise 6

Exercise 6 answer

6. What's wrong with this code? How can you fix it?

```
def calc(n1, n2):
    answer = n1 + (n1 * n2)
    print(answer)

num = int(input("Enter the second number: "))
result = calc(2, num)
print("The result is:", result)
```

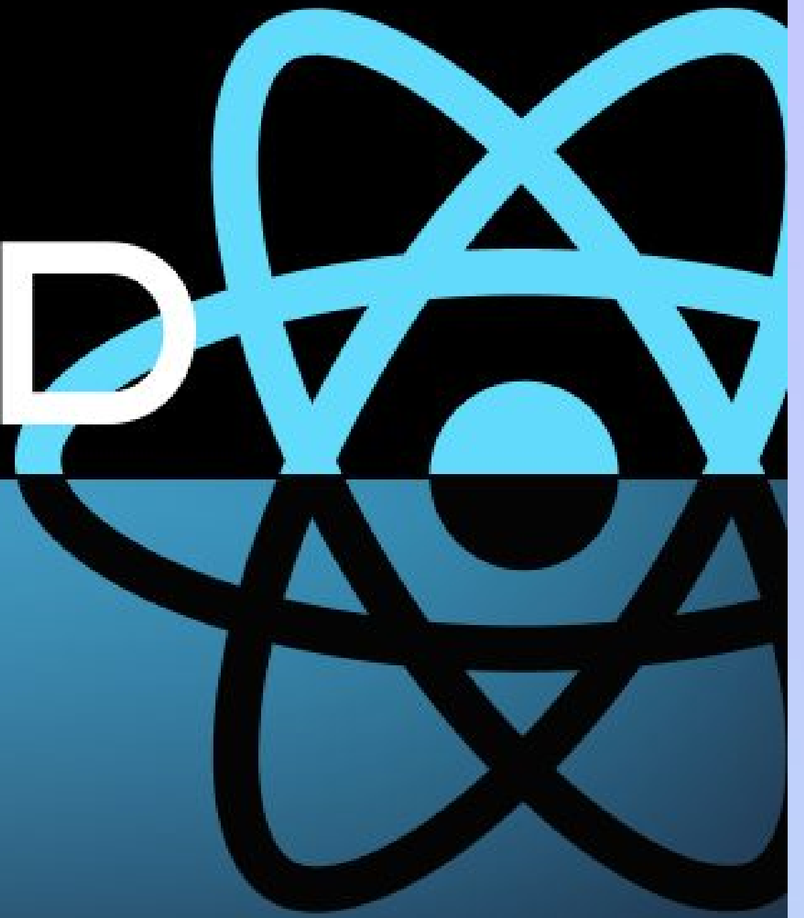
A: *This function prints the answer to the calculation it's performed rather than returning it. This means that the value of* `result` *will be* `None` *and the last line will not work as intended.*

CISSA

AFFILIATED CLUB

Intro to Web Development:

FRONTEND



THURSDAY MARCH 23rd

5:15PM - 6:45PM

OLD ARTS 155

WORKSHOP

Grok, problems from sheet, ask me questions :)

See you next week!