Coursework key dates

Assignment: Single piece of coursework, completed individually.

Theme: Write a program to perform and encryption and decryption task + a short (2 page) report

Set: Friday 19th November 2021 (Week 8)

Deadline: Friday 10th December 2021 (Week 11)

Drop-in support classes

On-campus Group 1	On-campus Group 2	Online
Friday	Friday	Thursday
13:00-14:00	14:00-15:00	9:00-10:00
MVB 1.15	MVB 1.15	remo

Group 1 : EMAT, Innovation, Biorobotics, everyone not listed in Groups 1/2

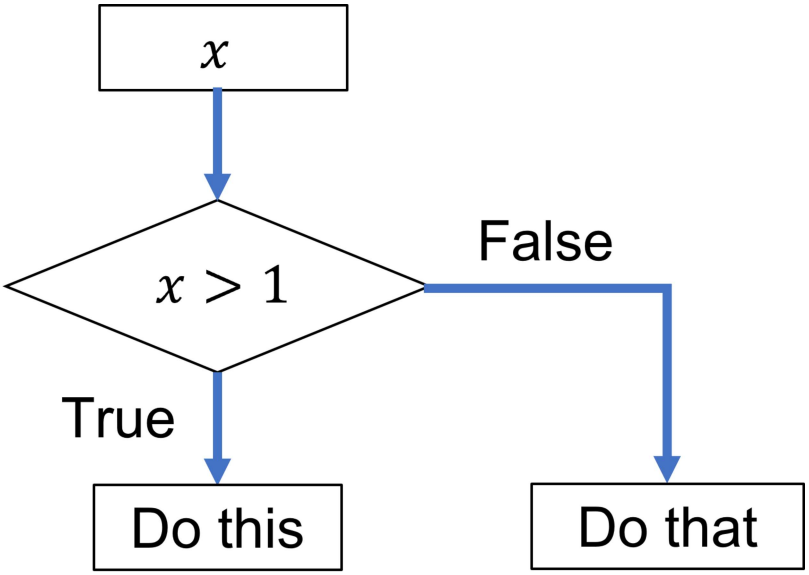
Group 2: EENG, EDES, Digital Health MSc & CDT

Introduction to Computer Programming

2.1 Control Flow

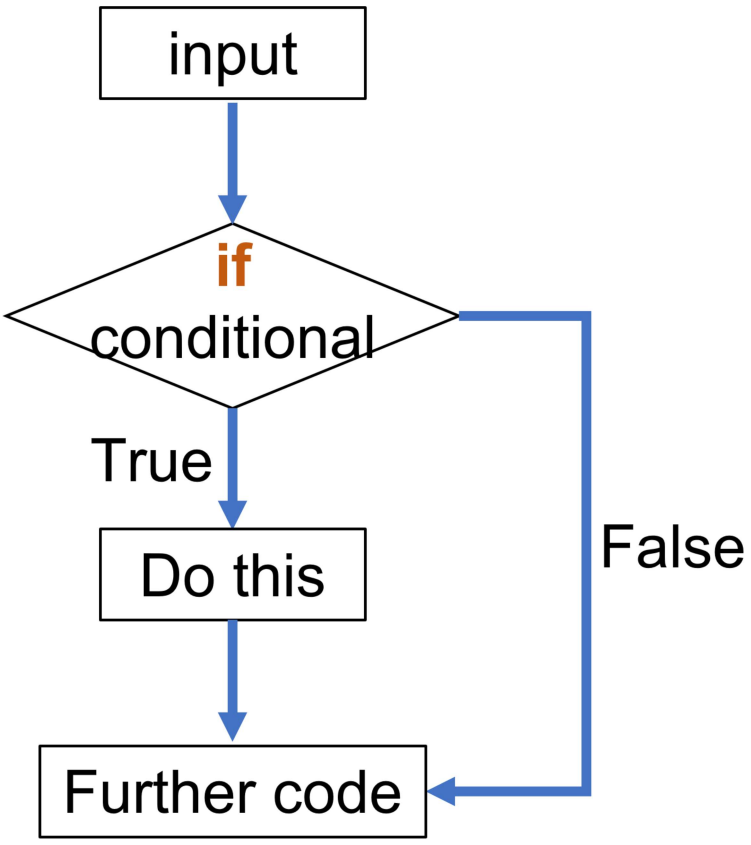
Conditional Statements

- Make decisions within a program and direct the flow.
- Run different blocks of code depending on whether a *Boolean expression* evaluates to `True` or `False` .
- This decision making is known as **Control Flow**



`if`

Runs a block of code only if a condition is `True`



In [20]:

```
x = 3

if x > 10:
    print("Do this") # block of code run if condition is True

print("Further code")
```

```
Further code
```

The colon `:` follows the condition to be evaluated.

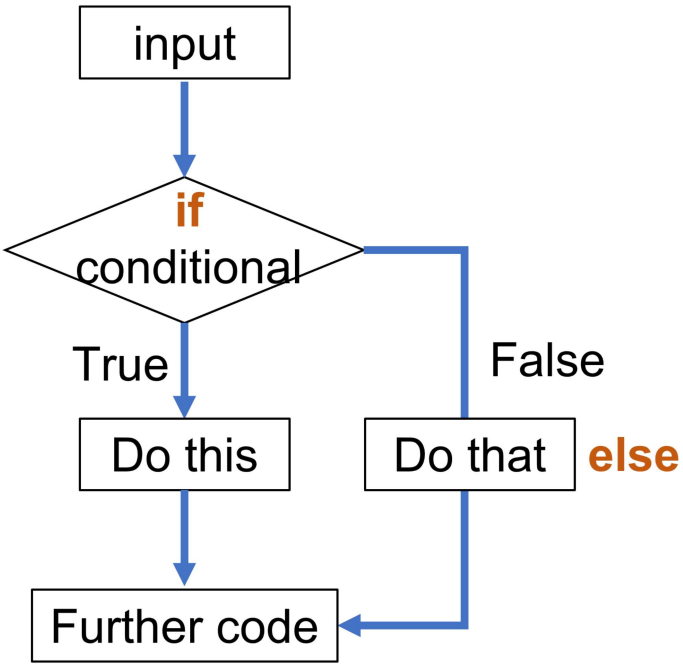
The indent is used to determine which pieces of code are executed in the case that the condition evaluates to `True`.

The indent can be any number of spaces.

- must be the same for all lines in a block of code.
- 4 spaces is considered best practise.
- Many IDEs (e.g. Spyder) automatically indent after you type `if:`.

`if… else`

Runs a block of code only if a condition is `True`
Otherwise runs a different block of code.



In [22]:

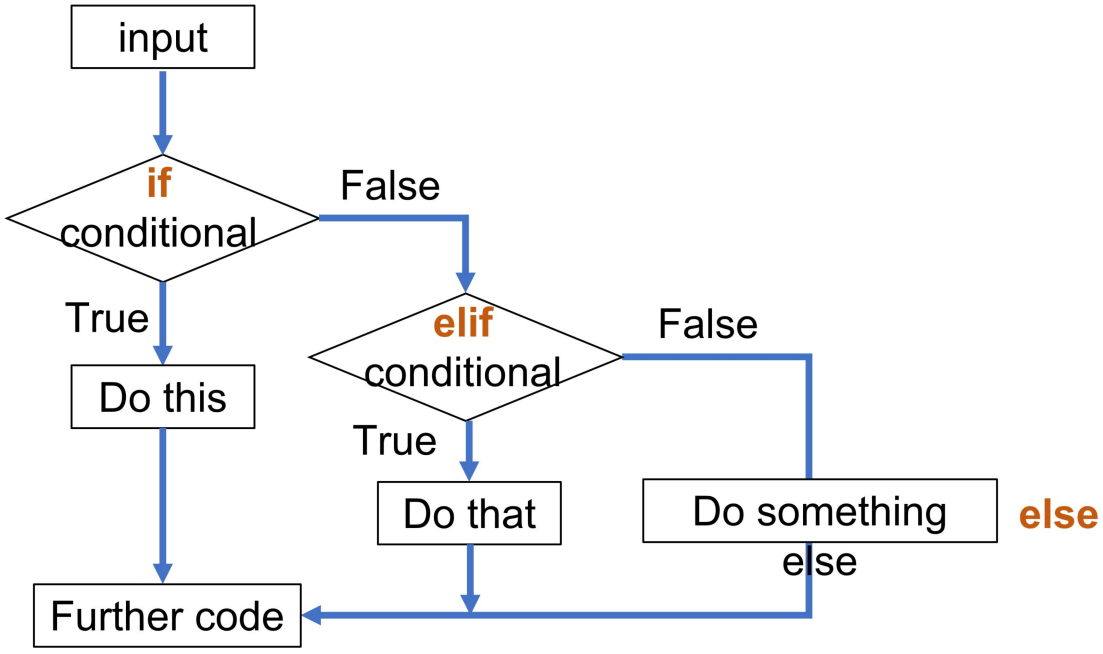
```
x = 2

if x > 10:
    print("Do this") # if condition True

else:
    print("Do that") # if condition False

print("Further code")
```

```
Do that
Further code
```

Note:
Only one of the indented blocks of code (after `if` **or** after `else`) is executed!

`if...elif...(else)`

Runs a block of code only if `if` conditional is `True`
Otherwise runs a different block of code if `elif` conditional is `True`
Otherwise runs a different block of code



Note:
Only one of the indented blocks of code (after `if` **or** after `elif` **or** after `else`) is executed!

In [24]:

```
x = 12

if x > 10:
    print("Do this")           # if condition is True

elif x > 5:
    print("Do that")           # if another condition is True

else:
    print("Do something else") # all preceding conditions False


print("Further code")
```

```
Do this
Further code
```

An unlimited numer of `elif` statements can be used after an `if` statement

The `else` statement is optional.

In [25]:

```
x = -1

if x > 10:
    print("Do this")

elif x > 5:
    print("Do that")

elif x > 0:
    print("Do something else")


print("Further code")
```

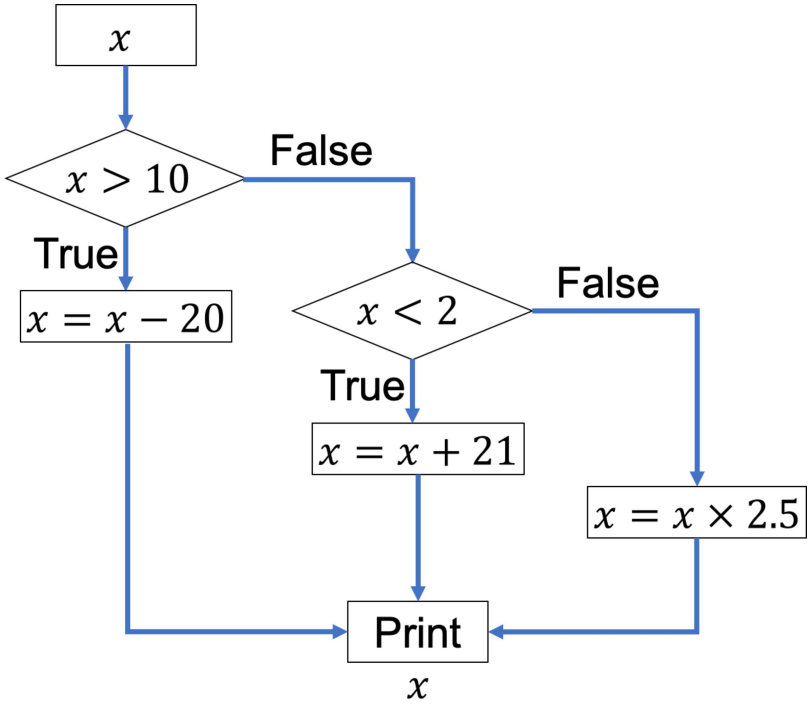
```
Further code
```

Summary

- Conditional statements (`if` , `elif` and `else`) perform a test on an expression with a a Boolean (`True` or `False`) value.
- Execute/skip blocks of code based on the `True` / `False` value of the expression.

In-class Demos

Example 1:
Write a program to modify the initial value of the variable `x` and print the new value, as shown in the flow diagram.



In [49]:

```
x = 20

if x > 10:
    x = x - 20

elif x < 2:
    x = x + 21

else:
    x = x * 2.5

print(x)
```

```
0
```

Variable re-assignment

Note: In programming, `x = x - 20` is used to set the value of `x` to the original value of `x` minus 21.

If this were a mathematical equation, no values of `x` would satisfy the equation.

In programming, however, an expression of this form is used to *reassign* the value of `x`.

Let's remind ourselves of an example from last week.

> **Is it lunchtime?**
> `True` if time between lunch start and end times.
> `False` if not.

Is it time for work?
`True` if time between work start and end times **and not** lunchtime.
`False` if not.

Let's build on the example from last week by including control statements.

Example 2:
Write a program that shows where a person will be based on the time of day.

- at the lab if it is lunchtime **or** time for work
- at home if it is before or after work

In [48]:

```
# --------- Program from last week ------------------
# Variables
t = 16.00           # current time
Ls = 13.00          # lunch starts
Le = 14.00          # lunch ends
Ws = 8.00           # work starts
We = 17.00          # work ends

is_lunchtime = Ls <= t < Le                    # lunchtime (boolean value)

is_work_time = Ws <= t < We and not lunchtime # work_time (boolean value)
#-------------------------------------------------------

if lunchtime == True or work_time == True:
    print('at the lab')
else:
    print('at home')
```

```
at the lab
```

As `is_lunchtime` is equal to either `True` or `False`, we can omit `==True`

In [47]:

```
if lunchtime or work_time:
    print('at the lab')
else:
    print('at home')
```

```
at the lab
```

Example 3:
Create three variables with numerical values.
Create a program that prints `'found'` if **any** of the variables are greater than 10.

Hint: In a conditional statement, non-zero values are treated as `True`, zero is treated as `False`.

In [36]:

```
A = 2
B = 2
C = 4

if A>10 or B>10 or C>10:
    print('found')
```

Notice that the following code **won't** work:

```
A = 2
B = 3
C = 4

if A or B or C>10:
    print('found')
```

This is because, in a conditional statement, *non-zero values* are treated as `True`, zero is treated as `False`.

A and B are non-zero numbers (2 and 3) so `A` is `True`, `B` is `True` and C>10 is `False`.

The `or` operators make the output of the condition `True`.

Note: All strings, with the exception of empty strings `''`, are treated as True.