Introduction to Computer Programming

2.2 User input & nested conditionals

`input`

Accepts typed input from the user.

Outputs the typed input as string data.

The function argument is a string, which is the prompt displayed to the user.

In [4]:

```
name = input("Enter your name: ") # type response when prompted and

print('My name is', name)
```

```
Enter your name: Ercan
My name is Ercan
```

This is a quick and easy way to add dynamic input to your program.

Input - a word of warning!

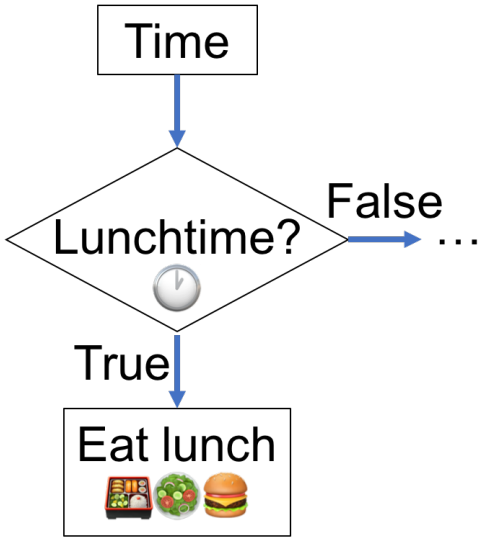
The input by the user is stored as a string.
Numbers entered will behave as text data unless converted to a numerical data type.
Example: `+` will join string data

In [7]:

```
A = input("Enter a number ")
B = input("Enter another number ")

print(A + B)

print(int(A) + int(B))
```

```
Enter a number 2
Enter another number 3
23
5
```

Example: Write a program that requests the time from the user and tells them to eat lunch if it is lunchtime.



In [9]:

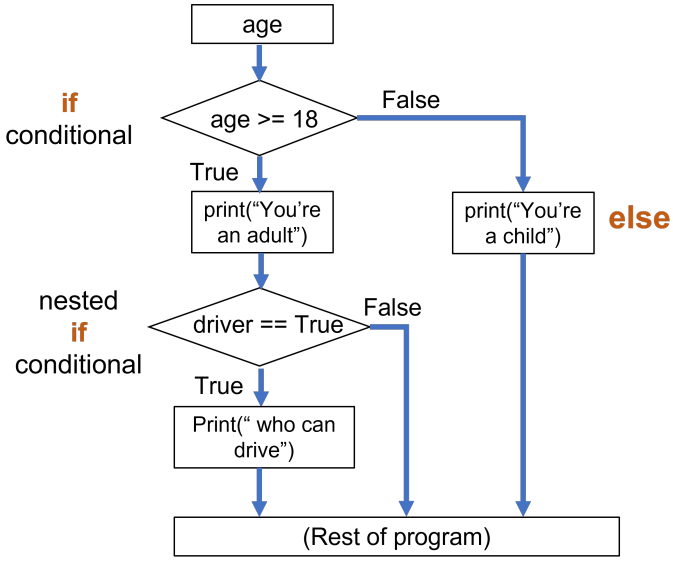
```
t = input('enter the time (24 hour clock) in format hh.mm : ')
t = float(t)    # convert string to float

# ----------------- Program from last week --------------------
Ls = 13.00      # lunch starts
Le = 14.00      # lunch ends

lunchtime = t >= Ls and t < Le
# -------------------------------------------------------------

if lunchtime:
    print("Eat lunch")
```

```
enter the time (24 hour clock) in format hh.mm : 9.00
```

Nested conditional statements

Conditional statements can be nested (a conditional statement within a conditional statement) to execute more complex decision making in a program.

Example: Translate the flow diagram into a programme



In [9]:

```
age = 17
driver = False

# Check if person is 18 years or older
if age >= 18:
    print("You're an adult", end='')

    if driver:
        print(' who can drive')
else:
    print("You're a child")
```

```
You're a child
```

Summary

- `input` : accepts typed input from the user.
- `input` outputs the typed input as string data!
- Conditional statements can be nested to create more complex decision-making within a program.

In-class Demos

Example 1: Write a program that asks for the user's age and checks if they are 18 or over.

In [ ]:

```

```

Example 2: Write a program that:

- asks the user to input a number
- checks if a number is odd or even
- checks if input even numbers are multiples of 4
- checks if input odd numbers are multiples of 3

In [14]:

```
Input a number 4
N is even and multiple of 4
```