Introduction to Computer Programming

2.2 User input & nested conditionals

University of BRISTOL

`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 [14]:

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

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

```
Enter your name: Hemma
My name is Hemma
```

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 [15]:

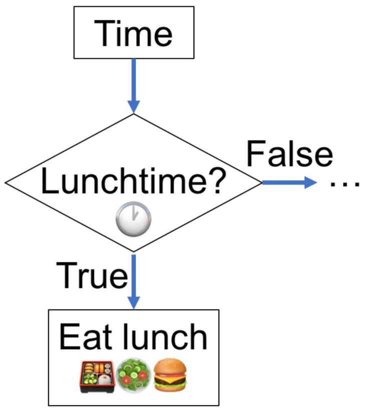
```python
A = input("Enter a number ")
B = input("Enter another number ")

print(A + B)

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

```
Enter a number 4
Enter another number 5
45
9
```

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



In [32]:

```
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

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

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

```
enter the time (24 hour clock) in format hh.mm : 13.00
Eat lunch
```

Notice, this has the same meaning as:

```
    if is_lunchtime == True:
        print("Eat lunch")
```

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

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 program

<center>
  <img src="img/adult_child.png" alt="Drawing" style="width: 300px;"/>
</center>
```

In [20]:

```
age = input('Enter your age: ')
driver = input('Can you drive? (True/False) ')


if age>=18:
    print("you're an adult")

    if driver:
        print(" who can drive")

else:
    print("you're a child")
```

```
you're an adult
```

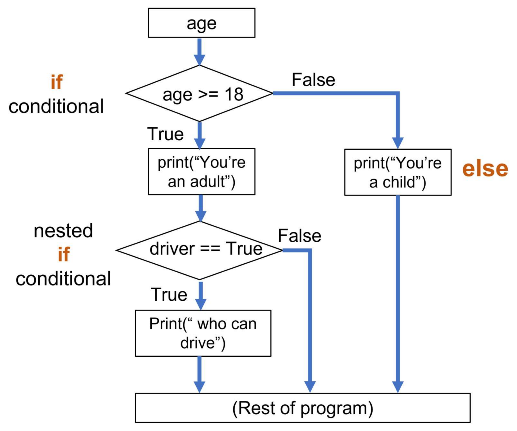
Summary

- `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:

Again, translate the flow diagram into a program.
This time, the program should accept the users age, and ability to drive (if they are an adult) using `input` .



In [31]:

```
age = input('enter your age: ')
age = int(age)

if age>=18:
    status = "adult"
    driver = input('can you drive? (True/False) ') # String 'True' cannot be converted to b

    if driver == 'True':                         # nested if
        status = status + " who can drive"

    print(status)

else:
    print("child")
```

```
enter your age: 18
can you drive? (True/False) True
adult who can drive
```

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

Hint: In a conditional statement, non-zero values are treated as `True` , zero is treated as `False` .
i.e. We can write shorter, neater code by omitting `==True`

In [25]:

```
N = input('Enter a number: ')
N = int(N)

if N % 2 :
    print('odd', end='')

    if not N % 3:
        print(' and multiple of 3')

else:
    print('even')

    if not N % 4 :
        print(' and multiple of 4')
```

```
Enter a number: 3
odd and multiple of 3
```