# Python summary

## Variables

- Variables are containers that you can use whenever you want
- It behaves as the value it contains

Ex: my_name = Anis - age = 20

- The variable my_name will behave as the valus “ Anis “

Using the print function
=> print(my_name)
=> Anis
It prints the value of the variable not the variable itself
Note: if you add quotations to the variable name it will act as a string literal not a variable. See next example.
=>print(“my_name”)
=>my_name
What happened here is that the print functions saw the quotations and knew that this is not a variable but a string literal and printed it as it is.

### Ways of printing variables

=> age = 21
=> players = 2

**1.string concatenation**

=> print(“My age is “ + str(age) + “ years old.”)
=> My ages is 21 years old.
=> print(“There are “ + str(players) + “ online”)
=> There are 2 players online

- String concatenation adds string literals to each others but there are some things you need to pay attention to while doing it

- When you use it to add a variable of type int or float you need to use typecasting to convert it to a string like this: str(age)
- It just add the variable to the end of the string so you if you need a spacing between them you need to put it yourself

**2.as arguments**

=> print(“My age is”, age, “years old.”)
=> My age is 21 years old.

- Printing the variable as an argument puts a space between every argument and you don’t need to typecasting as in string conatentation way

**3 f strings**

=> print(f”My age is {age} years old.”)
=> My age is 21 years old.

- Using the f string any variabel between curly braces is considered as a variable not a string literal and then uses the value stored in ti
- f  strings is preferred more than than the other two ways

# Data types

## Integer

Integer is represented as a whole number with no decimals.
Ex: 20, 21, 100, 5987
All these numbers are integers
Ex:
=> days = 7
=> print(f”there are {days} days in the week”)
=> there are 7 days in the week

## Float

Float is numbers that has a decimal portion in it.

Ex: 3.14, 27.8
=> price = 899.99
=> gpa = 4.0001
=> print(f"your gpa is not {gpa} as the maximum is 4 you cannot exceed it")
=>your gpa is not 4.0001 as the maximum is 4 you cannot exceed it

### Strings

Strings is just a seres of characters and it has to be between quotations marks
=> name = "mo salah"
=> email = "Idontcareatall@gmail.com" IT'S NOT A REAL EMAIL
=> print(name)
=> mo salah

### Boolean

=> alive = False
=> online = True
=> print(f"is he alive? {alive}")
=> is he a live? False

- Boolean values only evaluate to true or false
- We don't usually use them as in the example above but more in a conditioning way as in if statements. We will get through this later.
- Be careful that you don't put it between quotations nor write the frist letter small

## Tips & tricks for variables

### Multiple assignment

If you want to assign for example three variables at the same time
Traditional way:
X = 1
Y = 2

Z = 3
But with multiple assignment you can do this
X, Y, Z = 1, 2, 3
Easy ha

### Multiple variables same value

If you want to set the same value to some variables
Traditional way:
X = 1
Y = 1
Z = 1
Instead you can do it this way
X = Y = Z = 1
In this way all variables are assigned the same value 1

# Type casting

### Explicit

(string, int, float, bool)
=> name = “Amr”
=> age = 22
=> gpa = 2.9
=> chairman = True

- From string to bool —> bool(name) —> True

If name is empty it will result in False

- From int to float —> float(age) —> 22.0
- From float to int —> int(gpa) —-> 2
- From int to bool —> bool(age) —> True

If the number is anything but zero will evaluate to true otherwise False

### Implicit

=> x = 2
=> y = 1.0
=> x = x / y
If we printed x it will result in 2.0 although it was integer before but because we divided it by a float it was converted implicitly.

# User input

=> input(“here you can type a promt message”)
=> var = input(“here you stored the input in a variable so you can use it in your program”)

- If you need  to do math on the input you have to typecast it into an int or a float, because the input is always a string

=> x = input(“enter a number”)
=> 8
=> x = int(x)
=> y = x + 1
y  = 9

- If you don’t use the typecasting in the later example it will result in an error

# Math

### Arithmetic operations

**Addition**:

=> friends = 0
=> friends = friends + 1
=> friends += 1 - [ same as the last line known as an augmented assignment operator]

**Subtraction**:

=> friends = friends - 2

=> friends -= 2

**Multiplication**:

=> friends = friends * 3
=> friends *= 3

**Division**:

=> friends = friends / 2
=> friends /= 2

**Exponents**:

=> friends = friends ** 2
=> friends **= 2

**Remainder**:

=> friends = friends % 3
=> friends %= 3

## Bulit-in functions

=> x = 3.14
=> y = -4
=> z = 5

- Round —> result = round(x) —> 3
- Abs —> result = abs(y) —> 4
- Power —> result = pow(4, 3) —> 64
- Max —> result = max(x, y, z) —> 5
- Min —> result = min(x, y, z) —> -4

## Math module

=> import math ⇒ at the top of the file
=> pi = math.pi → 3.14159265359
=> e = math.e → 2.71828182846
=> result = math.sqrt(9) → 9

=> result = math.ceil(9.1) → 10
=> result = math.floor(9.9) → 9

# If statements

- If (condition) if the condition evaluates to true then the code below the if statement will execute otherwise nothing will happen
- Else: it executes if the condition in the if statement evaluates to False
- Elif (condition): if the condition in the if statement is False it checks the next elif statement if exists.

```
=> age = 17
=> if age >= 18:
        Print (“you are an adult now”)
=> elif age < 0:
        Print (“you are not born yet”)
=> else
        print(“grow up”)
```

# Logical operators

=> and ⇒ checks if two or more conditions if true
=> or ⇒ checks if at least one condition is true
=> not ⇒ converts the boolean value, True if False, False if True

```
=> gpa = 1.9
=> if gpa > 3.5 && gpa <= 4.0:
        print (“you are an excellent student”)
=> elif gpa > 2.5 & 3.5 <=:
        Print (“ you are very good student”)
And you get the point
=> student = True, teacher = False
=> if student or teacher:
        Print (“he is allowed to be in campus”)
=> bored = not True → False
```

# String methods

- Len ⇒ len(“string”) → 6
- Find ⇒ name.find(“ “) → 3 - given name = “lol lol again” - the first occurrence
- rfind ⇒ last occurrence – both methods will return -1 if not found
- capitalize ⇒ name.capitalize() → Anis - given name = “anis”
- upper ⇒ name.upper → ANIS - given name = “Anis”
- lower ⇒ name.lower → anis - given name = “ANIS”
- isdigit ⇒ name.isdigit() → if it’s a digit will return true otherwise false
- isalpha ⇒ name.isalpha() → if it contains only alpha letters will return true otherwise false
- count ⇒ name.count(“-”) → return the number of the occurrence of a character in a string
- replace ⇒ name.count(“-”, “ “) → will replace any characters with whatever you want

# String indexing

- You can access elements of a string using [ ] operator
  [ start : end : step ]
  Ex: name = “anis”
  name[0] → a
  name[0:] → anis
  name[:] → anis
  name[-1] → s
  name[:: -1] → sina ( the last number is a step)

# Loops

While loops:
⇒ while True:
    Do something
That’s infinite loop

⇒ num = 1
While num <= 3:
    Do something
    Num += 1
Loop that runs 3 times

For loops:
For i in range(1, 5):
    Do something
This loop runs 5 times
For I in string:
    Do something
This loop iterate over a string
And you can iterate over a list too