Data Structures

1st section: Python Basics

Variables:

- we use variables to refer to the data's location

- Variables are created the moment you assign value to it:

```
x = 10
name = "Martin"
print(x)
print(name)
```

- Python is ***dynamically typed*** language:
  - There is no advance declaration associating an identifier with a particular data type.
  - It can change the type of the data during runtime

```
y = 9
y = "John"
print(y)
```

type() function is used to get the data type of the variable.

- Assign multiple vars at the same time:

```
a, b, c = 1,2,3
print(a, b, c)
```

Data Types:

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey" : "value" , "name" : "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating **True** or **False**

- String concatenation with numeric types requires conversion to strings using the str function
  - print("name: " + name + " age: " + str(age))
- print(f"Name: {name}, age: {age}, weight: {weight}, is a stuents? {is_student}")

Python Collections (Arrays):

- There are four collection data types in the Python programming language:
  - **List** is a collection which is ordered and changeable. Allows duplicate members.
  - **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
  - **Set** is a collection which is unordered, unchangeable, and unindexed. No duplicate members.
  - **Dictionary** is a collection which is ordered and changeable. No duplicate members.
- When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

List	Tuple	Set	Dict
[ ]	( )	{ }	{ key : value}
Ordered	Ordered	Not Ordered	Not Ordered
indexed	indexed	Access by item	Access by key
Mutable (Add, Delete, Edit)	Immutable (CANNOT Add, Delete, Edit)	Immutable datatypes only (cannot hold lists or dicts)	Key: must be immutable (num, str, tuple) Value: can be any datatype.
Items are NOT unique (can have duplicates)	Items are NOT unique (can have duplicates)	Items are UNIQUE (it deletes any duplicated items)	Key must be unique.

mylist = [1, 2, True, "A" , 3.5 , "A"]	mytuple = (1 , 3 , 8.6, False, "Hi" , 3)	myset = { 1, 2, 3, "One", 4.2 , True }	mydict = {
			"name" : "Ahmed",
			"age" : 20 ,
			"height" : 175.6
			}

Input:

Conversion:

- input() function coverts whatever the user input into string
  - An explicit casting is required

```
name = input("Enter your name: ")
print( "Hello " + name)
```

- Casting/conversion can be done by tying the data type constructor.
- Since data types in Python are Classes, using the constructors to set a specific data type and for conversion

```
num = int(input("Enter a number: "))
print( num + 1)
```

Operators:

Python operation	Arithmetic operator	Algebraic expression	Python expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	$b \cdot m$	`b * m`
Exponentiation	**	x^y	`x ** y`
True division	/	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Floor division	//	$\lfloor x/y \rfloor$ or $\left\lfloor \frac{x}{y} \right\rfloor$ or $\lfloor x \div y \rfloor$	`x // y`
Remainder (modulo)	%	$r \bmod s$	`r % s`

Exercise 1:

- 1) Write a Python program that prompts the user to enter name and his/her birth year, calculate the age based on the current year, then print his/her name and age.

  Welcome NAME you are AGE years old

  e.g.: Welcome Ahmed you are 30 years old

Exercise 2:

- 2) Write a Python program that calculate the area and perimeter of a circle its radius is given from the user.

$$A = \pi r^2$$
$$P = 2\pi r$$

Assignment Operators:

Operator	Example	Equivalent Expression (m=15)	Result
=	y = a+b	y = 10 + 20	30
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=10	m = m/10	1.5
%=	m %=10	m = m%10	5
**=	m**=2	m = m**2 or $m = m^2$	225
//=	m//=10	m = m//10	1

Comparison Operators:

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

Logical (Boolean) Operators:

Operation	Result
`x or y`	if *x* is false, then *y*, else *x*
`x and y`	if *x* is false, then *x*, else *y*
`not x`	if *x* is false, then `True`, else `False`

if … elif … else Statements:

```
if expression:
   statement(s)
else:
   statement(s)
```

```
if expression1:
   statement(s)
elif expression2:
   statement(s)
elif expression3:
   statement(s)
else:
   statement(s)
```

Exercise 3:

- 3) Write a Python program that asks the user to input two integers, then prints which of the two numbers is larger or if the numbers are equal.

Loops:

- While Loop:

```
while expression:
    statement(s)
```

```
while condition:
    # execute these statements
else:
    # execute these statements
```

```
a = 0

while a <= 10:
    print(a)
    a += 1
else:
    print("Loop terminates.")
```

- For Loops:

```
for iterator_var in sequence:
    statements(s)
```

```
myNumbers = [10, 20, 30, 40]

# To print all numbers in the list
for number in myNumbers:
    print(number)
```

Exercise 4:

- 4) Write a Python program that counts the number of even and odd numbers from the following series of numbers (1, 2, 3, 4, 5, 6, 7, 8, 9)

Expected Output :
Number of even numbers : 4
Number of odd numbers : 5

Thank You…

Eng. Alaa Abdulfattah