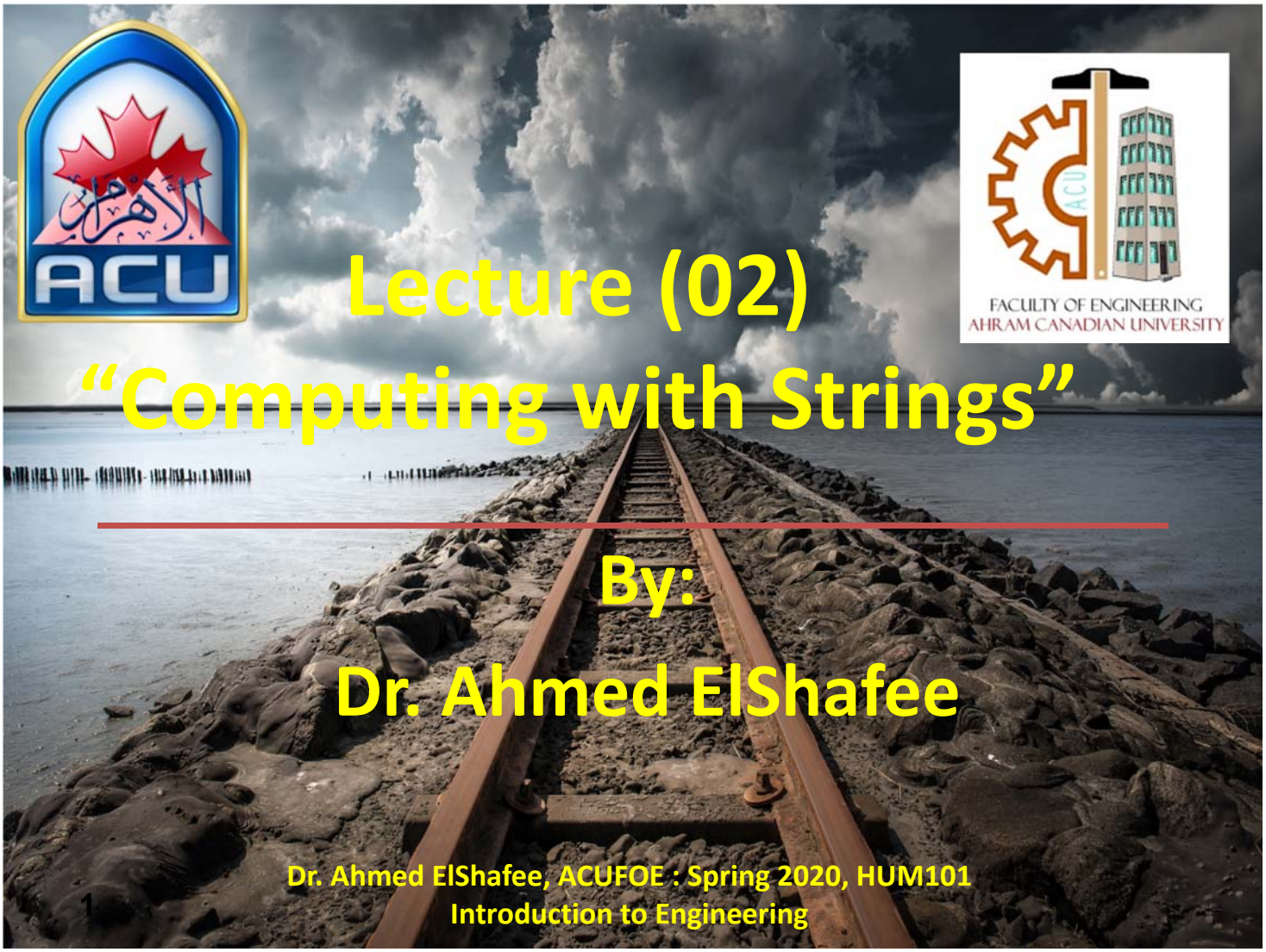


From last lecture,....

Assigning Input

- The purpose of an input statement is to get some information from the user
- **Example:**
  - x = input( )
  - x=int(input())
    - If the user enters 5 then x = 5
- Input with Prompt:
  - grade = input ("please enter a grade:")
- Note: the user can enter an expression
  - x = input("Please enter an expression:")
- If the user enters 3 + 2 then x = 3 + 2

Simultaneous Assignment

- Python allows assigning values to multiple variables in 1 statement
- **Examples:**

```
>>> x,y = 3,5
x = 3 and y = 5
>>> x1,y1 = input("please enter 2 numbers:").split()
20   30
X1 = 20 and y1 = 30
```

The Math Library

- Python provides many other useful mathematical functions in a special **math library**
- A **library** is just a file that contains some useful functions
- To use a library you need to **import** the file
- **Example:**

  from math import *
- After importing a library you can use all the functions defined inside

The Math Library

Python	Mathematics	English
`pi`	π	An approximation of pi.
`e`	e	An approximation of e.
`sin(x)`	$\sin x$	The sine of x.
`cos(x)`	$\cos x$	The cosine of x.
`tan(x)`	$\tan x$	The tangent of x.
`asin(x)`	$\arcsin x$	The inverse of sine x.
`acos(x)`	$\arccos x$	The inverse of cosine x.
`atan(x)`	$\arctan x$	The inverse of tangent x.
`log(x)`	$\ln x$	The natural (base e) logarithm of x
`log10(x)`	$\log_{10} x$	The common (base 10) logarithm of x.
`exp(x)`	e^x	The exponential of x.
`ceil(x)`	$\lceil x \rceil$	The smallest whole number $>= x$
`floor(x)`	$\lfloor x \rfloor$	The largest whole number $<= x$

Using the Math Library

- **The square root function (sqrt)**

```
>>> x = sqrt(9)
x = 3.0
```

- **The sin function**

```
>>> x = sin(30)
-0.988031624092861 8
>>> x = radians(30)
>>> sin(x)
0.4999999999999994
```

- **pi variable**

```
>>> print(pi)
3.14159265359
```

Casting (Type Conversion)

- In Python you can convert a variable from a type to another
- To get the type of a variable use the type(var) function
- Example:

```
>>> x = 3
>>> type(x)
<type 'int'>
>>> x = float(x)
>>> type(x)
<type 'float'>
>>> str (1 + 3)
'4'
```

float(expr(	Convert expr to a floating point value
int(expr(	Convert expr to an integer value
str(expr(	Return a string representation of expr

Input From User

```
x=input('Enter any number')
Print(x+2)
>>>TypeError: must be str, not int

x=int(input('enter num1'))
print(x+2)
>>>4
```


This week,....

Outline

- **What is a String?**
- **String Operations**
  - Concatenation
  - Repetition
  - Indexing
  - Length
  - Slicing
- **String Printing**
  - Escape characters
  - Multiline string
  - Formatted printing

Outline - cont

- **String Input**
- **Casting**
- **Evaluating Strings**
- **String Representation – (Ord and chr)**
- **The String Library**

Why Strings?

- Many programs depend on string usage and manipulation
- An example problem:
  - Write a program to emulate an intelligent ChatBot…
- To solve a problem like that, you need to learn about strings

What is a String?

- A string is a **sequence of characters**
- Strings are delimited by single ‘ ‘ or double “ “ quotes
- Strings can be stored in variables
- **Example:**

```
>>> Str1 = "hello"
>>> Name = 'Ahmed'
>>> 'that'll not work'
SyntaxError: invalid syntax
>>> "that'll work"
```

What if you want to put both kinds of quote in one string?

String Operations: 1. Concatenation

- The **+** operator can be used to join 2 strings
- **Example:**

Note the space here

```
>>> print ('hello ' + "world")
'hello world'
>>> print ('1' + '2')
'12'
>>> Fname = 'Foo'
>>> print ('Welcome ' + Fname)
'Welcome Foo'
```

String Input

- Python provides input function called **input()**
- Example:

```
>>> name = input()
Ahmed
>>> print ('hello', name)
hello Ahmed
>>> expression = input("enter an expression: ")
enter an expression: 3 + 2 - 1
>>> print ('x = ' + expression)
x = 3 + 2 - 1
```

Simple ChatBot

- Believe it or not! That's all you need to know to solve the problem given at the beginning. Here is the code:

```
print ("Hi There...")
print ("What's your name?")
name = input("My name is: ")
print ("Nice to meet you " + name)
print ("How are you feeling today?")
feeling =input("I am feeling ")
print ("Glad you are feeling "+ feeling)
print ("It was nice meeting you " +
name+ ". Hope you have a very nice
day!")
print ("Have fun in HUM101 today ;-)")
```

String Operations: Repetition

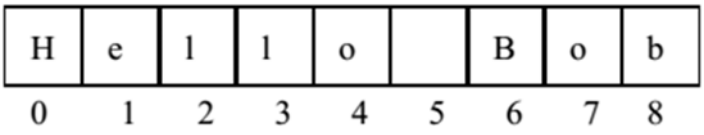
- Repetition builds a string by multiple **concatenations** of a string with itself
- We can repeat a string using the * operator
- **Example:**

```
>>> 'hi' * 3
'hihihi'
>>> '12' * 2
'1212'
```

String Operations: Indexing

- Indexing is used to access the individual characters that make up the string
- Characters in a string are indexed from **0** to the **length of the string - 1**
- **Example:**

```
>>> greet = 'Hello Bob'
>>>greet[1]
'e'
>>>greet[9]
IndexError: string index out of range
```



- **Note:** Strings are **immutable** i.e. you can not edit a string
- **Example:**

```
>>> greet[0] = 'p'
TypeError: 'str' object does not support item assignment
```

String Operations: Length

- Python provides a built-in function called **len** that returns the number of characters in a string
- **Example:**

```
>>> name = "Bob"
>>> len (name)
3
>>> len ('hello')
5
```

- To get the last letter of a string

```
>>> x = len(name)
>>> print (name[x-1])
'b'
```

Note the -1

String Operations: Slicing

- Slicing is used to access a substring from a string
- Slicing takes the form
  - string [start : end]

Including start and excluding end

- **Example:**

```
>>> country = 'Egypt'
>>> country [1:3]
'gy'
>>> country [2:]
'ypt'
>>> country [:4]
'Egyp'
>>> country [-3:4]
'yp'
```

-5	-4	-3	-2	-1
E	**g**	**y**	**p**	**t**
0	1	2	3	4

Note that the slice stops before the end index

String Printing
1. Escape Characters

- Escape characters are used inside a string to "escape" from Python's usual syntax rules for a moment
- **Example:**

```
>>> print ('she said, "that\'s hard to read"')
she said, "that's hard to read"
>>> print ("hello \n world")
hello
world
```

Escape Sequence	Description
\n	End of line
\\	Backslash
\'	Single quote
\"	Double quote
\t	Tab

String Printing
2. Multiline String

- If you create a string using single or double quotes, it must fit on a single line
- **Example:**

```
>>> 'one
EOL while scanning string literal
```

- To span multiple lines, put three single quotes or three double quotes
- **Example:**

```
>>> '''one
... two
... three'''
'one\ntwo\nthree'
```

String Printing
3. Formatted Printing

- To print the value of a variable with a string we can use

1. **Comma-separation**

```
>>> p = 3
>>> print ("price =", p, "$")
price = 3 $
```

Comma-separated strings will be separated by a space

2. **Concatenation**

```
>>> name = 'Ahmed'
>>> print ('welcome ' + name)
welcome Ahmed
>>> print( "price = " + p + "$")
TypeError: cannot concatenate 'str' and 'int'
```

Concatenation can only be used between 2 strings

String Printing
3. Formatted Printing

{:d}	Integer
{:f}	Float
{:s}	String

3. **String formatting**

```
>>> p = 5      >>> t = 34.5      >>> c = 'Egypt'
>>> print ("price = {:d}".format(p))
price = 5
>>> print ("temp in {:s} = {:f} degrees".format(c,t))
temp in Egypt = 34.500000 degrees
>>> print ("temp in {:s} = {:.2f} degrees".format(c,t))
temp in Egypt = 34.5 degrees
>>> print ("temp in {:s} = {:10.2f} degrees".format(c,t))
temp in Egypt =       34.5 degrees
```

Indicates precision

Indicates width

Casting (Type Conversion)

- In Python you can convert a variable from a type to another
- To get the type of a variable use the type(var) function
- **Example:**

```
>>> x = 3
>>> type(x)
<type 'int'>
```

- To convert x to a float use the float(var) function
- **Example:**

```
>>> x = float(x)
>>> type(x)
<type 'float'>
>>> str (1 + 3)
'4'
```

float (expr)	Convert expr to a floating point value
int (expr)	Convert expr to an integer value
str (expr)	Return a string representation of expr

Evaluating Strings

- Python provides a function **eval(expr)** used to evaluate a **string** as an **expression**
- **Example:**

```
>>> eval ('1 + 4 * 2')
9
>>> eval ('3.5 – 1')
2.5
>>> int ('1+5')
ValueError: invalid literal for int()
```

Note that int() is used for conversion not evaluation

String Representation

- Python provides built-in functions to switch between characters and their numeric codes
- The **ord()** function returns the numeric "ordinal" code of a single-character string
- **Example:**

```
>>> ord ("a")
97
>>> ord ("A")
65
```

Note that 'a' ≠ 'A'

- The **chr()** function is the opposite of **ord()**
- **Example:**

```
>>> chr(97)
 'a'
```

A common usage of **ord** and **chr** is in **Encryption**

The String Library

- Python provides a handful set of functions to manipulate strings available in the **string library**
- **Example:**

```
>>>from string import *
>>> s = 'hello world'
>>> s.capitalize ()
'Hello world'
>>> s.upper ()
'HELLO WORLD'
>>> s.split ()
['hello', 'world']
>>>s.split ( 'o')
['hell', ' w', 'rld']
```

The String Library (a sample)

Function	Meaning
`capitalize(s)`	Copy of `s` with only the first character capitalized
`capwords(s)`	Copy of `s` with first character of each word capitalized
`center(s, width)`	Center `s` in a field of given `width`
`count(s, sub)`	Count the number of occurrences of `sub` in `s`
`find(s, sub)`	Find the first position where `sub` occurs in `s`
`join(list)`	Concatenate `list` of strings into one large string
`ljust(s, width)`	Like `center`, but `s` is left-justified
`lower(s)`	Copy of `s` in all lowercase characters
`lstrip(s)`	Copy of `s` with leading whitespace removed
`replace(s,oldsub,newsub)`	Replace all occurrences of `oldsub` in `s` with `newsub`
`rfind(s, sub)`	Like `find`, but returns the rightmost position
`rjust(s,width)`	Like `center`, but `s` is right-justified
`rstrip(s)`	Copy of `s` with trailing whitespace removed
`split(s)`	Split `s` into a list of substrings (see text).
`upper(s)`	Copy of `s` with all characters converted to upper case

Exercise

- s1 = ‘welcome to’ s2 = ‘HUM’ s3 = ‘101’
- Show the result of each of the following:
  1. print (s1 + s2, s3)
  2. print (s3*3)
  3. print (int(s3)*3)
  4. print (s1[2:5])
  5. print (s1.upper())
  6. print (s2.split(’s’))
  7. print (s1+’\\’+s2+’\t’+s3)