Introduction to Python

By: Mohammed Al-Qassabi
Head of Robotics & AI Department at
Qatar Scientific Club

m.alqassabi@qsc.org.qa

Day 1:

Introduction to Programming

Print function
Data types (str, int, float)
Variables
Comment
Turtle Library
Defining a function
Calling a function

Mohammed Al-Qassabi
Head of Robotics & AI Department
Qatar Scientific Club

```python
import turtle

def intro():
    print("Hello World")
    print(Hello World) # This should give an error because the quotation marks are
missing
    print(4)
    x = 4
    print(x)
    print("x")
    myStr = "String"
    myInt = 10
    myFlt = 3.14
    """ Hello World! """
    4 = x # this should give an error because you cant name a variable with an integer
    # while = 8
    name1 = "Mohammed"
    name2 = Mohammed
    print(name1, name2)
    mohammed = 14
    name2 = mohammed
    print(name2)
    num = 1 + 1
    print(num)
    print(1+1)
    print("1+1")
    myName = "Mohammed"
    print(myName[2])
    print(myName[1:4])
    print(len(myName))

def normal_square():
    turtle.fd(50)
    turtle.right(90)
    turtle.fd(50)
    turtle.right(90)
    turtle.fd(50)
    turtle.right(90)
    turtle.fd(50)
    turtle.right(90)

def normal_reqtangle():
    turtle.fd(50)
    turtle.right(90)
    turtle.fd(80)
```

```
    turtle.right(90)
    turtle.fd(50)
    turtle.right(90)
    turtle.fd(80)
    turtle.right(90)

def my_circle():
    turtle.circle(120)

def my_triangle():
        turtle.right(60)
        turtle.fd(100)
        turtle.right(120)
        turtle.fd(100)
        turtle.right(120)
        turtle.fd(100)

def loop_square():
    for i in range(4):
        turtle.fd(50)
        turtle.right(90)

def loop_rectangle():
    for i in range(2):
        turtle.fd(50)
        turtle.right(90)
        turtle.fd(80)
        turtle.right(90)
# Introduction to Functions
def addition(x, y):
    return x + y
def subs(x, y):
        return x - y

def my_square(x):
    for i in range(4):
        turtle.fd(x)
        turtle.right(90)
def my_rectangle(x, y):
    if x == y:
        my_square(x)
    else:
        for i in range(2):
                turtle.fd(x)
                turtle.right(90)
                turtle.fd(y)
                turtle.right(90
```

Day 2:

Decision Structure & Boolean Logic

Input, processing, output
Perform Calculations
If statement
If-else statement
Comparing variables
Nested decision structure (if-elif-else)
If *not* operator

Mohammed Al-Qassabi
Head of Robotics & AI Department
Qatar Scientific Club

```
# Question: Write a function that takes x and y as input, and return their sum,
substraction, multiplication and division
def addition(x, y):
    return x + y
def subs(x, y):
    return x - y
def multiplication(x, y):
    return x * y
def division(x, y):
    return x / y

def testing_if():
    a = 10
    b = 20
    if a > b: # if a is greater than b
        print(a, " is greater")
    if b > a: # if b is greater than a
        print(b, " is greater")
    if a == b: # if the value of a is equal to the valuse of b
        print(a, " is equal to ", b)

def testing_elif():
    a = 10
    b = 20
    if a > b:
        print(a, " is greater")
    elif b > a:
        print(b, " is greater")
    else:
        print(a, " is equal to ", b)

def testing_else:
    a = 10
    b = 20
    if a >= b: # if a is greater than or equal to b
        print(a)
    else:
        print(b)

# Question: Write the function calculator(a, b, operator) that takes two integers a &
b and the operator for the calculation, and returns the value.
# Write a function that takes x and y as input, and return their sum, substraction,
multiplication and division
# But this time, add an additional paramater that declares the mathematical operation
(+,-,*,/) as string
def calculator(x, y, operator):
    if operator == "*":
```

```
        return x * y
    elif operator == "/":
        return x / y
    elif operator == "+":
        return x + y
    elif operator == "-":
        return x - y
print(calculator(10, 5, "*"))
print(calculator(10, 5, "/"))
print(calculator(10, 5, "+"))
print(calculator(10, 5, "-"))

x = "4"
print(type(x)) # check if the type of x is string or integer
y = int(x) # changing the type of x from string to an integer
print(y)

# Explain // and % operators
print(10//3) # prints the numbers before the decimal
print(10%3) # prints the numbers after the decimal point (the remainder)

def testing_ifor_ifand():
    a = 10
    b = 20
    if a > b or a == b:
        print(a)

def testing_ifnot():
    a = 10
    b = 20
    if not a == b:
        return False
    if a != b:
        return False

# Question 1: write a function that prints the greater number among a, b, and c
def greatestNum(a, b, c):
    if a >= b and a >= c: # if a is greater than or equal to b and c
        return(a)
    elif b >= a and b >= c: # if b is greater than or equal to a and c
        return(b)
    elif c >= a and c >= b: # if c is greater than or equal to b and a
        return(c)
print(greatestNum(10,20,30))

def testing_nested():
    x = 40
```

```
    if x > 10:
      print("Above ten,")
      if x > 20:
        print("and also above 20!")
      else:
        print("but not above 20.")


# Question 2: Positive, Negative, or Zero: Write a program that takes an integer as
input and prints whether it's positive, negative, or zero.
def main(n):
    if n > 0:
        return "Positive"
    elif n < 0:
        return "Negative"
    else:
        return "Zero"
print(main(0))
print(main(-1))
print(main(1))

# Introducing Lists []

def listIntro():
    myList = [1, 2, 3]
    print(myList)
    myList = ["a", "b", "c"]
    print(myList)
    myList = ["a", 2, 3.14]
    print(myList)
    myList.append("hi") # adding 'hi' to the list
    print(myList)

    newList = []
    for i in range(5):
        newList.append(i) # adding the numbers from 0 to 4 to the list
    print(newList)


# Question 3: Even or Odd: Write a program that takes an integer as input and prints
whether it's even or odd.
def isEven(n):
    if n > 1 and n%2==0:
        return True
    else:
        return False
print(isEven(2))
```

```python
print(isEven(3))


# Question 4: Vowel or Consonant: Write a program that takes a character as input and
determines if it's a vowel or a consonant.
def isVowel(letter):
    v = ["a","e","i","o","u"] # a list of vowels
    if letter in v: # check if the letter is in the list.
        return True # If the letter is in the list, then it is a vowel
    return False
print(isVowel("a"))
print(isVowel("b"))


# Question 5: Temperature Converter: Write a program that converts temperatures
between Celsius and Fahrenheit. The program should take an input temperature and its
unit (C or F) and convert it to the other unit. (Celsius to Fahrenheit: F = (C * 9/5)
+ 32, Fahrenheit to Celsius: C = (F - 32) * 5/9)
def tempConverter(temp, unit):
    if type(temp) == int:
        if unit == "c" or unit == "C": # if the unit is celsius
            newTemp = (temp * 9/5) + 32
            print(newTemp, "F")
        elif unit == "f" or unit == "F": # if the unit is fahrenheit
            newTemp = (temp - 32) * 5/9
            print(newTemp, "C")
    else:
        print("ERROR!")
tempConverter(50, "c")


# BONUS Question:

# Question 6: Age Classifier
"""
Age Classifier: Write a program that takes a person's age as input and classifies them
into the following categories:
Child: 0 to 12 years
Teenager: 13 to 19 years
Adult: 20 to 59 years
Senior: 60 years and above
"""
```

Day 3:

Repetition Structure and Lists

For loop
While loop
Nested loop
Lists
Problem solving

Mohammed Al-Qassabi
Head of Robotics & AI Department
Qatar Scientific Club

```
def testing_forloop():
    for i in range(10):
        print(i)

    for x in range(2, 6):
      print(x)

    for i in ragne(0, 50, 10): # Increase 10 every time i iterates
        print(i)

    for i in range(10, 1, -1): # from 10 to 1
        print(i)

    fruit = "banana"
    print(fruit)
    print(len(fruit)) # prints the number of letters
    for i in range(len(fruit)):
        print(fruit[i]) # prints every letter in the word 'banana'
    for x in "banana":
        print(x) # prints every letter in the word 'banana'

    fruits = ["apple", "banana", "cherry"]
    print(fruits)
    print(len(fruits)) # prints the number of items in the list (3)
    for i in range(len(fruits)):
        print(fruits[i]) # prints every item in the list
    for x in fruits:
        print(x) # prints every item in the list

    value = 0
    for i in ragne(5):
        value = value + i # add all the values from 0 to 4
    print(value)

    s = ""
    for i in "apple":
        s = s + i # adds every letter in the word 'apple' to the variable s
    print(s)

    for i in range(10, 0, -1):
        print(i)

    for x in range(2, 30, 3):
          print(x)
```

```
    for x in range(6):
        print(x)
    else:
        print("Finally finished!")

# Question 1: Given a string input, print the vowels in the str
def myVowels(s):
    v = ["a","e","i","o","u"]
    ans = []
    for i in s: # iterates over evey letter in the word s
        if i in v and i not in ans: # checks if the letter is in the list of vowels
and is not in the list ans
            ans.append(i) # if letter is vowel, it will added to list and
    return ans
print(myVowels("hi my name is Mohammed"))

# Question 2: Given a string input, print it backwords
def backwords(s):
    ans = ""
    for i in range(len(s)-1, -1, -1): # goes over the word letter by letter backwords
        ans = ans + s[i]
    print(ans)
backwords("Hello World")


def testing_whileloop(): # while loops are used mostly for conditions
    i = 1
    while i < 6: # as long as i is less than 6, keep printing i
        print(i)
        i = i + 1

    i = 1
    while i < 6:
        print(i)
        i = i + 1
    else:
        print("i is no longer less than 6")


# Number Guessing Game: Write a program that generates a random number between 1 and
100 and asks the user to guess it. The program should provide hints like "Too high" or
"Too low" until the user guesses the correct number.
def guessingGame():
    n = random.randint(1, 100)
    ans = int(input("Guess: "))
    while ans != n: # as long as the random number n is not equal to the guess ans,
keep asking the user to guess
```

```
        if ans < n:
            ans = int(input("Too Low! Guess again: "))
        elif ans > n:
            ans = int(input("Too High! Guess again: "))
    print("That's Correct! It's ", n)
guessingGame()


# use while loops when there is an indeterminate number of iterations
def leftmostDigit(n):
    while (n >= 10): # as long as the number is greater than 9 (2 digits), keep
dividing it by 10
        n = n//10
    return n
print(leftmostDigit(72658489290098)) # should return 7


# Question 3: Sum of Numbers: Write a program that takes a positive integer as input
and calculates the sum of all numbers from 1 to that integer.
def intSum(n):
    total = 0
    for i in range(n+1):
        total = total + i
    print(total)
intSum(5)


# Question 4: Find n's factor. 3! = 1x2x3 = 6
def nFactor(n):
    total = 1
    for i in range(1, n+1):
        total = total * i
    print(total)
nFactor(3)

# Question 5: Write a function that takes 2 integeres as inputs and return the sum of
the odd numbers between them
def sumOfOddsFromMToN(m, n):
    total = 0
    for x in range(m, n+1):
        if (x % 2 == 1):
            total += x
    return total
print(sumOfOddsFromMToN(4, 10))
```

```python
# Question 5: Write a function that takes 2 integeres as inputs and return the sum of
the odd numbers between them
# But use while loop this time
def sumOfOddsFromMToN(m, n):
    if m % 2 != 0: #check if m is odd number
        total = m
    else:
        total = m + 1 # the next odd number
    while m <= n:
        print(m, total)
        total = total + m
        m += 2
    return total - 1
print(sumOfOddsFromMToN(1,7))

def testing_nested():
    adj = ["red", "big", "tasty"]
    fruits = ["apple", "banana", "cherry"]
    for x in adj:
        for y in fruits:
            print(x, y)


# Question 6: Multiplication Table: Write a program that takes an integer as input and
prints the multiplication table for that number from 1 to 10.
def multiplicationTable(n):
    for i in range(11):
        print(n, " * ", i, " = ", n*i)
multiplicationTable(5)


# Question 7: Palindrome Check: Write a program that takes a string as input and
checks if it's a palindrome (reads the same forwards and backwards).
def isPalindrome(s):
    word = ""
    for i in range(len(s)-1, -1, -1):
        word = word + s[i]
    if word == s:
        return True
    else:
        return False
print(isPalindrome("hala"))
print(isPalindrome("wow"))
```

Day 4:

Python Marathon!

Solve as much questions as possible

Mohammed Al-Qassabi
Head of Robotics & AI Department
Qatar Scientific Club

1. **isEvenPositiveInt(n)** [5 pts]
   Write the function isEvenPositiveInt(n) which, given a value n, returns True if it is even, positive, and an integer, and False otherwise.

2. **Calculator**: Write a simple calculator program that takes two numbers and an operator (+, -, *, /) as input and performs the corresponding calculation [5 pts]

3. **Grade Classifier**: Write a program that takes a percentage as input and prints the corresponding grade according to the following criteria: [10 pts]
   1. 90% or above: A
   2. 80% to 89%: B
   3. 70% to 79%: C
   4. 60% to 69%: D
   5. Below 60%: F

4. **numberLength [10 pts]**
   Write the function numberLength(x) that takes a non-negative integer x and returns the number of digits in x. For example, numberLength(12) would return 2.

5. **threeLinesArea(m1, b1, m2, b2, m3, b3) and helpers** [30 pts]
   Write the function threeLinesArea(m1, b1, m2, b2, m3, b3) that takes six int or float values representing the 3 lines:

```
y = m1*x + b1
y = m2*x + b2
y = m3*x + b3
```

   First find where each pair of lines intersects, then return the area of the triangle formed by connecting these three points of intersection. If no such triangle exists (if any two of the lines are parallel), return 0.

   To do this, you must write three helper functions:

1. **lineIntersection(m1, b1, m2, b2)** to find where two lines intersect (which you will call three times)
    1. This function takes four int or float values representing two lines and returns the x value of the point of intersection of the two lines. If the lines are parallel, or identical, the function should return None.
    2. Hint: the equation for line intersection is = (b2 - b1) / (m2 - m1)
    3. Hint: lineIntersection(3,-5,1,5) should return 5
2. **distance(x1, y1, x2, y2)** to find the distance between two points (again called three times)
    1. This function takes four int or float values representing two points and returns the distance between those points.
    2. Hint: distance(20, 20, 23, 24) should return 5
3. **triangleArea(s1, s2, s3)** to find the area of a triangle given its side lengths (which you will call once).
    1. This function takes three int or float values representing side lengths of a triangle, and returns the area of that triangle. To do this, you may wish to to use Heron's Formula.
    2. Hint: triangleArea(3,4,5) should return 6

You may write other helper functions if you think they would be useful, but you must at least write these three exactly as described, and then you must use them appropriately in your solution. Once you have written and tested your helper functions, then move on to writing your threeLinesArea function, which of course should use your helper functions. That's the whole point of helper functions. They help!

Note that helper functions help in several ways. First, they are logically **simpler**; they break down your logic into smaller chunks that are easier to reason over. Second, they are independently **testable**, so you can more easily isolate and fix bugs. And third, they are **reusable**, so you can use them as helper functions for other functions in the future. All good things!

6. **getKthDigit(n, k)** [10 pts]
   Write the function getKthDigit(n, k) that takes a possibly-negative int n and a non-negative int k, and returns the kth digit of n, starting from 0, counting from the right. So:

```
getKthDigit(789, 0) == 9
getKthDigit(789, 1) == 8
getKthDigit(789, 2) == 7
getKthDigit(789, 3) == 0
```

7. **containsOddDigits [10 pts]**
   Write the function containsOddDigits(x) that takes an integer, x, and returns True if it contains any odd digits, and False otherwise.
   Hint: In this and in future problems, you'll need to get individual digits from numbers. You may want to reuse your solution to getKthDigit and numberLength to do this.

8. **setKthDigit(n, k, d=0)** [15 pts]
   Write the function setKthDigit(n, k, d=0) that takes three integers -- n, k, and d -- where n is a possibly-negative int, k is a non-negative int, and d is a non-negative single digit (between 0 and 9 inclusive) with a default value of 0. This function returns the number n with the kth digit replaced with d. Counting starts at 0 and goes right-to-left, so the 0th digit is the rightmost digit. For example:

```
setKthDigit(468, 0, 1) == 461
setKthDigit(468, 1, 1) == 418
setKthDigit(468, 2, 1) == 168
setKthDigit(468, 3, 1) == 1468
```

```
setKthDigit(468, 1) == 408
```