THE VILLAGE
INTERNATIONAL SCHOOL
"We Nurture Dreams"

GRADE :12: COMPUTER SCIENCE-QUESTION BANK

Chapter 1:Python Revision Tour

Question 1

Which of the following is an invalid variable?

1. my_day_2
2. **2nd_day** ✓
3. Day_two
4. _2

Question 2

Which of the following is not a keyword?

1. **eval** ✓
2. assert
3. nonlocal
4. pass

Question 3

Which of the following cannot be a variable?

1._init_
2. **in** ✓
3. it
4. on

Question 4

Which of these is not a core data type?

1. Lists
2. Dictionary
3. Tuples
4. **Class** ✓

Question 5

How would you write x^y in Python as an expression ?

1. x^y
2. **x**y** ✓
3. x^^y
4. none of these

Question 6

What will be the value of the expression?
14 + 13 % 15

1. 14
2. **27** ✓
3. 12
4. 0

Question 7

Evaluate the expression given below if A = 16 and B = 15.
A % B // A

1. 0.0
2. **0** ✓
3. 1.0
4. 1

Question 8

What is the value of x?
x = int (13.25 + 4/2)

1. 17
2. 14
3. **15** ✓
4. 23

Question 9

The expression 8/4/2 will evaluate equivalent to which of the following expressions:

1. 8/(4/2)
2. **(8/4)/2** ✓

Question 10

Which among the following list of operators has the highest precedence?

+, -, **, %, /, <<, >>, |

1. <<, >>
2. ****** ✓
3. I
4. %

Question 11

Which of the following expressions results in an error?

1. float('12')
2. int('12')
3. float('12.5')
4. **int('12.5')** ✓

Question 12

Which of the following statement prints the shown output below?
hello\example\test.txt

1. print("hello\example\test.txt")
2. **print("hello\\example\\test.txt")** ✓
3. print("hello\"example\"test.txt")
4. print("hello"\example"\test.txt")

Question 13

Which value type does input() return ?

1. Boolean
2. **String** ✓
3. Int
4. Float

Question 14

Which two operators can be used on numeric values in Python?

1. @
2. **%** ✓
3. **+** ✓
4. #

Question 15

Which of the following four code fragments will yield following output?

```
Eina
Mina
Dika
```

Select all of the function calls that result in this output

1. print('''Eina\nMina\nDika''')
2. print('''EinaMinaDika''')
3. **print('Eina\nMina\nDika')** ✓
4. print('EinaMinaDika')

Question 16

Which of the following is valid arithmetic operator in Python :

1. **//** ✓
2. ?
3. <
4. and

Fill in the Blanks

Question 1

The smallest individual unit in a program is known as a ***token***.

Question 2

A token is also called a ***lexical unit***.

Question 3

A ***keyword*** is a word having special meaning and role as specified by programming language.

Question 4

The data types whose values cannot be changed in place are called ***immutable*** types.

Question 5

In a Python expression, when conversion of a value's data type is done automatically by the compiler without programmer's intervention, it is called ***implicit type conversion***.

Question 6

The explicit conversion of an operand to a specific type is called ***type casting***.

Question 7

The ***pass*** statement is an empty statement in Python.

Question 8

A ***break*** statement skips the rest of the loop and jumps over to the statement following the loop.

Question 9

The ***continue*** statement skips the rest of the loop statements and causes the next iteration of the loop to take place.

Question 10

Python's ***keywords*** cannot be used as variable name.

True/False Questions

Question 1

The expression int(x) implies that the variable x is converted to integer.
True

Question 2

The value of the expressions 4/(3*(2 - 1)) and 4/3*(2 - 1) is the same.
True

Question 3

The value of the expressions 4/(3*(4 - 2)) and 4/3*(4 - 2) is the same.
False

Question 4

The expression 2**2**3 is evaluated as: (2**2)**3.
False

Question 5

A string can be surrounded by three sets of single quotation marks or by three sets of double quotation marks.
True

Question 6

Variables can be assigned only once.
False

Question 7

In Python, a variable is a placeholder for data.
False

Question 8

In Python, only if statement has else clause.
False

Question 9

Python loops can also have else clause.
True

Question 10

In a nested loop, a break statement terminates all the nested loops in one go.
False

Type A: Short Answer Questions/Conceptual Questions

Question 1

What are tokens in Python? How many types of tokens are allowed in Python? Exemplify your answer.

Answer

The smallest individual unit in a program is known as a Token. Python has following tokens:

1. **Keywords** — Examples are import, for, in, while, etc.
2. **Identifiers** — Examples are MyFile, _DS, DATE_9_7_77, etc.
3. **Literals** — Examples are "abc", 5, 28.5, etc.
4. **Operators** — Examples are +, -, >, or, etc.
5. **Punctuators** — ' " # () etc.

Question 2

How are keywords different from identifiers?

Answer

Keywords are reserved words carrying special meaning and purpose to the language compiler/interpreter. For example, if, elif, etc. are keywords. Identifiers are user defined names for different parts of the program like variables, objects, classes, functions, etc. Identifiers are not reserved. They can have letters, digits and underscore. They must begin with either a letter or underscore. For example, _chk, chess, trail, etc.

Question 3

What are literals in Python? How many types of literals are allowed in Python?

Answer

Literals are data items that have a fixed value. The different types of literals allowed in Python are:

1. String literals
2. Numeric literals
3. Boolean literals
4. Special literal None
5. Literal collections

Question 4

Can nongraphic characters be used and processed in Python? How? Give examples to support your answer.

Answer

Yes, nongraphic characters can be used in Python with the help of escape sequences. For example, backspace is represented as \b, tab is represented as \t, carriage return is represented as \r.

Question 5

Out of the following, find those identifiers, which cannot be used for naming Variables or Functions in a Python program:

- Price*Qty
- class
- For
- do
- 4thCol
- totally
- Row31
- _Amount

Answer

- Price*Qty ⇒ Contains special character *
- class ⇒ It is a keyword
- 4thCol ⇒ Begins with a digit

Question 6

How are floating constants represented in Python? Give examples to support your answer.

Answer

Floating constants are represented in Python in two forms — Fractional Form and Exponent form. Examples:

1. **Fractional Form** — 2.0, 17.5, -13.0, -0.00625
2. **Exponent form** — 152E05, 1.52E07, 0.152E08, -0.172E-3

Question 7

How are string-literals represented and implemented in Python?

Answer

A string-literal is represented as a sequence of characters surrounded by quotes (single, double or triple quotes). String-literals in Python are implemented using Unicode.

Question 8

What are operators ? What is their function? Give examples of some unary and binary operators.

Answer

Operators are tokens that trigger some computation/action when applied to variables and other objects in an expression. Unary plus (+), Unary minus (-), Bitwise complement (~), Logical negation (not) are a few examples of unary operators. Examples of binary operators are Addition (+), Subtraction (-), Multiplication (*), Division (/).

Question 9

What is an expression and a statement?

Answer

An expression is any legal combination of symbols that represents a value. For example, 2.9, a + 5, (3 + 5) / 4.
A statement is a programming instruction that does something i.e. some action takes place. For example:
print("Hello")
a = 15
b = a - 10

Question 10

What all components can a Python program contain?

Answer

A Python program can contain various components like expressions, statements, comments, functions, blocks and indentation.

Question 11

What are variables? How are they important for a program?

Answer

Variables are named labels whose values can be used and processed during program run. Variables are important for a program because they enable a program to process different sets of data.

Question 12

Describe the concepts of block and body. What is indentation and how is it related to block and body?

Answer

A block in Python, represents a group of statements executed as a single unit. Python uses indentation to create blocks of code. Statements at same indentation level are part of same block/suite and constitute the body of the block.

Question 13

What are data types? How are they important?

Answer

Data types are used to identify the type of data a memory location can hold and the associated operations of handling it. The data that we deal with in our programs can be of many types like character, integer, real number, string, boolean, etc. hence programming languages including Python provide ways and facilities to handle all these different types of data through data types. The data types define the capabilities to handle a specific type of data such as memory space it allocates to hold a certain type of data and the range of values supported for a given data type, etc.

Question 14

How many integer types are supported by Python? Name them.

Answer

Two integer types are supported by Python. They are:

1. Integers (signed)
2. Booleans

Question 15

What are immutable and mutable types? List immutable and mutable types of Python.

Answer

Mutable types are those whose values can be changed in place whereas Immutable types are those that can never change their value in place.

Mutable types in Python are:

1. Lists
2. Dictionaries
3. Sets

Immutable types in Python are:

1. Integers
2. Floating-Point numbers
3. Booleans
4. Strings
5. Tuples

Question 16

What is the difference between implicit type conversion and explicit type conversion?

Answer

Implicit Type Conversion	Explicit Type Conversion
An implicit type conversion is automatically performed by the compiler when differing data types are intermixed in an expression.	An explicit type conversion is user-defined conversion that forces an expression to be of specific type.
An implicit type conversion is performed without programmer's intervention.	An explicit type conversion is specified explicitly by the programmer.
Example: a, b = 5, 25.5 c = a + b	Example: a, b = 5, 25.5 c = int(a + b)

Question 17

An immutable data type is one that cannot change after being created. Give three reasons to use immutable data.

Answer

Three reasons to use immutable data types are:

1. Immutable data types increase the efficiency of the program as they are quicker to access than mutable data types.
2. Immutable data types helps in efficient use of memory storage as different variables containing the same value can point to the same memory location. Immutability guarantees that contents of the memory location will not change.
3. Immutable data types are thread-safe so they make it easier to parallelize the program through multi-threading.

Question 18

What is entry controlled loop? Which loop is entry controlled loop in Python?

Answer

An entry-controlled loop checks the condition at the time of entry. Only if the condition is true, the program control enters the body of the loop. In Python, for and while loops are entry-controlled loops.

Question 19

Explain the use of the pass statement. Illustrate it with an example.

Answer

The pass statement of Python is a do nothing statement i.e. empty statement or null operation statement. It is useful in scenarios where syntax of the language requires the presence of a statement but the logic of the program does not. For example,

```
for i in range(10):
    if i == 2:
        pass
    else:
        print("i =", i)
```

Question 20

Below are seven segments of code, each with a part coloured. Indicate the data type of each coloured part by choosing the correct type of data from the following type.

(a) int
(b) float
(c) bool
(d) str
(e) function
(f) list of int
(g) list of str

(i)
```
if temp < 32 :
    print ("Freezing")
```

(ii)
```
L = ['Hiya', 'Zoya', 'Preet']
   print(L[1])
```

(iii)
```
M = []
for i in range (3) :
    M.append(i)
    print(M)
```

(iv)
```
L = ['Hiya', 'Zoya', 'Preet']
n = len(L)
if 'Donald' in L[1 : n] :
     print(L)
```

(v)
```
if n % 2 == 0 :
   print("Freezing")
```

(vi)

```
L = inputline.split()
while L != ( ) :
    print(L)
    L = L[1 :]
```
(vii)

```
L = ['Hiya', 'Zoya', 'Preet']
print(L[0] + L[1])
```
Answer

(i) bool
(ii) str
(iii) list of int
(iv) int
(v) bool
(vi) list of str
(vii) str

Type B: Application Based Questions

Question 1

Fill in the missing lines of code in the following code. The code reads in a limit amount and a list of prices and prints the largest price that is less than the limit. You can assume that all prices and the limit are positive numbers. When a price 0 is entered the program terminates and prints the largest price that is less than the limit.

```
#Read the limit
limit = float(input("Enter the limit"))
max_price = 0
# Read the next price
next_price = float(input("Enter a price or 0 to stop:"))
while next_price > 0 :
    <write your code here>
    #Read the next price
    <write your code here>
if max_price > 0:
    <write your code here>
else :
    <write your code here>
```
Answer

```
#Read the limit
limit = float(input("Enter the limit"))
max_price = 0
# Read the next price
next_price = float(input("Enter a price or 0 to stop:"))
while next_price > 0 :
    if next_price < limit and next_price > max_price:
        max_price = next_price
    #Read the next price
    next_price = float(input("Enter a price or 0 to stop:"))
```

```
if max_price > 0:
    print("Largest Price =", max_price)
else :
    print("Prices exceed limit of", limit);
```

Question 2a

Predict the output of the following code fragments:

```
count = 0
while count < 10:
    print ("Hello")
    count += 1
```

Answer

Output

```
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
Hello
```

Question 2b

Predict the output of the following code fragments:

```
x = 10
y = 0
while x > y:
   print (x, y)
   x = x - 1
   y = y + 1
```

Answer

Output

```
10 0
9 1
8 2
7 3
6 4
```

Explanation

X	y	Output	Remarks

X	y	Output	Remarks
10	0	10 0	1st Iteration
9	1	10 0 9 1	2nd Iteration
8	2	10 0 9 1 8 2	3rd Iteration
7	3	10 0 9 1 8 2 7 3	4th Iteration
6	4	10 0 9 1 8 2 7 3 6 4	5th Iteration

Question 2c

Predict the output of the following code fragments:

```
keepgoing = True
x=100
while keepgoing :
    print (x)
    x = x - 10
    if x < 50 :
       keepgoing = False
```

Answer

Output

```
100
90
80
70
60
50
```

Explanation

Inside while loop, the line `x = x - 10` is decreasing x by 10 so after 5 iterations of while loop x will become 40. When x becomes 40, the condition `if x < 50` becomes true so `keepgoing` is set to `False` due to which the while loop stops iterating.

Question 2d

Predict the output of the following code fragments:

```
x = 45
while x < 50 :
    print (x)
```

Answer

This is an endless (infinite) loop that will keep printing 45 continuously.

As the loop control variable x is not updated inside the loop neither there is any break statement inside the loop so it becomes an infinite loop.

Question 2e

Predict the output of the following code fragments:

```
for x in [1,2,3,4,5]:
    print (x)
```

Answer

Output

```
1
2
3
4
5
```

Explanation

x will be assigned each of the values from the list one by one and that will get printed.

Question 2f

Predict the output of the following code fragments:

```
for p in range(1,10):
   print (p)
```

Answer

Output

```
1
2
3
4
5
6
7
8
9
```

Explanation

`range(1,10)` will generate a sequence like this [1, 2, 3, 4, 4, 5, 6, 7, 8, 9]. p will be assigned each of the values from this sequence one by one and that will get printed.

Question 2g

Predict the output of the following code fragments:

```
for z in range(-500, 500, 100):
    print (z)
```

Answer

Output

```
-500
-400
-300
-200
-100
0
100
200
300
400
```

Explanation

`range(-500, 500, 100)` generates a sequence of numbers from -500 to 400 with each subsequent number incrementing by 100. Each number of this sequence is assigned to `z` one by one and then `z` gets printed inside the for loop.

Question 2h

Predict the output of the following code fragments:

```
x = 10
y = 5
for i in range(x-y * 2):
    print (" % ", i)
```

Answer

This code generates No Output.

Explanation

The `x-y * 2` in `range(x-y * 2)` is evaluated as below:
x - y * 2
⇒ 10 - 5 * 2
⇒ 10 - 10 [∵ * has higher precedence than -]
⇒ 0

Thus `range(x-y * 2)` is equivalent to `range(0)` which returns an empty sequence — [ ].

Question 2i

Predict the output of the following code fragments:

```
c = 0
for x in range(10):
    for y in range(5):
        c += 1
print (c)
```

Answer

Output

```
50
```

Explanation

Outer loop executes 10 times. For each iteration of outer loop, inner loop executes 5 times. Thus, the statement `c += 1` is executed 10 * 5 = 50 times. c is incremented by 1 in each execution so final value of c becomes 50.

Question 2j

Predict the output of the following code fragments:

```
x = [1,2,3]
counter = 0
while counter < len(x):
    print(x[counter] * '%')
    for y in x:
        print(y * '* ')
    counter += 1
```

Answer

Output

```
%
*
* *
* * *
%%
*
* *
* * *
%%%
*
* *
* * *
```

Explanation

In this code, the for loop is nested inside the while loop. Outer while loop runs 3 times and prints % as per the elements in x in each iteration. For each iteration of while loop, the inner for loop executes 3 times printing * as per the elements in x.

Question 2k

Predict the output of the following code fragments:

```
for x in 'lamp':
    print(str.upper(x))
```

Answer

Output

```
L
A
M
P
```

Explanation

The for loop extracts each letter of the string 'lamp' one by one and place it in variable x. Inside the loop, x is converted to uppercase and printed.

Question 2l

Predict the output of the following code fragments:

```
x = 'one'
y = 'two'
counter = 0
while counter < len(x):
    print(x[counter], y[counter])
    counter += 1
```

Answer

Output

```
o t
n w
e o
```

Explanation

Inside the while loop, each letter of x and y is accessed one by one and printed.

Question 2m

Predict the output of the following code fragments:

```
x = "apple, pear, peach"
y = x.split(", ")
for z in y :
   print(z)
```

Answer

Output

```
apple
pear
peach
```

Explanation

x.split(", ") breaks up string x into a list of strings so y becomes ['apple', 'pear', 'peach']. The for loop iterates over this list and prints each string one by one.

Question 2n

Predict the output of the following code fragments:

```
x ='apple, pear, peach, grapefruit'
y = x.split(', ')
for z in y:
    if z < 'm':
        print(str.lower(z))
    else:
        print(str.upper(z))
```

Answer

Output

```
apple
PEAR
PEACH
grapefruit
```

Explanation

x.split(', ') breaks up string x into a list of strings so y becomes ['apple', 'pear', 'peach', 'grapefruit']. The for loop iterates over this list. apple and grapefruit are less than m (since a and g comes before m) so they are converted to lowercase and printed whereas pear and peach are converted to uppercase and printed.

Question 3

Find and write the output of the following python code:

```
for Name in ['Jayes', 'Ramya', 'Taruna', 'Suraj'] :
    print (Name)
    if Name[0] == 'T' :
        break
    else :
        print ('Finished!')
print ('Got it!')
```

Answer

Output

```
Jayes
Finished!
Ramya
Finished!
Taruna
```

```
Got it!
```

Explanation

The for loop iterates over each name in the list and prints it. If the name does not begin with the letter T, Finished! is printed after the name. If the name begins with T, break statement is executed that terminates the loop. Outside the loop, Got it! gets printed.

Question 4(i)

How many times will the following for loop execute and what's the output?

```
for i in range(-1, 7, -2):
    for j in range (3):
        print(1, j)
```

Answer

The loops execute 0 times and the code produces no output. range(-1, 7, -2) returns an empty sequence as there are no numbers that start at -1 and go till 6 decrementing by -2. Due to empty sequence, the loops don't execute.

Question 4(ii)

How many times will the following for loop execute and what's the output?

```
for i in range(1,3,1):
    for j in range(i+1):
        print('*')
```

Answer

Loop executes for 5 times.

Output

```
*
*
*
*
*
```

Explanation

range(1,3,1) returns [1, 2]. For first iteration of outer loop j is in range [0, 1] so inner loop executes twice. For second iteration of outer loop j is in range [0, 1, 2] so inner loop executes 3 times. This makes the total number of loop executions as 2 + 3 = 5.

Question 5

Is the loop in the code below infinite? How do you know (for sure) before you run it?

```
m = 3
n = 5
```

```
while n < 10:
    m = n - 1
    n = 2 * n - m
    print(n, m)
```

Answer

The loop is not infinite. To know this without running it we can analyze how n is changed inside the loop in the following way:

n = 2 * n - m

Substituting value of m from m = n - 1,

n = 2 * n - (n - 1)
⇒ n = 2 * n - n + 1
⇒ n = 2n - n + 1
⇒ n = n + 1

Therefore, inside the loop n is incremented by 1 in each iteration. Loop condition is n < 10 and initial value of n is 5. So after 5 iterations, n will become 10 and the loop will terminate.

Type C: Programming Practice/Knowledge based Questions

Question 1

Write a program to print one of the words negative, zero, or positive, according to whether variable x is less than zero, zero, or greater than zero, respectively

Solution

```
x = int(input("Enter x: "))

if x < 0:
    print("negative")
elif x > 0:
    print("positive")
else:
    print("zero")
```

Output

```
Enter x: -5
negative

Enter x: 0
zero

Enter x: 5
positive
```

Question 2

Write a program that returns True if the input number is an even number, False otherwise.

Solution

```
x = int(input("Enter a number: "))

if x % 2 == 0:
    print("True")
else:
    print("False")
```

Output

```
Enter a number: 10
True

Enter a number: 5
False
```

Question 3

Write a Python program that calculates and prints the number of seconds in a year.

Solution

```
days = 365
hours = 24
mins = 60
secs = 60
secsInYear = days * hours * mins * secs
print("Number of seconds in a year =", secsInYear)
```

Output

```
Number of seconds in a year = 31536000
```

Question 4

Write a Python program that accepts two integers from the user and prints a message saying if first number is divisible by second number or if it is not.

Solution

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))

if a % b == 0:
    print(a, "is divisible by", b)
else:
```

```
    print(a, "is not divisible by", b)
```

Output

```
Enter first number: 15
Enter second number: 5
15 is divisible by 5

Enter first number: 13
Enter second number: 7
13 is not divisible by 7
```

Question 5

Write a program that asks the user the day number in a year in the range 2 to 365 and asks the first day of the year — Sunday or Monday or Tuesday etc. Then the program should display the day on the day-number that has been input.

Solution

```
dayNames = ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY",
"SATURDAY", "SUNDAY"]

dayNum = int(input("Enter day number: "))
firstDay = input("First day of year: ")

if dayNum < 2 or dayNum > 365:
    print("Invalid Input")
else:
    startDayIdx = dayNames.index(str.upper(firstDay))
    currDayIdx = dayNum % 7 + startDayIdx - 1

    if currDayIdx >= 7:
        currDayIdx = currDayIdx - 7

    print("Day on day number", dayNum, ":", dayNames[currDayIdx])
```

Output

```
Enter day number: 243
First day of year: FRIDAY
Day on day number 243 : TUESDAY
```

Question 6

One foot equals 12 inches. Write a function that accepts a length written in feet as an argument and returns this length written in inches. Write a second function that asks the user for a number of feet and returns this value. Write a third function that accepts a number of inches and displays this to the screen. Use these three functions to write a program that asks the user for a number of feet and tells them the corresponding number of inches.

Solution

```
def feetToInches(lenFeet):
    lenInch = lenFeet * 12
    return lenInch

def getInput():
    len = int(input("Enter length in feet: "))
    return len

def displayLength(l):
    print("Length in inches =", l)

ipLen = getInput()
inchLen = feetToInches(ipLen)
displayLength(inchLen)
```

Output

```
Enter length in feet: 15
Length in inches = 180
```

Question 7

Write a program that reads an integer N from the keyboard computes and displays the sum of the numbers from N to (2 * N) if N is nonnegative. If N is a negative number, then it's the sum of the numbers from (2 * N) to N. The starting and ending points are included in the sum.

Solution

```
n = int(input("Enter N: "))
sum = 0
if n < 0:
    for i in range(2 * n, n + 1):
        sum += i
else:
    for i in range(n, 2 * n + 1):
        sum += i

print("Sum =", sum)
```

Output

```
Enter N: 5
Sum = 45

Enter N: -5
Sum = -45
```

Question 8

Write a program that reads a date as an integer in the format MMDDYYYY. The program will call a function that prints print out the date in the format <Month Name> <day>, <year>.

Sample run :

```
Enter date : 12252019
December 25, 2019
```

Solution

```
months = ["January", "February", "March", "April", "May", "June",
"July", "August", "September", "October", "November", "December"]

dateStr = input("Enter date in MMDDYYYY format: ")
monthIndex = int(dateStr[:2]) - 1
month = months[monthIndex]
day = dateStr[2:4]
year = dateStr[4:]

newDateStr = month + ' ' + day + ', ' + year
print(newDateStr)
```

Output

```
Enter date in MMDDYYYY format: 12252019
December 25, 2019
```

Question 9

Write a program that prints a table on two columns — table that helps converting miles into kilometres.

Solution

```
print('Miles | Kilometres')
print(1, "\t", 1.60934)
for i in range(10, 101, 10):
    print(i, "\t", i * 1.60934)
```

Output

```
Miles | Kilometres
1        1.60934
10       16.0934
20       32.1868
30       48.2802
40       64.3736
50       80.467
60       96.5604
70       112.6538
```

```
80        128.7472
90        144.8406
100       160.934
```

Question 10

Write another program printing a table with two columns that helps convert pounds in kilograms.

Solution

```
print('Pounds | Kilograms')
print(1, "\t", 0.4535)
for i in range(10, 101, 10):
    print(i, "\t", i * 0.4535)
```

Output

```
Pounds | Kilograms
1         0.4535
10        4.535
20        9.07
30        13.605
40        18.14
50        22.675
60        27.21
70        31.745
80        36.28
90        40.815
100       45.35
```

Question 11

Write a program that reads two times in military format (0900, 1730) and prints the number of hours and minutes between the two times.

A sample run is being given below :

```
Please enter the first time : 0900
Please enter the second time : 1730
8 hours 30 minutes
```

Solution

```
ft = input("Please enter the first time : ")
st = input("Please enter the second time : ")

# Converts both times to minutes
fMins = int(ft[:2]) * 60 + int(ft[2:])
sMins = int(st[:2]) * 60 + int(st[2:])

# Subtract the minutes, this will give
# the time duration between the two times
```

```
diff = sMins - fMins;

# Convert the difference to hours & mins
hrs = diff // 60
mins =  diff % 60

print(hrs, "hours", mins, "minutes")
```

Output

```
Please enter the first time : 0900
Please enter the second time : 1730
8 hours 30 minutes

Please enter the first time : 0915
Please enter the second time : 1005
0 hours 50 minutes
```