# ITI 1120

# Lab # 3

Boolean expressions, if statements, debugging, and …

# What is in this lab?

1. This lab bas 3 Tasks (3rd task has several parts). You should try to do these parts at home.

   Task 2 and 3 are in the interactive textbook. Watch the video I provided (follow the link at page 7) before doing these two tasks.

2. And 5 programming exercises (0 to 4)

The slides that are included here but do not explain either Tasks or Programming exercises are there as reminder of some relevant material that is needed for this lab.

# Lab 3 overview

- Open a browser and log into Brightspace
- On the left hand side under Labs tab, find lab3 material contained in either lab3-students.zip or lab3-studnets.pdf file
- Download that file to the Desktop and open it.
- Read slides one by one and follow the instructions explaining what to do. When stuck ask for help from TA. Ideally you should try to complete some, or all of this, at home and then use the lab as a place where you can get help with things you may have had difficulties with.

# Before starting, always make sure you are running Python 3

This slide is applicable to all labs, exercises, assignments … etc

ALWAYS MAKE SURE FIRST that you are running Python 3.4 (3.5 and 3.6 are fine too). That is, when you click on IDLE (or start python any other way) look at the first line that the Python shell displays. It should say Python 3.4 or 3.5 or 3.6 (and then some extra digits)

If you do not know how to do this, read the material provided with Lab 1. It explains it step by step

# div // and mod % operators in Python

If uncertain, here is how to compute `a//b` (i.e. integer division) and `a%b` (i.e remainder)

1. Compute `x=a/b`

2. `a//b` is then equal to the whole (i.e integer) part of `x`. More precisely `a//b` is equal the integer that is closest to `a/b` but not bigger than `a/b`

3. `a%b` is equal to `a - (a//b) * b`

# Task 1

What is the type and value of each of the following expressions in Python? Do this in you head (and/or paper) first. The check both columns in Python shell. eg. You can test what kind of value the first expression returns by typing `type(13*0.1)` in Python shell

| Expression | Type | Value |
|---|---|---|
| `13 * 0.1` | `float` | `1.3` |
| `int(13) * 0.1` | | |
| `13 * int(0.1)` | | |
| `int(13 * 0.1)` | | |
| `13 % 7` | | |
| `6%3` | | |
| `6//2.5` | | |
| `6%2.5` | | |
| `2<3== 4<5` | | |

# Interactive textbook (Reminder)

Here is a link to the interactive textbook:

https://runestone.academy/runestone/static/thinkcspy/index.html

# Task 2: Debugging

Follow this link, and read, run and do the exercises there:

https://runestone.academy/runestone/static/thinkcspy/Debugging/HowtoAvoidDebugging.html

An example used in the above link refers to the following problem from Chapter: Simple Python Data, Section: Exercises, Question 3:
https://runestone.academy/runestone/static/thinkcspy/SimplePythonData/Exercises.html

```
Many people keep time using a 24 hour clock (11 is 11am and 23 is
11pm, 0 is midnight). If it is currently 13 and you set your alarm
to go off in 50 hours, it will be 15 (3pm). Write a Python program
to solve the general version of the above problem. Ask the user
for the time now (in hours), and then ask for the number of hours
to wait for the alarm. Your program should output what the time
will be on the clock when the alarm goes off.
```

# Programming exercise 0:
# More debugging and coding

Open file q0.py and solve the 3 programming exercises as instructed.

# Boolean Expressions

Boolean expressions evaluate to `True` or `False`

Logical/Boolean operators in Math and Python:

| Math | Python |
|---|---|
| AND | and |
| OR | or |
| NOT | not |

Here is how your compare two variables a and b in Math and Python

| Math | Python |
|---|---|
| a = b | a == b |
| a ≤ b | a <= b |
| a ≥ b | a >= b |
| a ≠ b | a != b |

# Truth table

A TRUTH TABLE for a compound Boolean expression shows the results for all possible combinations of the simple expressions:

| x | y | | x and y | x or y | not x |
|---|---|---|---|---|---|
| False | False | | False | False | True |
| True | False | | False | True | False |
| False | True | | False | True | |
| True | True | | True | True | |

# Testing if two strings are equal

For one of the programming exercise you will need to know that how to compare if two strings are equal. You can do that simply buy using == operator. Here are some examples

```
>>> 'A'=='A'
True
>>> 'Anna'=='Anna'
True
>>> 'Anna'='anna'
False
>>> a='June'
>>> a=='june'
False
>>> a=='June'
True
>>> b='Ju' + 'ne'
>>> a==b
True
```

ps. Do not copy paste the above into python shell. It will likely give you syntax errors since quotes do not copy/paste correctly from Word.

# Examples of compound boolean expressions:

- This is how you would test if `age` is at least 18 and at most 65:

```
age>=18 and age <= 65
```

- `not` is an operator to negate the value of a simple or compound Boolean expression. Suppose `age = 15`. Then:

`age > 16` evaluates to `False`, and `not(age > 16)` evaluates to `True`

- Suppose day refers to a string which is a day of a week. Here is how you would test if `day` is a weekend:

```
day=="Saturday" or day=="Sunday"
```

- Here are two ways to test if `age` is less than 18 and greater than 65. Think about the 2nd one

  - 1st way: `age<18 and age > 65`

  - 2nd way: `not(age>=18 and age <= 65)`

# Task 3: if statements

Follow all the links below. Read, run and do the exercises in each of them. No need to watch the videos.

## Boolean Expressions
https://runestone.academy/runestone/static/thinkcspy/Selection/BooleanValuesandBooleanExpressions.html

## Logical/Boolean operators
https://runestone.academy/runestone/static/thinkcspy/Selection/Logicaloperators.html

## PrecedenceOfOperations
https://runestone.academy/runestone/static/thinkcspy/Selection/PrecedenceofOperators.html

## Two-way if statement:
https://runestone.academy/runestone/static/thinkcspy/Selection/ConditionalExecutionBinarySelection.html

## One-way if statement:
https://runestone.academy/runestone/static/thinkcspy/Selection/OmittingtheelseClauseUnarySelection.html

## Nested if statement:
https://runestone.academy/runestone/static/thinkcspy/Selection/Nestedconditionals.html

## Chained conditionals (i.e elif)
https://runestone.academy/runestone/static/thinkcspy/Selection/Chainedconditionals.html

# DOCSTRINGS

For all the python functions that you will write in the following questions (and in the future) make sure you document your functions by writing docstrings including the type contract (as we have done in class).

Test if you docstrings appear as help of your function by running `help(your_function_name)` in python shell.

# Programming exercises: Question 1

Write **function** called `pay` that takes as input an hourly wage and the number of hours an employee worked in the last week. The function should compute and return the employee's pay. Overtime work should be paid in this way: Any hours beyond 40 but less than or equal 60 should be paid at 1.5 times the regular hourly wage. Any hours beyond 60 should be paid at 2 times the regular hourly wage. Important:

Important: Note that I did not give you either the number of parameters or the names of parameters. You will have to figure it out on your own for this function. Looking at the test examples below should help.

Example tests:

```
>>> pay(10, 35)
350
>>> pay(10, 45)
475.0
>>> pay(10, 61)
720.0
```

# Programming exercises: Question 2

Rock, Paper, Scissors is a two-player game in which each player chooses one of three items. If both players choose the same item, the game is tied. Otherwise, the rules that determine the winner are:

(a)Rock always beats Scissors (Rock crushes Scissors)

(b) Scissors always beats Paper (Scissors cut Paper)

(c) Paper always beats Rock (Paper covers Rock)

Write **a function** called `rps` that takes the choice `'R'`, `'P'`, or `'S'` of player 1 and the choice of player 2, and returns –1 if player 1 wins, 1 if player 2 wins, or 0 if there is a tie."

Note that I did not give you either the number of parameters or the names of parameters. You will have to figure it out on your own. Looking at the example runs below should help too:

Example tests:

```
>>> rps('R', 'P')
1
>>> rps('R', 'S')
-1
>>> rps('S', 'S')
0
```

# Programming exercises: Question 3a

Open a new file with IDLE. Write **a program** that has a function called `is_divisible`

- The function `is_divisible` has two input parameters that are integers `n` and `m` and returns `True` if `n` is divisible by `m` and `False` otherwise.
- Outside of that function, your program should interact with the user to get two integers. To determine if the 1st is divisible by the 2nd it should call `is_divisible` function. It should print a message explaining the result.

Two example tests (one on the left and one on the right)

```
>>>
Enter 1st integer:
9
Enter 2nd integer:
3
9 is divisble by 3
```

```
>>>
Enter 1st integer:
8
Enter 2nd integer:
3
8 is not divisble by 3
```

# Programming exercises: Question 3b

Open a new file with IDLE. Write **a program** that has two functions one called `is_divisible` and the other called `is_divisible23n8`

- The function `is_divisible` is the same as in the previous questions so you can copy/paste it to the beginning of the new file.
- The function `is_divisible23n8` has one input parameter, an integer. It should return string `"yes"` if the given number is divisible by 2 or 3 but not 8. Otherwise it should return a string `"no"`. Your function `is_divisible23n8` must use, i.e make a call to, `is_divisible`
- Outside of that function, your program should interact with the user to get one integer. It should call `is_divisible23n8` function to deterimen if the number the user gave is divisible by 2 or 3 but not 8. It should print a message explaining the result.

```
>>>
Enter an integer: 18
18 is divisible by 2 or 3 but not 8
>>>
Enter an integer: 16
It is not true that 16 is divisible by 2 or 3 but not 8
>>>
Enter an integer: 3
3 is divisible by 2 or 3 but not 8
```

# Bonus programming exercises:

For those who are done and want to more programming exercises for the lab or home, follow this link and complete any, or ideally all, exercises there:

https://runestone.academy/runestone/static/thinkcspy/Selection/Exercises.html