COMP7035

Python for Data Analytics and Artificial Intelligence

Renjie Wan, Xue Wei

3/9/2022

香港浸會大學
HONG KONG BAPTIST UNIVERSITY

DEPARTMENT OF
COMPUTER SCIENCE
計算機科學系

Lecturers for this course

Renjie Wan (万人杰)

Xue Wei (雪巍)

What we will learn?

		Topic	Hours
Renjie Wan	I.	Python Fundamentals A. Program control and logic B. Data types and structures C. Function D. File I/O	12
	II.	Numerical Computing and Data Visualization Tools and libraries such as A. NumPy B. Matplotlib C. Seaborn	9
Xue Wei	III.	Exploratory Data Analysis (EDA) with Python Tools and libraries such as A. Pandas B. Sweetviz	9
	IV.	Artificial Intelligence and Machine Learning with Python Tools and libraries such as A. Keras B. Scikit-learn	9

Setup of our course

- Continuous Assessments (40 %)
  - Homework and Exercise
- Tests (20%)
- Examination (40%)

Setup of our course

- Lectures
  - 3 hours in total for each course
  - About 1.5 hours for lectures and the left time for exercise
- Portfolio and Exercise
  - Portfolios are just copy of some examples in the lecture notes. Play them yourself. No need to submit them.
  - Exercises will be released in second half of the class.
  - Please understand each exercise and submit them before the next course.
  - Each exercise is with 2 scores. We will have 10 exercises in total.

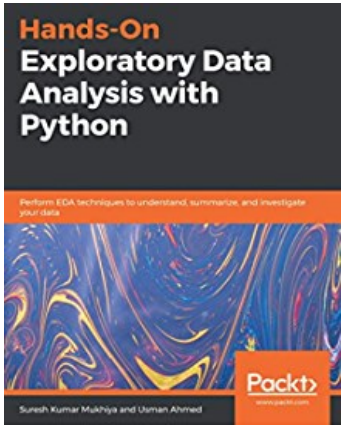
Setup of our course

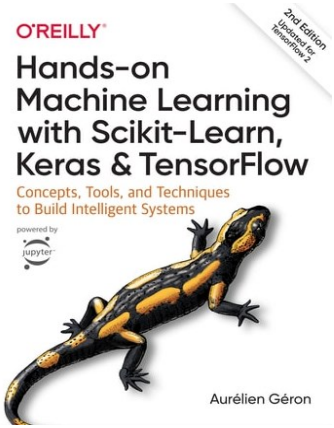
- Homework
  - We will have two homework. Remember to submit them before/by the ddl.
    - The first one is assumed to be released in Week 3, and its ddl is week 5
    - The second one is assumed to be released in Week 7, and its ddl is week 9
- Test
  - We will have two tests on the evening of Oct 10/11 and Nov 7/8.
  - The final time will be announced on Moodle.
  - Each test will be 1 hour ~ 1.5 hour.

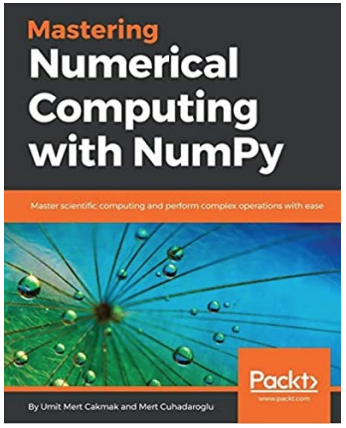
All announcements will be release on Moodle.

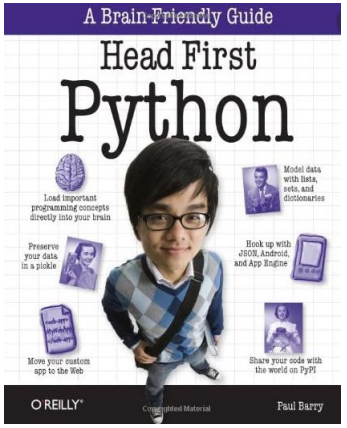
References

- The internet is an excellent source, and Google is a perfect starting point.
- The official documentation is also good, always worth a try: https://docs.python.org/.









Chinese Resources:

https://www.runoob.com/python/python-tutorial.html

Workload

- The only way to learn Python, is by writing Python a lot. So you are expected to put in effort.
- The examples used in our course will be provided for you. You can play them yourself.
- If you are new to programming, consider this a hard class where you will have to figure out quite a bit on your own. However, if you have a solid background in Python or any another languages, this class should be pretty easy.
- If you are an expert in Python, please help your classmates.

Suggestions for new beginners

- The course is relatively short. We will go fast to cover many important knowledge about scientific python programming.
- Alternative: spend some time learning on your own (Codecademy / Udacity etc). There are so many excellent resources online these days.
- Try to learn Python from your friends.

How to install Python

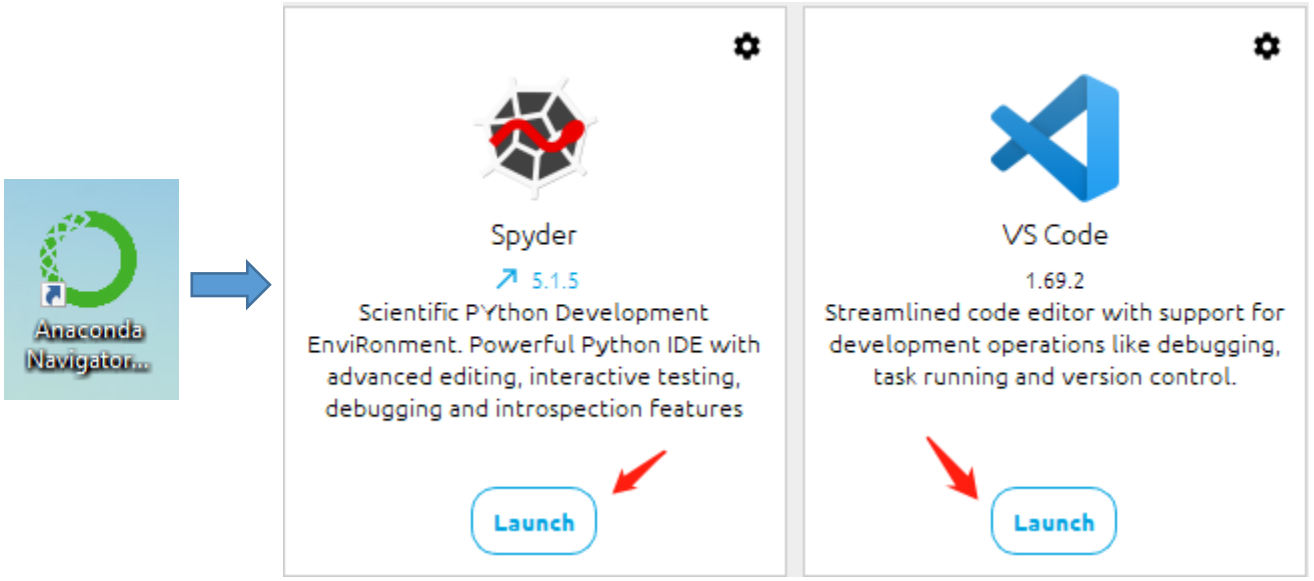
- Many alternatives, but a prepackaged distribution is suggested, such as Anaconda:

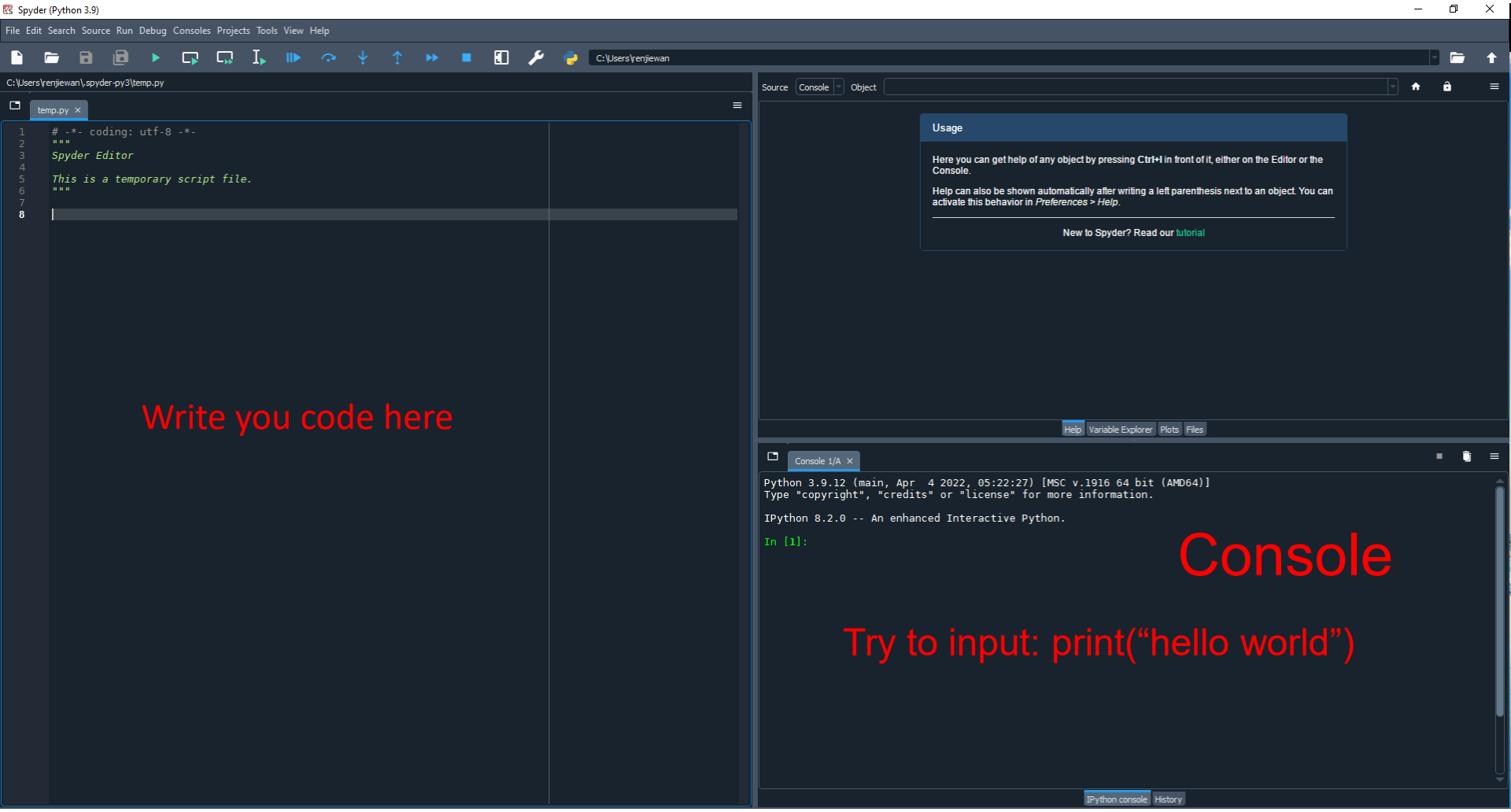
https://www.anaconda.com/products/distribution

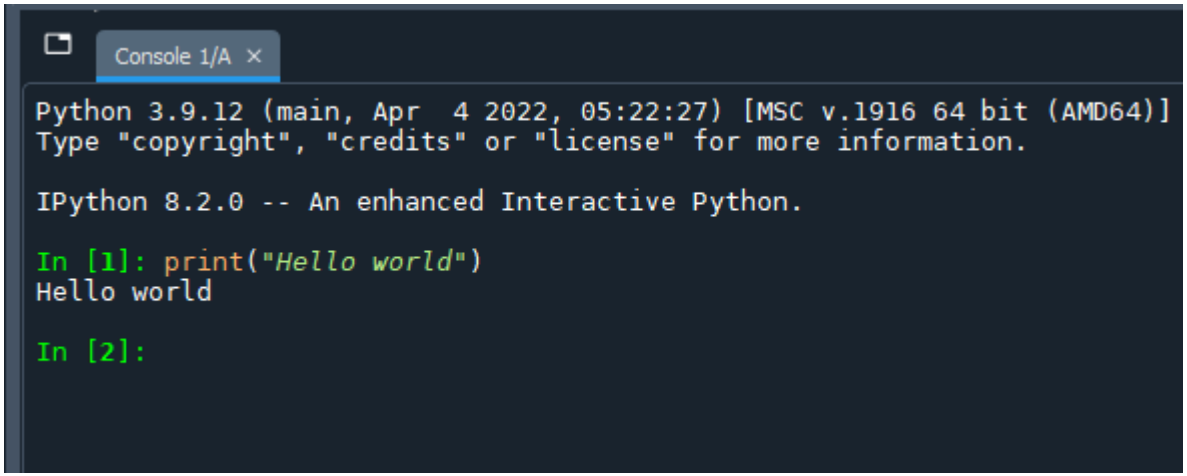
This is very easy to install and also comes with a lot of packages.

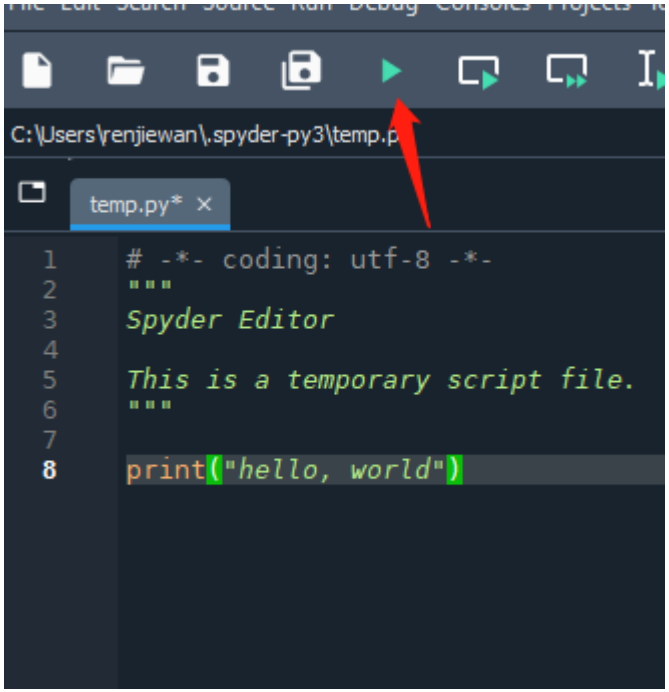
- Or you can also try google colab for learning purposes

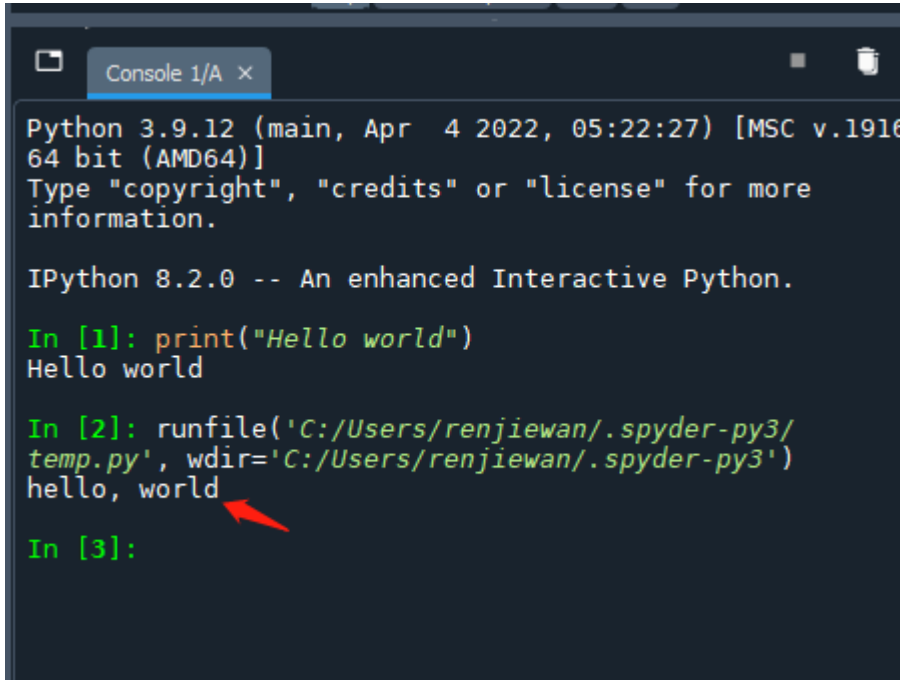
How to use Anaconda




Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\renjiewan
C:\Users\renjiewan\.spyder-py3\temp.py
temp.py
-*- coding: utf-8 -*-
"""
Spyder Editor
This is a temporary script file.
"""
Write you code here
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help Variable Explorer Plots Files
Console 1/A
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.2.0 -- An enhanced Interactive Python.
In [1]:
Console
Try to input: print("hello world")
IPython console History

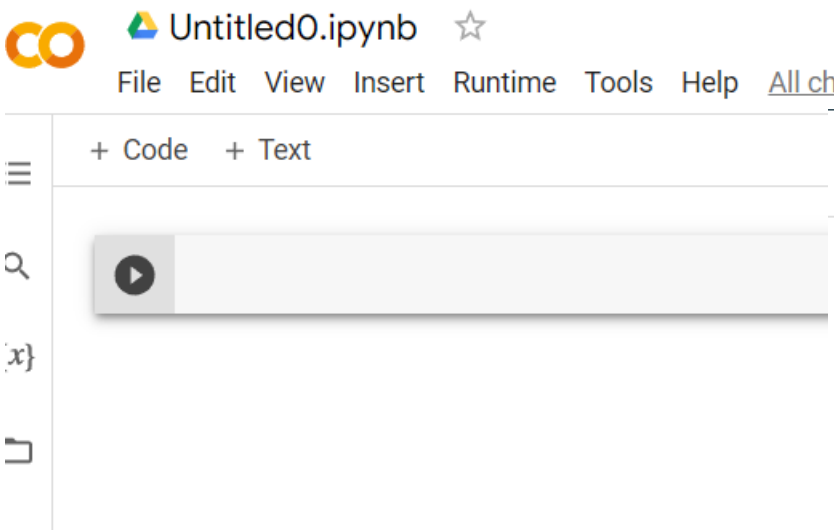

Console 1/A
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 8.2.0 -- An enhanced Interactive Python.
In [1]: print("Hello world")
Hello world
In [2]:


C:\Users\renjiewan\.spyder-py3\temp.p
temp.py*
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7
8 print("hello, world")


Console 1/A
Python 3.9.12 (main, Apr 4 2022, 05:22:27) [MSC v.1916
64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.
IPython 8.2.0 -- An enhanced Interactive Python.
In [1]: print("Hello world")
Hello world
In [2]: runfile('C:/Users/renjiewan/.spyder-py3/
temp.py', wdir='C:/Users/renjiewan/.spyder-py3')
hello, world
In [3]:

Jupyter Lab

- You can also use google colab directly.
- link: https://colab.research.google.com/
- Click file and then “new notebook”. You can see the following and can begin to write your own code.



Google Colab

- Use google Colab to open "*.ipynb" file

File->Upload Notebook

The properties of Python

- Relatively easy to learn
- Fast to write code
- Very versatile (vs Matlab/R)
- Widely used in Deep Learning like Pytorch, Tensorflow, and JAX

Compare with other languages

```java
// Your First Program

class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

JAVA

```cpp
#include <iostream>

int main() {
    std::cout << "Hello World!";
    return 0;
}
```

C++

例 1.1 要求在屏幕上输出以下一行信息。

This is a C program.

解题思路：在主函数中用 printf 函数原样输出以上文字。

编写程序：

```c
#include <stdio.h>                  //这是编译预处理指令
int main( )                         //定义主函数
{                                   //函数开始的标志
   printf ("This is a C program.\n");  //输出所指定的一行信息
   return 0;                        //函数执行完毕时返回函数值 0
}                                   //函数结束的标志
```

C

```python
print("Hello, world!")
```

print statement

- We can print output to screen using the **print** command

E1

```
print("Hello, world!")
```

E2

```
import math
print(math.pi)
```

E3

```
import math
print(math.floor(1.5))
```

Values

- A value is the fundamental thing that a program manipulates.
- Values can be “Hello, world!”, 42, 12.34, True/False
- Values have types. . .

Types

- Each value has its corresponding types

  boolean True/False
  string “Hello, world!”
  integer 92
  float 3.1415

- Use **type** to find out the type of a value, as in

E5-E8

```
type(2/4)       float
type("Hello, World") String
type(True)    boolean
type(False)   boolean
```

Variables

- One of the most basic and powerful concepts is that of a variable.
- A variable *assigns* a name to a value.

E9

```
x = 1
print(x)
```

```
#E11
y = "test"
print(y)
```

E10

```
import math
a = 1.5
print(math.floor(a))
```

```
#E12
x = 1
x = "string"
print(x)
```

Try E13 to E16 yourself

Some hints about **variables**

- Almost always preferred to use variables over values:
  - Easier to update code
  - Easier to understand code (useful naming)

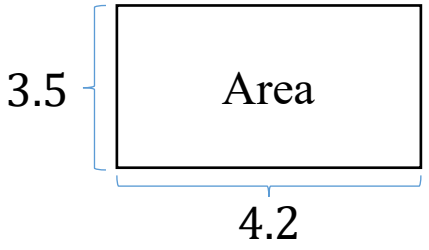
What does the following code do:

```
x = 4.2
y = 3.5
c = x * y
print(c)
```

Unclear about its usage

```
length = 4.2
height = 3.5
area = length * height
print(area)
```

It is calculating the area of a rectangle



Booleans

- Boolean expressions:
  - == equals: 5==5 yields True
  - != does not equal: 1!= 1 yields False, while 2!=1 yields True
  - > greater: 2 > 1 yields True, while 1 > 2 yields False
  - >= greater than or equal: 5 >= 5 yields True
  - Similarly, we have < and <=

```
[ ]  #E22
     2 < 1
```
False

```
[ ]  #E23
     2 == 2
```
True

```
[ ]  #E24
     2 != 2
```
False

Keywords

- Not allowed to use keywords, they define structure and rules of a language.
- Python has 29 keywords, they include:
  - and
  - def
  - for
  - return
  - is
  - in
  - class

Python operators

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

```
#E26
5/2
```

```
2.5
```

Divides the number on its left by the number on its right and returns a floating point value.

- Modulus (%)

```
#E33
5%3
```

```
2
```

```
#E32
6%3
```

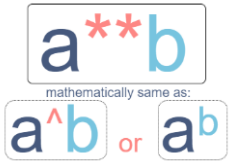
```
0
```

The % symbol in Python is called the Modulo Operator. It returns the remainder of dividing the left hand operand by right hand operand. It's used to get the remainder of a division problem.

- Exponentiation (**)

```
#E34
4**3
```

```
64
```



- Floor division (//)

```
#E31
5//3 #=1 remainder 2
```

```
1
```

Divides the number on its left by the number on its right, rounds down the answer, and returns a whole number.

String

- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- 'hello' is the same as "hello".

```
#E35
str1 = "Hello, "
str2 = "World!"
```

```
#E36
str3 = str1 + str2
str3
print(str3)
```

```
Hello, World!
```

```
# E39
str1 = "Hello, World!"
print(str1)
```
Hello, World!
```
print(str1.upper())
```
HELLO, WORLD!
```
print(str1.lower())
```
hello, world!

```
#E40
str1 = "Hello, World!"
str1.replace('l', 'p')
```
Heppo, Worpd!

String

- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- 'hello' is the same as "hello".

You can also compare two strings using booleans

```
#E41
str1 = '11111'
str2 = '11111'
str3 = '222222'
print(str1 == str2)

print(str3 > str2)
```

```
True
True
```

Control statement

- Control statements allow you to do more complicated tasks:
  - for
  - while
  - if

if

- Using **if**, we can execute part of a program conditional on some statement being true.

```
#E41
if True:
   Conduct statement
```

True: 1==1, 2>1, 3!=1, 10>0, 0<2

False: 1!=1, 2<1, 3==1, 10<0, 0>2

```
#E41
if 2 > 0:
    print("Two is greater than zero")
```

```
Two is greater than zero
```

Program **logic**

- **and**: Logic **AND**: True if both the operands are true.
  - True and False yields False
- **or**: Logic **OR**: True if either of the operands is true.
  - True or False yields True
- **not**: Logic NOT: True if operand is false.
  - not True yields False, not False yields True

```
[2]  #E17
     True and False

     False
```

```
[6]  #E19
     not True

     False
```

```
[7]  #E18
     True or False

     True
```

```
[5]  #E20
     not False

     True
```

Use if with logic **and**

- Think about a scenario:
  - You have two numbers and want to know whether they are all greater than 0

If a is greater than 0 and b is greater than 0, then the two numbers are greater than 0
(a > 0 and b > 0)

```
#E43
a = 10
b = 10
#Write your code here
```

Result: The numbers are greater than 0

Use if with logic **and**

- Think about a scenario:
  - You have two numbers and want to know whether they are all greater than 0

If a is greater than 0 and b is greater than 0, then the two numbers are greater than 0
(a > 0 and b > 0)

```
#E43
a = 10
b = 10
if a > 0 and b > 0:
  print("The numbers are greater than 0")
```

Result: The numbers are greater than 0

if-else statement

- We can add more conditions to the **if** statement using **else**
- **else** is used to cover conditions not covered by **if**

```
if x is equal to y, then
display “The two number are equal”
```

```
x = 1
y = 1
if x == y:
    print("The two numbers are equal")
```

```
if x is equal to y, then
display “The two number are equal” otherwise
display “The two numbers are not equal”.
```

```
#E42
x = 1
y = 2

#Write code here
```

if-else statement

- We can add more conditions to the **if** statement using **else**
- **else** is used to cover conditions not covered by **if**

```
if x is equal to y, then
display "The two number are equal"
```

```
x = 1
y = 1
if x == y:
    print("The two numbers are equal")
```

```
if x is equal to y, then
display "The two number are equal" otherwise
display "The two numbers are not equal".
```

```
#E42
x = 1
y = 2

if x == y:
    print("The two numbers are equal")
else:
    print("The two numbers are not equal")
```

Use if with logic **or**

- Think about a scenario:
  - Today is Saturday, you want to know whether you need to go to work on weekend?

if today is on Saturday or Sunday.
On the two days, you just need to rest at home

```
#E43
today = 'Sunday'
if today=='Sunday' or today=='Saturday':
  print('Today is off. Rest at home.')
```

if-else statement

- We can add more conditions to the **if** statement using **else**
- **else** is used to cover conditions not covered by **if**

```
#E44
today = 'Friday'
if today=='Sunday' or today=='Saturday':
    print('Today is off. Rest at home')
else:
    print('go to work')
```

Indentation

- In Python, blocks of code are defined using indentation.
- This means that everything indented after an **if** statement is only executed if the statement is true.
- If the statement is **False**, the program skips all indented code and resumes at the first line of unindented code

```
#E-Indentation-1
    if 2<1:
      print("2<1")
      print("1>2")
```

show nothing

```
#E-Indentation-2
    if 2<1:
      print("2<1")
    print("1>2")
```

1>2

for loops

```
for item in iterable:
    statement(s)
```

- When some actions are repeated, this can be achieved by a **for** loop.

```
#E46
print("loop 1")
for i in range(5): # default - start at 0, increment by 1
    print(i)
```

Result: 0 1 2 3 4

```
print("\nloop 2")
for i in range(2, 10): # inputs are start, stop
    print(i)
```

Result: 2 3 4 5 6 7 8 9

Here, **range(n)** gives us a **list** with integers 0,……, $n-1$.
For example, **range(5)** gives us a list with 0, 1, 2, 3, 4

while loops

- When we do not know how many iterations are needed, we can use **while**

```
#E47
i = 1
while i < 10:
    print(i)
    i = i + 1
```

Result: 1 2 3 4 5 6 7 8 9

continue

- **continue** continues with the next iteration of the smallest enclosing loop.

```
#E48
for num in range(2, 10):
    if num % 2 == 0:
        continue # this jumps us back to the top
    print(f"Found {num}, an odd number")
```

break

- The **break** statement allows us to jump out of the smallest enclosing **for** or **while** loop.

You have a word. for all letters in this word, find whether “e” exists or not. If it indeed exits, immediately stop the program

```
#E49
for letter in "Hello":
    if letter == "e":
        print(letter)
        break
```

pass

- The **pass** statement does nothing, which can come in handy when you are working on something and want to implement some parts of your code later.

```
a = 10
b = 20
if(a<b):
  pass
else:
  print("b<a")
```

```
li =['a', 'b', 'c', 'd']
for i in li:
  if(i =='a'):
    pass
  else:
    print(i)
```

Please play the code yourself!