Python for Data Analysis

Kann Bonpagna

Trainer’s Background

2021 - Present	: PhD Student at the Université Grenoble Alpes, France
2019 – Present	: Lecturer and Researcher at Cambodia Academy of Digital Technology
2018 – 2019	: Master Student in Informatics – Data Science, Grenoble INP & Université Grenoble Alpes, France
2013 – 2018:	Bachelor of Engineering in Information and Communication (GIC-Gen. 17), ITC
2013 – 2017	: Bachelor of Education in English, Institute of Foreign Languages, RUPP

How about you?

1. What is your name?
2. Are you currently an employee or a student?
3. What is your professional/academic background?
4. What is your primary purpose of joining this training?

Training Objectives

1. Demonstrate the knowledge regarding the basics of Python programming environment, including fundamental python programming techniques.

2. Apply data manipulation and cleaning techniques using the popular Python’s Pandas library.

3. Visualize the data using Python libraries such as matplotlib, and seaborn.

4. Discuss the abstraction of the use of statistical analysis in data processing and data analysis.

About Python

Python is an interpreted language

Created in 1989 by Guido Van Rossum

Looking at each instruction, one at a time.

Turns that instruction into something that can be run.

Example:

```
>>> x = 1;
>>> y = 2;
>>> x+y
3
```

Answer !

"Hello world" in different languages
Python:
PHP:
C:
Assembly:
FFFFFFF FFFFFFF FFFFFF FFFUU UUUU UUUU UUUU UUUU UUUU-
VIA 9GAG.COM

```
print "Hello world!"
```

```
<?php
echo 'Hello world!';
?>
```

```
#include <stdio.h>
main()
{
        printf ("Hello World!\n");
}
```

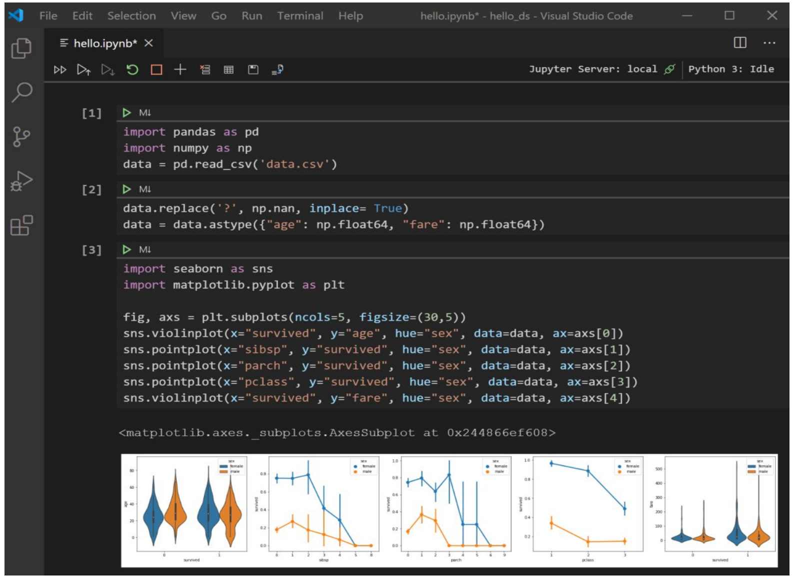
```
.model small
.stack 100h
.data
   bonjour  db  "Hello world!$"
.code
   main  proc
      mov  AX,@data
      mov  DS, AX
      mov  DX, offset bonjour
      mov  AX,0900h
      int  21h
      mov  AX,4C00h
      int  21h
      main endp
   end main
```

Applications of Python

1. **Scientific and Numeric Applications**

   Libraries :

   - **SciPy** (scientific numeric library)
   - **Pandas** (data analytics library)
   - **Natural Language Toolkit** (Mathematical And text analysis)


File Edit Selection View Go Run Terminal Help
hello.ipynb* - hello_ds - Visual Studio Code
hello.ipynb*
Jupyter Server: local
Python 3: Idle

```
[1]
import pandas as pd
import numpy as np
data = pd.read_csv('data.csv')
```

```
[2]
data.replace('?', np.nan, inplace= True)
data = data.astype({"age": np.float64, "fare": np.float64})
```

```
[3]
import seaborn as sns
import matplotlib.pyplot as plt

fig, axs = plt.subplots(ncols=5, figsize=(30,5))
sns.violinplot(x="survived", y="age", hue="sex", data=data, ax=axs[0])
sns.pointplot(x="sibsp", y="survived", hue="sex", data=data, ax=axs[1])
sns.pointplot(x="parch", y="survived", hue="sex", data=data, ax=axs[2])
sns.pointplot(x="pclass", y="survived", hue="sex", data=data, ax=axs[3])
sns.violinplot(x="survived", y="fare", hue="sex", data=data, ax=axs[4])
```

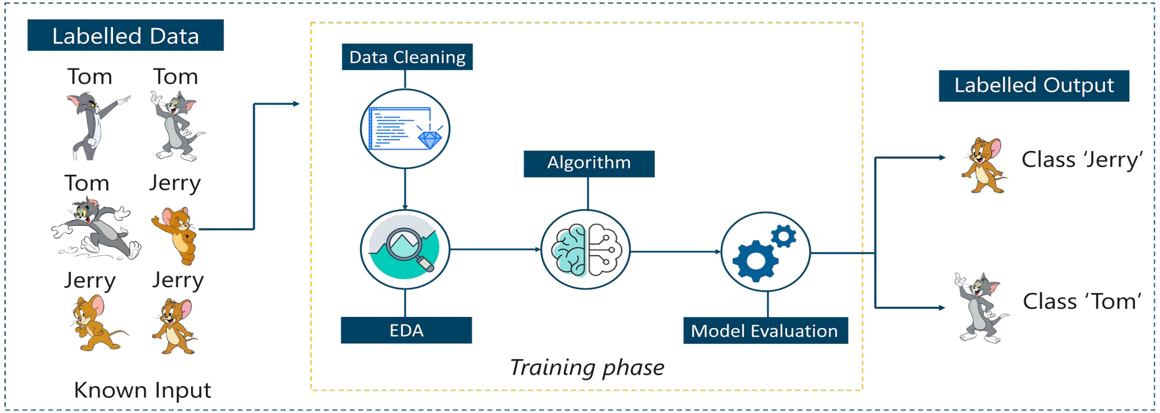
```
<matplotlib.axes._subplots.AxesSubplot at 0x244866ef608>
```

Applications of Python

2. Artificial Intelligence and Machine Learning

Libraries :

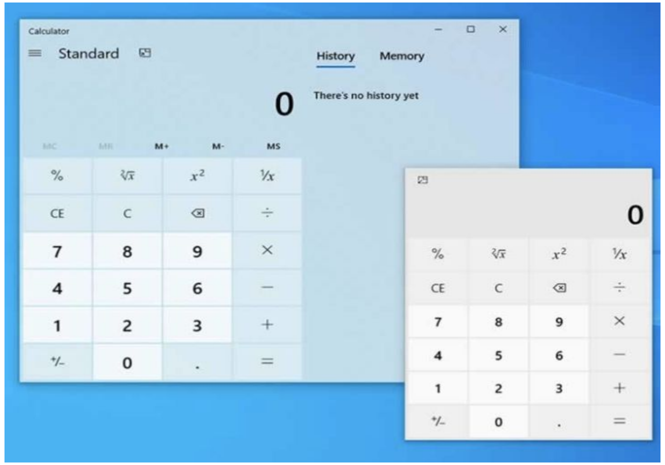
- **SciPy** for advanced computing
- **Pandas** for general-purpose data analysis
- **Seaborn** for data visualization
- **Scikit-learn** for ML
- **NumPy** for high-performance scientific computing and data analysis



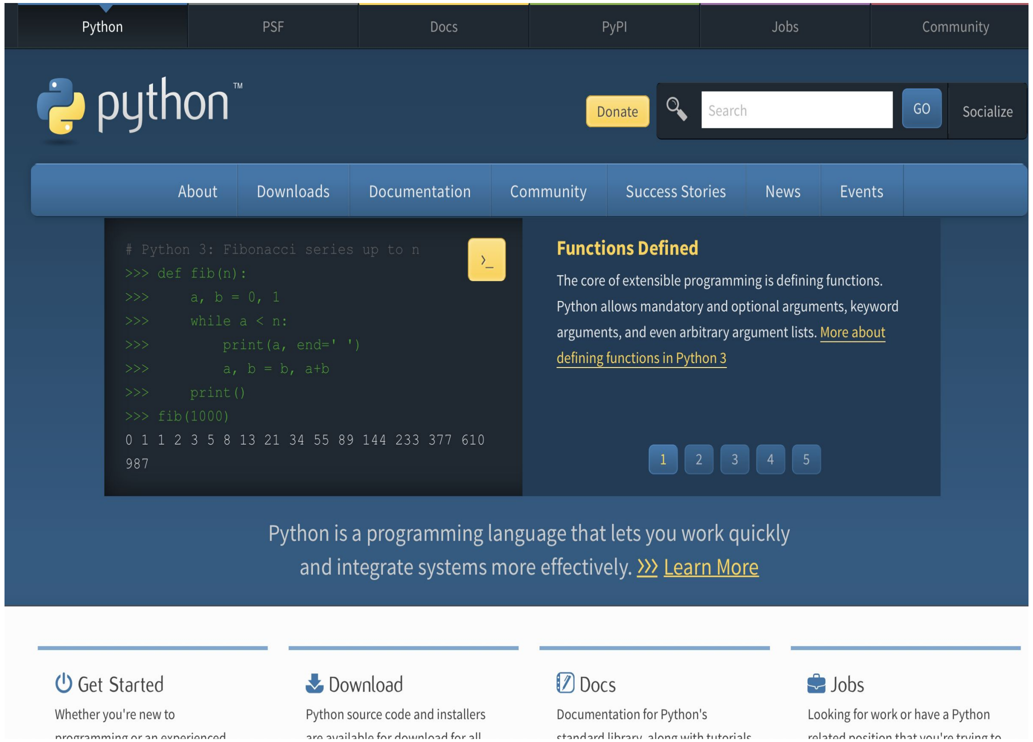
Applications of Python

3. Desktop GUI

Libraries : PyQt, PyGtk, Kivy, Tkinter, WxPython, PyGUI



Install Python on Your Computer



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Get Started
Whether you're new to

Download
Python source code and installers

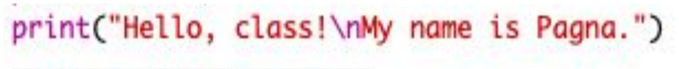
Docs
Documentation for Python's

Jobs
Looking for work or have a Python

Tabs & New lines

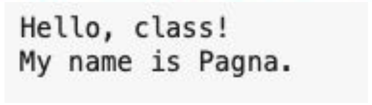
- **New lines:** We use “ \n ” for creating new line.

Code :



```
print("Hello, class!\nMy name is Pagna.")
```

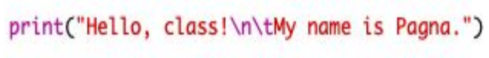
Output :



```
Hello, class!
My name is Pagna.
```

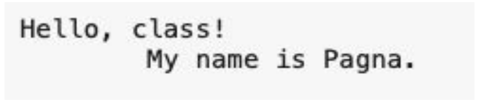
- **Tabs**: We use “ \t ” for tabbing.

Code :



```
print("Hello, class!\n\tMy name is Pagna.")
```

Output :



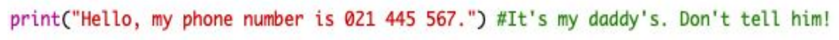
```
Hello, class!
        My name is Pagna.
```

Tabs & New lines

- **Comments:**

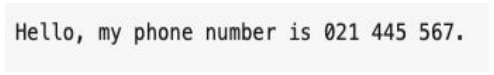
  We use “ # ” or “ ''' ” for commenting.

Code :

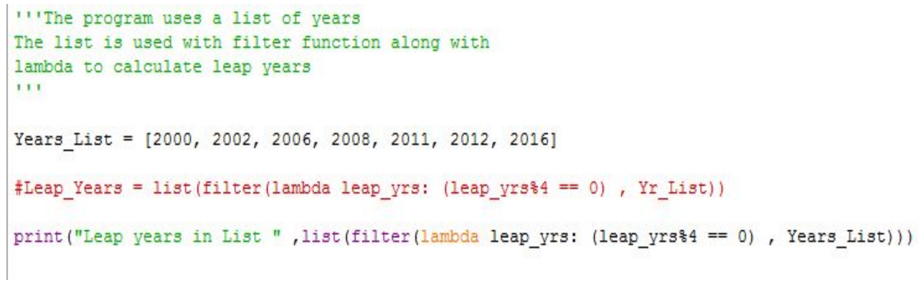


```
print("Hello, my phone number is 021 445 567.") #It's my daddy's. Don't tell him!
```

Output :



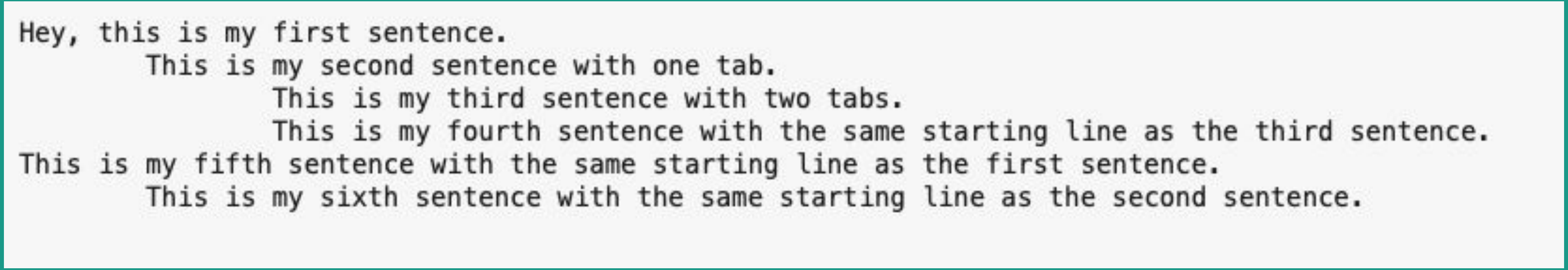
```
Hello, my phone number is 021 445 567.
```



```
'''The program uses a list of years
The list is used with filter function along with
lambda to calculate leap years
'''

Years_List = [2000, 2002, 2006, 2008, 2011, 2012, 2016]

#Leap_Years = list(filter(lambda leap_yrs: (leap_yrs%4 == 0) , Yr_List))

print("Leap years in List " ,list(filter(lambda leap_yrs: (leap_yrs%4 == 0) , Years_List)))
```

Let’s practise !

Print this sentence



```
Hey, this is my first sentence.
        This is my second sentence with one tab.
                This is my third sentence with two tabs.
                This is my fourth sentence with the same starting line as the third sentence.
This is my fifth sentence with the same starting line as the first sentence.
        This is my sixth sentence with the same starting line as the second sentence.
```

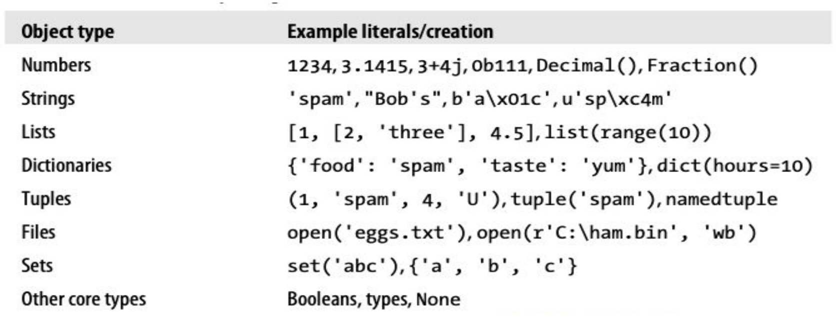
Introducing Python Object Types

Presented by: Bonpagna KANN

Content

1

Types of object in Python



Object type	Example literals/creation
Numbers	`1234`, `3.1415`, `3+4j`, `0b111`, `Decimal()`, `Fraction()`
Strings	`'spam'`, `"Bob's"`, `b'a\x01c'`, `u'sp\xc4m'`
Lists	`[1, [2, 'three'], 4.5]`, `list(range(10))`
Dictionaries	`{'food': 'spam', 'taste': 'yum'}`, `dict(hours=10)`
Tuples	`(1, 'spam', 4, 'U')`, `tuple('spam')`, `namedtuple`
Files	`open('eggs.txt')`, `open(r'C:\ham.bin', 'wb')`
Sets	`set('abc')`, `{'a', 'b', 'c'}`
Other core types	Booleans, types, None

Numbers

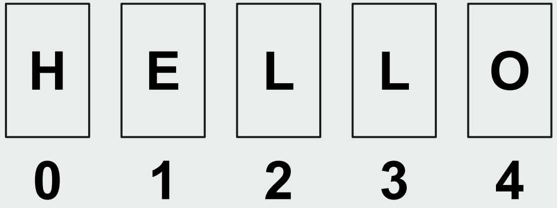
1. Basic arithmetics

```
# Integer addition
>>> 123 + 222
345


# Floating-point multiplication
>>> 1.5 * 4
6.0


# 2 to the power 55
>>> 2 ** 55
36028797018963968
```

2. Numeric Modules

```
>>> import math
>>> math.pi
3.141592653589793

>>> math.sqrt(81)
9
```

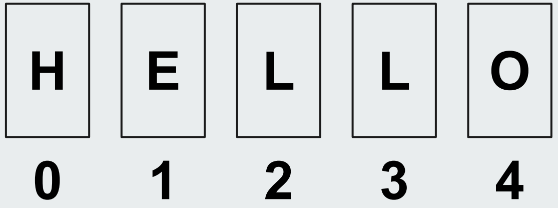
Strings



1. String Operations

```
# A 5-character string 'HELLO'
>>> S = 'HELLO'


#Check the length of the string
>>> len(S)
5

#Print the first character of the string
>>> S[0]
'H'

#Print the last character of the string
>>> S[len(S)-1]
'O'

# Substring of S from second character to the third one.
>>> S[0:4]

>>> S[1:]

>>> S[:3]
```

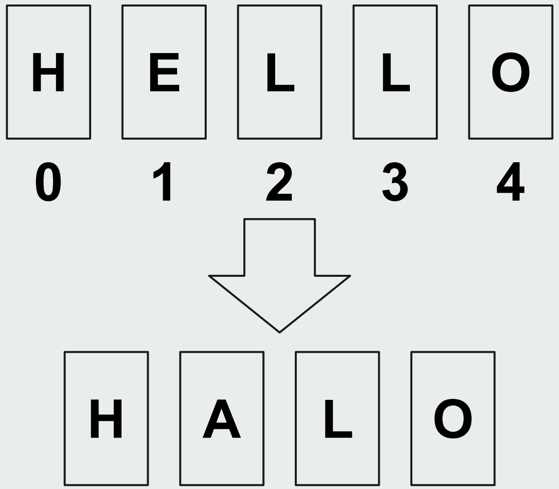
Strings



2. String Operations

```
>>> S = 'HELLO '

# Concatenation
>>> S + 'Python'
'HELLO Python'

# Repetition
>>> S * 5
'HELLO HELLO HELLO HELLO HELLO'

>>> S = 'Programming'

# Expand to a list: [...]
>>> L = list(S)
>>> L

['P', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g']

# Change it in place
>>> L[1] = 'c'

# Join with empty delimiter
>>> ''.join(L)
'Pcogramming'
```

Strings

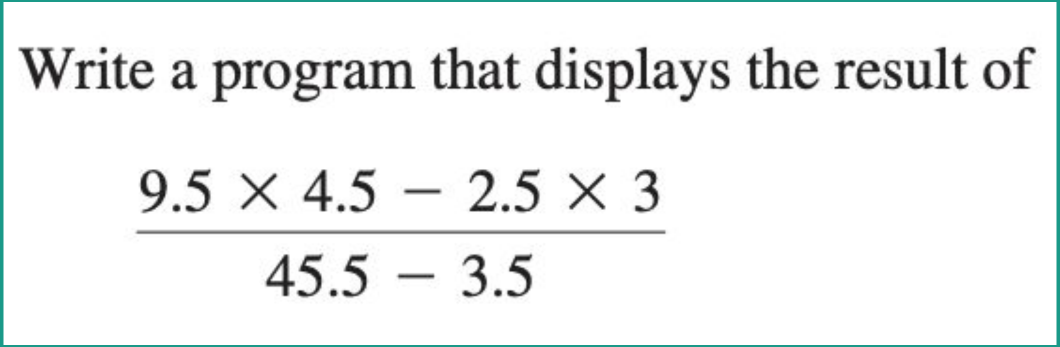


3. String Operations

```
# Replace occurrences of a string.
>>> L=S.replace('EL', 'A')
'HALO'

>>> line = 'aaa,bbb,ccccc,dd'
>>> line.split(',')   #delimiter
['aaa', 'bbb', 'ccccc', 'dd']

>>> S.upper()
'HELLO'

>>> line = 'aaa,bbb,ccccc,dd\n'

>>> line.rstrip()
'aaa,bbb,ccccc,dd'

>>> line.rstrip().split(',')
['aaa', 'bbb', 'ccccc', 'dd']

#print the variable with the message.
>>>print ('%s, Paragon' % S.upper())
'HELLO, Paragon'
>>>A = 'IU'
>>>print('%s, Paragon %s' % (S, A))
>>>print(S, ', Paragon' , A)
```

Let’s practise !

1



Write a program that displays the result of

$$\frac{9.5 \times 4.5 - 2.5 \times 3}{45.5 - 3.5}$$

2

Translate the following algorithm into Python code:

Step 1: Use a variable named “fahrenheit” with initial value 212.

Step 2: Subtract the value by 32. Then multiply by 100/180. Assign it to a variable named “celsius”.

Step 3: Display the value of Celsius. What is value of celsius after Step 3?

Reading input from user

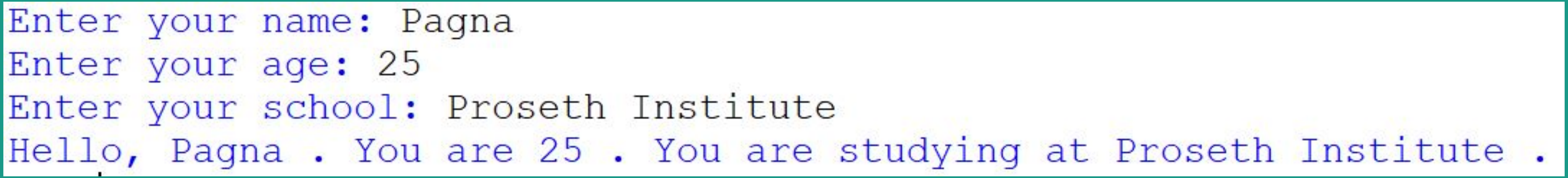
```
x = input("Enter your name: ")
print("Hello, ", x)
```

```
x = input("Enter your number: ")
# input value = 6
result = x * 4
print(result)  # result = 6666
```

```
x = eval(input("Enter your number: "))
# input value = 6
result1 = x // 4
Result2 = x%4
print(result1)   #result = 1
print(result2)          # result = 2
```

3

Write a program to get the input information from user and display it as follows:

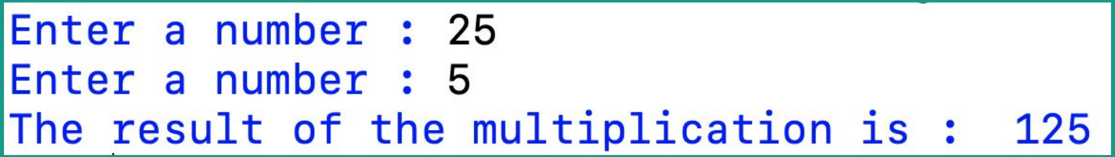


```
Enter your name: Pagna
Enter your age: 25
Enter your school: Proseth Institute
Hello, Pagna . You are 25 . You are studying at Proseth Institute .
```

4

Write a Python Program to:

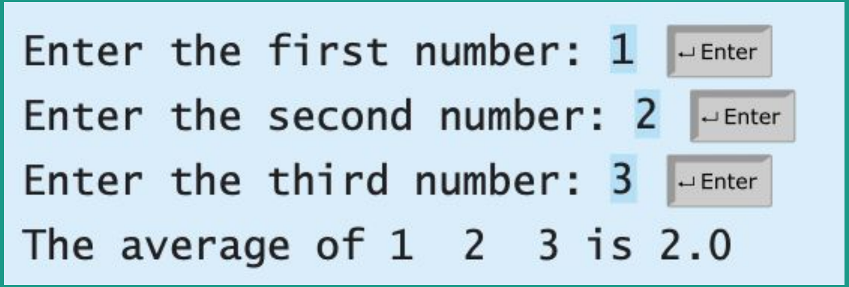
- Get two integers from users.
- Return the result of multiplication of both values



```
Enter a number : 25
Enter a number : 5
The result of the multiplication is :  125
```

5

Write a program to calculate the average of three numbers which are obtained from user. The program should print out the result as follows:



```
Enter the first number: 1 ↵Enter
Enter the second number: 2 ↵Enter
Enter the third number: 3 ↵Enter
The average of 1  2  3 is 2.0
```

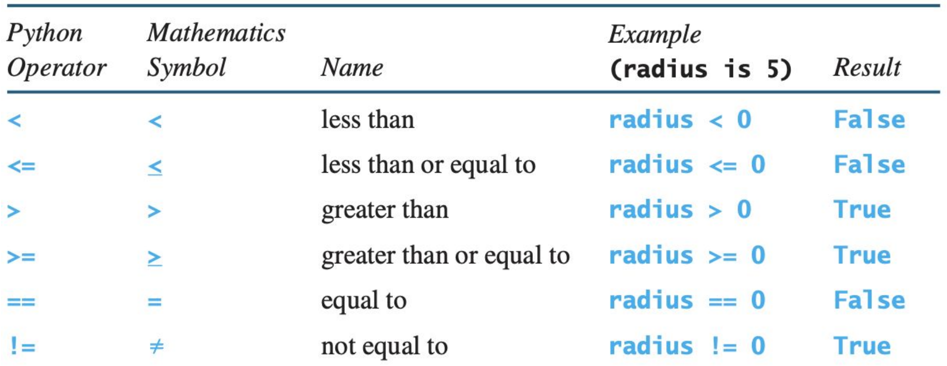
Conditional Structure

Presented by: Bonpagna KANN

Content

1

List of comparison operators



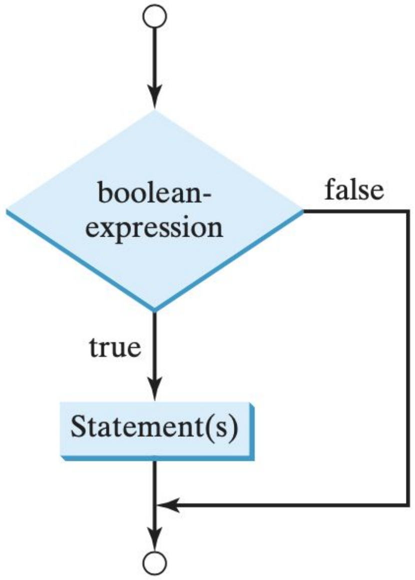
Python Operator	*Mathematics Symbol*	*Name*	*Example* **`(radius is 5)`**	*Result*
`<`	$<$	less than	`radius < 0`	`False`
`<=`	$\leq$	less than or equal to	`radius <= 0`	`False`
`>`	$>$	greater than	`radius > 0`	`True`
`>=`	$\geq$	greater than or equal to	`radius >= 0`	`True`
`==`	$=$	equal to	`radius == 0`	`False`
`!=`	$\neq$	not equal to	`radius != 0`	`True`

2. “If” statement

Condition

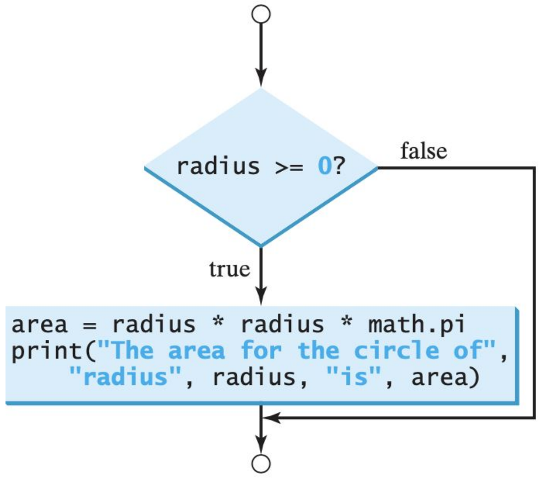
```
if boolean-expression:
    statement(s)-for-the-true-case
```

One indent



2. “If” statement

```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area for the circle of radius", radius, "is", area)
```



```
area = radius * radius * math.pi
print("The area for the circle of",
    "radius", radius, "is", area)
```

Example

The program prompts the user to enter an integer and displays as following:

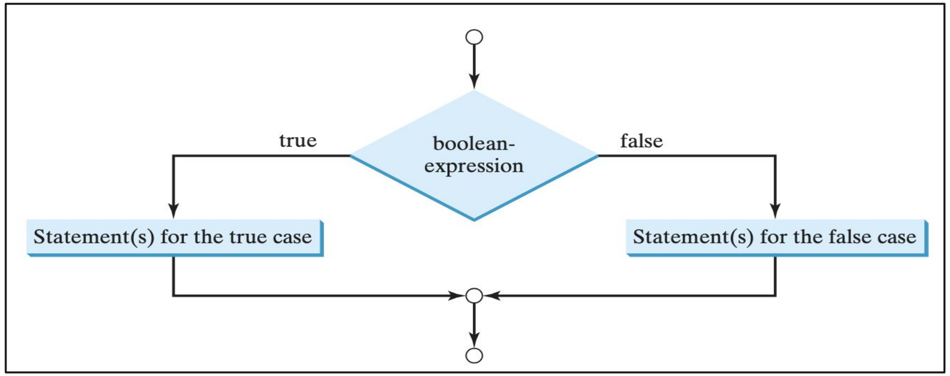
- Print “Hi-Five” if it is divisible by 5
- Print “Hi-Even” if it is divisible by 2.

3. “If-else” statement

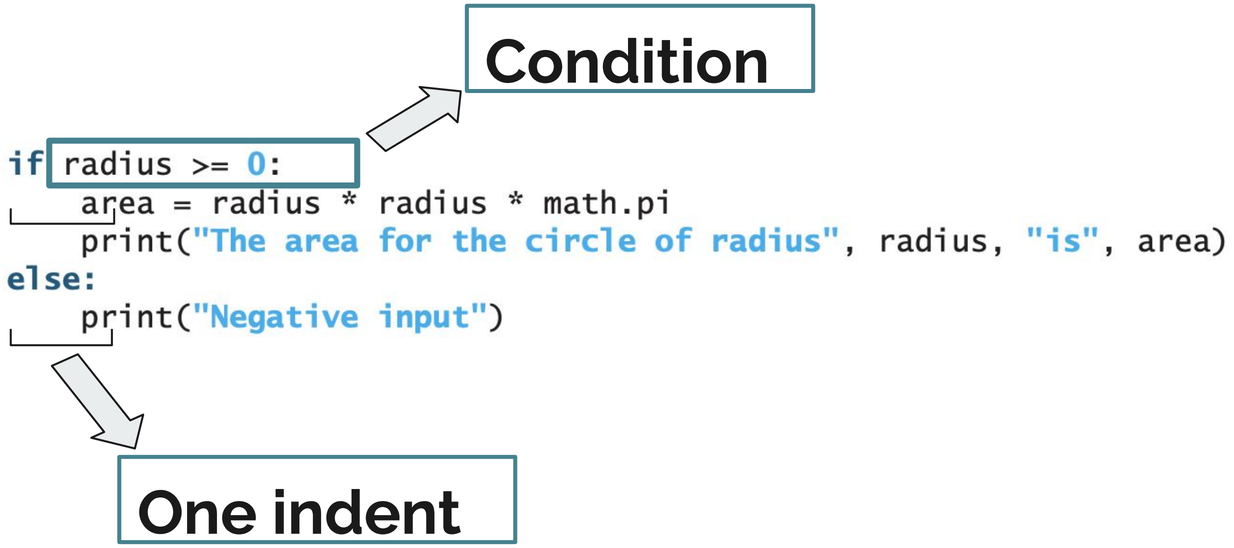
Condition

```
if boolean-expression:
    statement(s)-for-the-true-case
else:
    statement(s)-for-the-false-case
```

One indent


true
boolean-expression
false
Statement(s) for the true case
Statement(s) for the false case

3. “If-else” statement



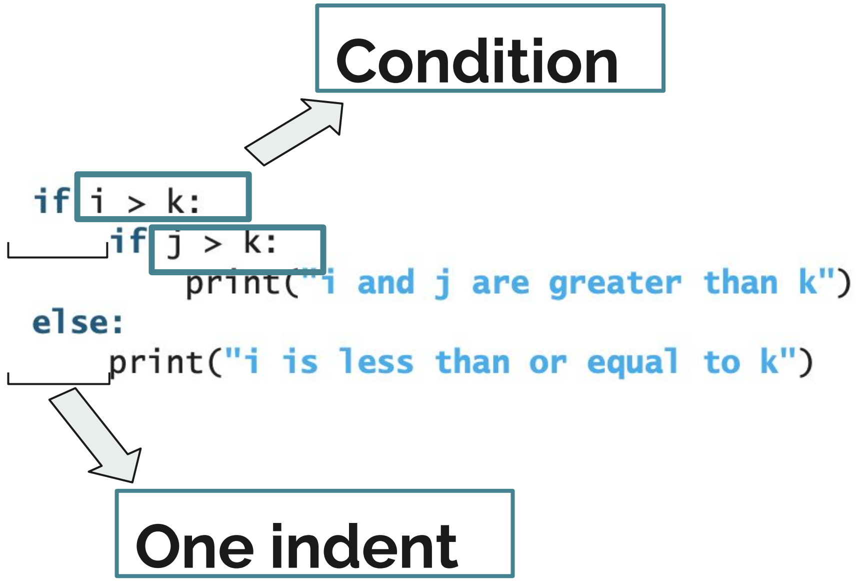
```
if radius >= 0:
    area = radius * radius * math.pi
    print("The area for the circle of radius", radius, "is", area)
else:
    print("Negative input")
```

Example

The program prompts the user to enter an integer and display as following:

- Print “The number is even.” if it is even.
- Print “The number is odd.” if it is odd.

4. “Nested if” Statements



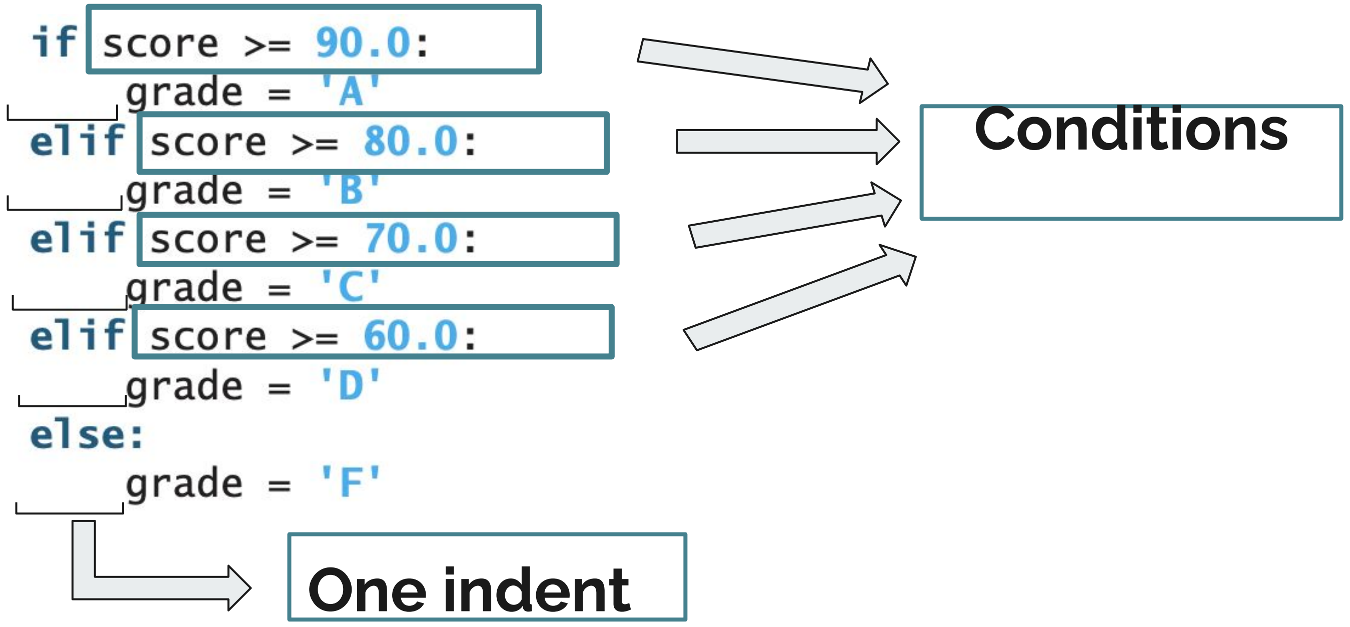
```
if i > k:
    if j > k:
        print("i and j are greater than k")
else:
    print("i is less than or equal to k")
```

Example

Using nested-if statement, write the program prompts the user to enter their age and display as following:

- Print “You need to finish college first.” if their age is under 22.
- Print “It's time to apply for a job, getting out of unemployment.” if their age is from 23 but lower than 30.
- Print “It is just a beginning of the middle age. Don’t be hopeless in life yet.” if their age is from 30.

5. “if-elif-else” Statements

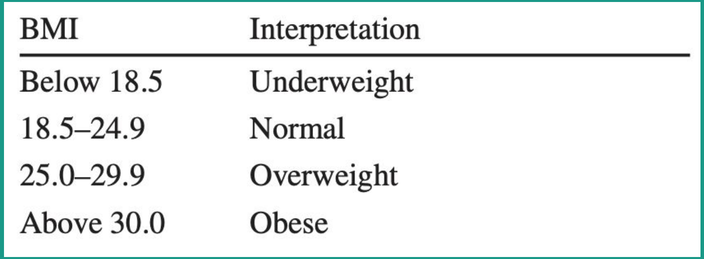

```
if score >= 90.0:
    grade = 'A'
elif score >= 80.0:
    grade = 'B'
elif score >= 70.0:
    grade = 'C'
elif score >= 60.0:
    grade = 'D'
else:
    grade = 'F'
```

Example

- Body mass index (BMI) is a measure of health based on weight.

$$BMI = Weight / (Height * Height)$$

- The interpretation of BMI for people 16 years and older is as follows:



BMI	Interpretation
Below 18.5	Underweight
18.5–24.9	Normal
25.0–29.9	Overweight
Above 30.0	Obese

Write a program that prompts the user to enter a weight (kg) and height (m) and then displays the BMI interpretation.

Iterative Structure (Loop)

Presented by: Bonpagna KANN

Content

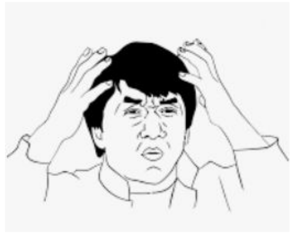
1

Introduction to Loop statement

"**Loop**" can be used to tell a program to execute statements repeatedly.

```
print("Programming is fun!")
```

```
           ⎧ print("Programming is fun!")
           ⎪ print("Programming is fun!")
100 times  ⎨ ...
           ⎩ print("Programming is fun!")
```



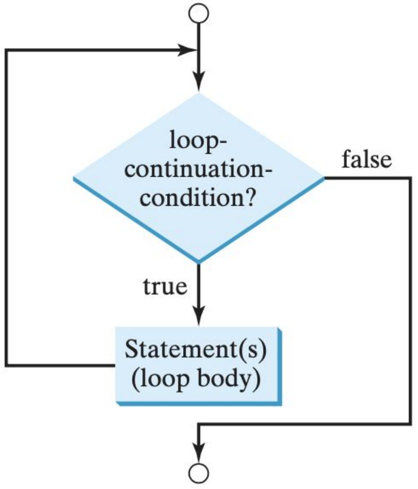
```
count = 0
while count < 100:
    print("Programming is fun!")
    count = count + 1
```

2. "While" statement

```
while loop-continuation-condition:
    # Loop body
    Statement(s)
```

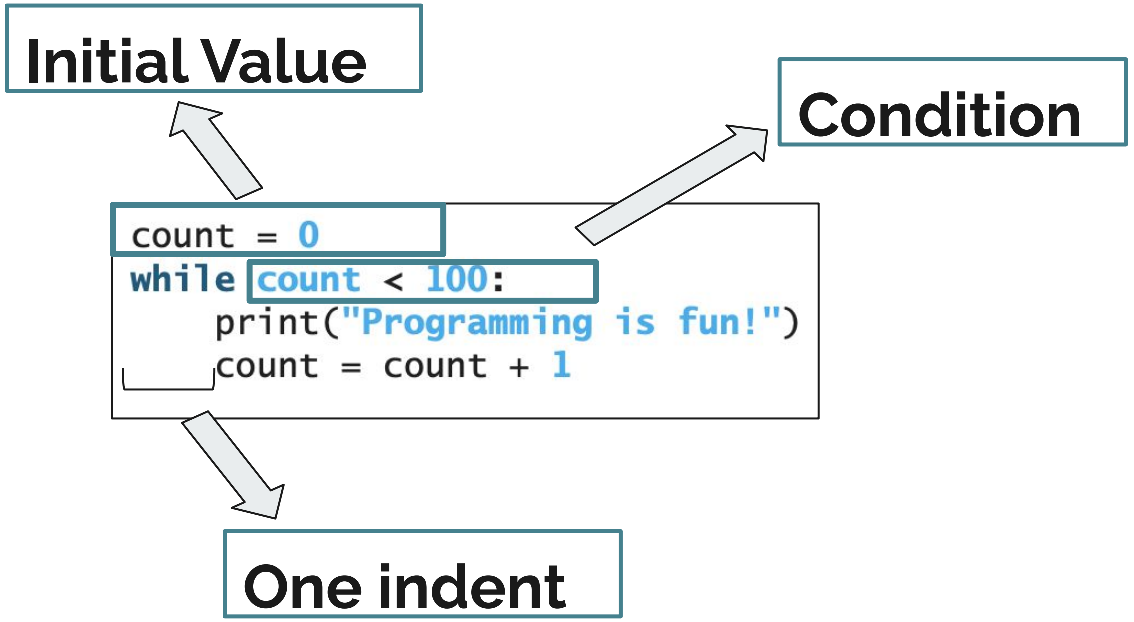
Condition

One indent

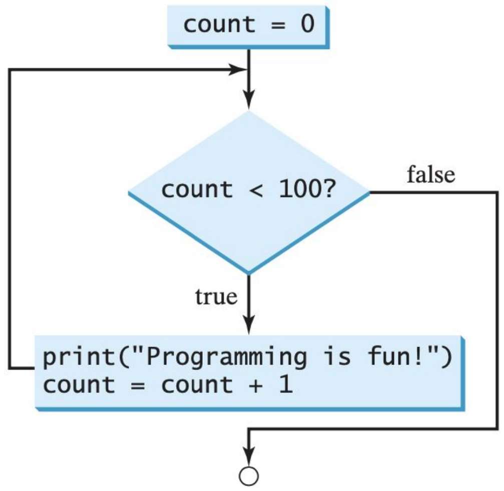


(a) A while loop

2. “While” statement

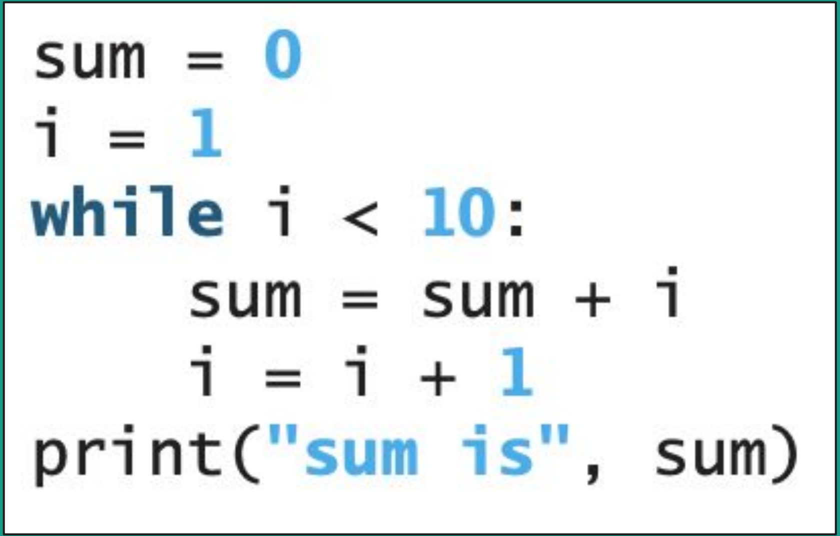


```python
count = 0
while count < 100:
    print("Programming is fun!")
    count = count + 1
```



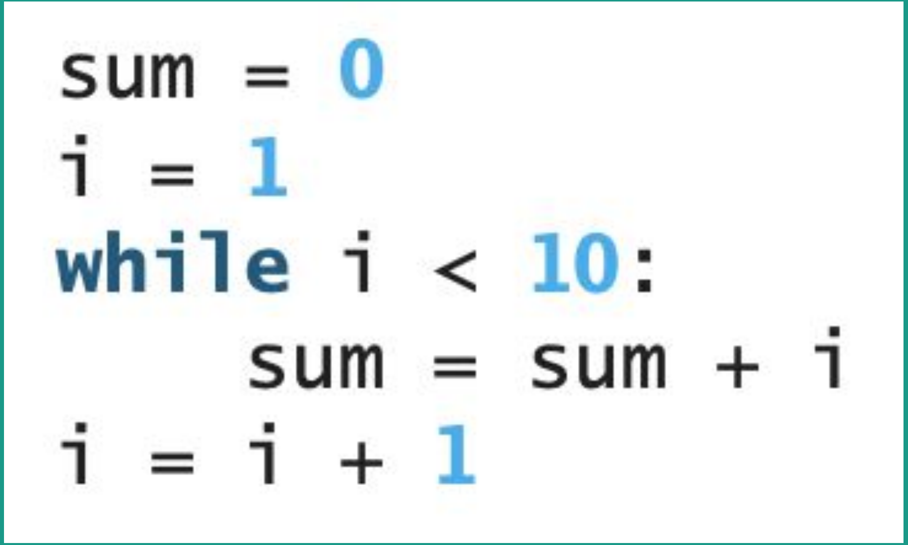
(b) A while loop example

Example



```
sum = 0
i = 1
while i < 10:
    sum = sum + i
    i = i + 1
print("sum is", sum)
```

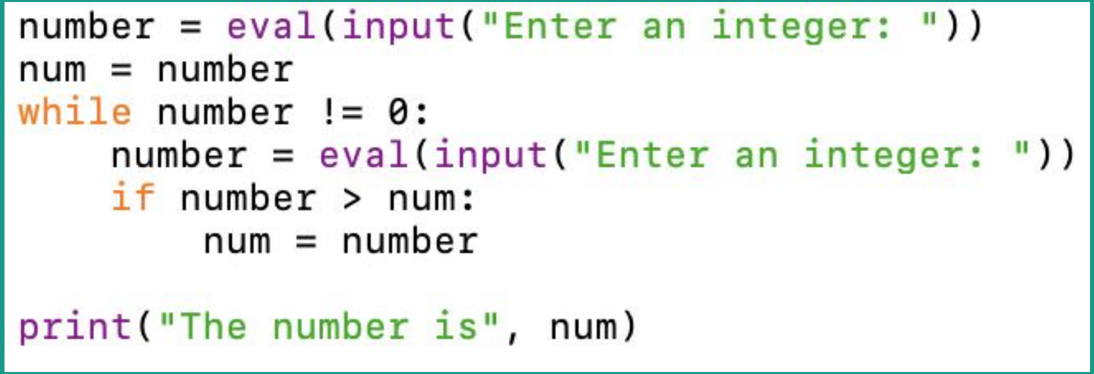
Example



```
sum = 0
i = 1
while i < 10:
    sum = sum + i
i = i + 1
```

Example

What is the output of this code?



```python
number = eval(input("Enter an integer: "))
num = number
while number != 0:
    number = eval(input("Enter an integer: "))
    if number > num:
        num = number

print("The number is", num)
```

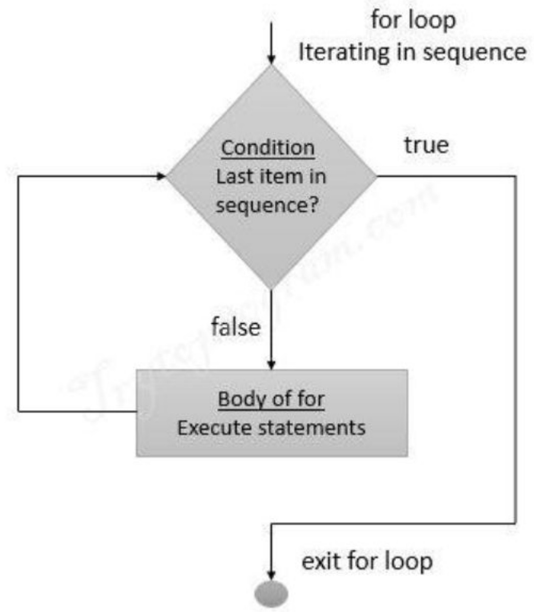
3. “For” statement

```
i = initialValue  # Initialize loop-control variable
while i < endValue:
    # Loop body
    ...
    i += 1  # Adjust loop-control variable
```

```
for var in sequence:
    # Loop body
```

```
for i in range(initialValue, endValue):
    # Loop body
```

One indent



3. “For” statement

```
for v in range(5):
    print(v)
```

```
0
1
2
3
4
```

```
for v in range(4, 8):
    print(v)
```

One indent

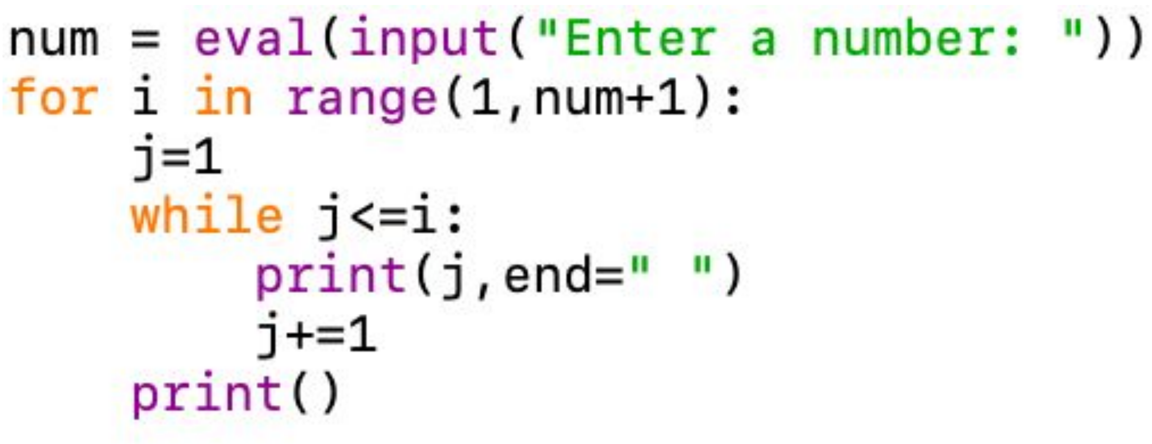
```
4
5
6
7
```

```
for v in range(3, 9, 2):
    print(v)
```

One indent

```
3
5
7
```

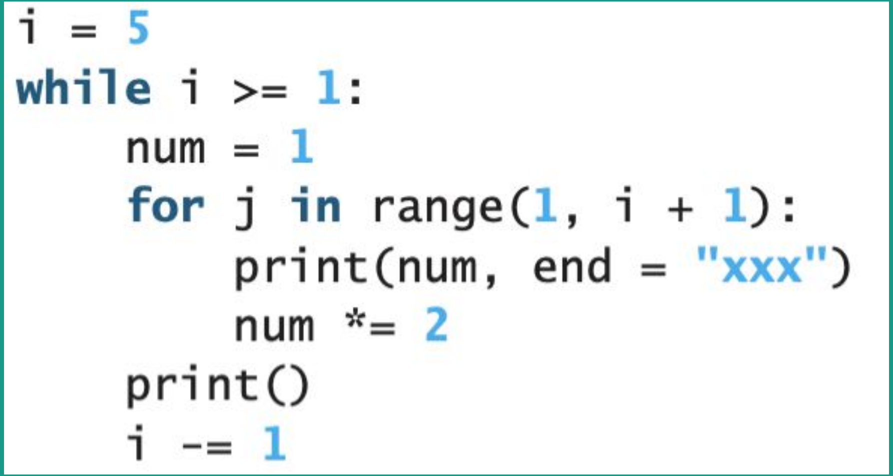
4. “Nested loop” Statements



```
num = eval(input("Enter a number: "))
for i in range(1,num+1):
    j=1
    while j<=i:
        print(j,end=" ")
        j+=1
    print()
```

Example

- Show the output of this program



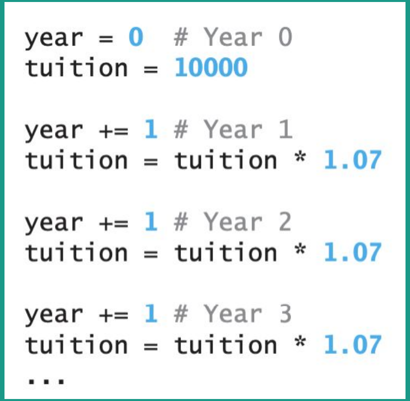
```
i = 5
while i >= 1:
    num = 1
    for j in range(1, i + 1):
        print(num, end = "xxx")
        num *= 2
    print()
    i -= 1
```

Example

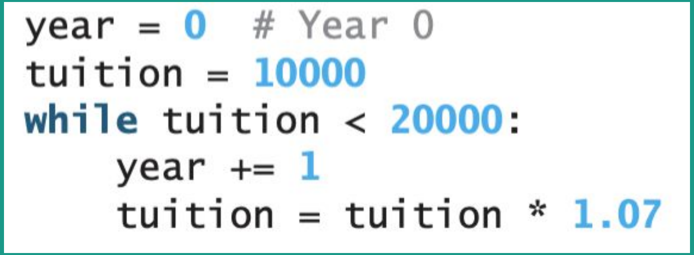
Suppose that the tuition for a university is $10,000 this year and increases 7% every year. In how many years will the tuition have doubled?

Example

Suppose that the tuition for a university is $10,000 this year and increases 7% every year. In how many years will the tuition have doubled?

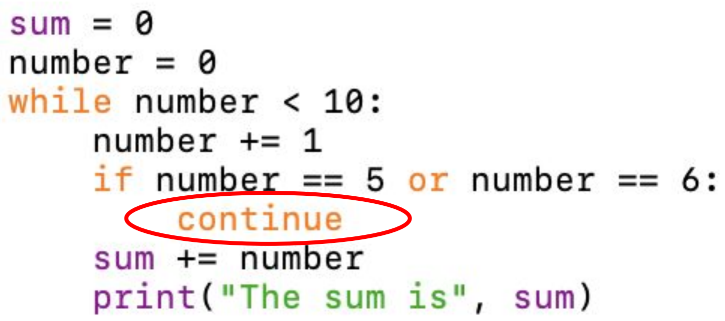


```
year = 0   # Year 0
tuition = 10000

year += 1 # Year 1
tuition = tuition * 1.07

year += 1 # Year 2
tuition = tuition * 1.07

year += 1 # Year 3
tuition = tuition * 1.07
...
```



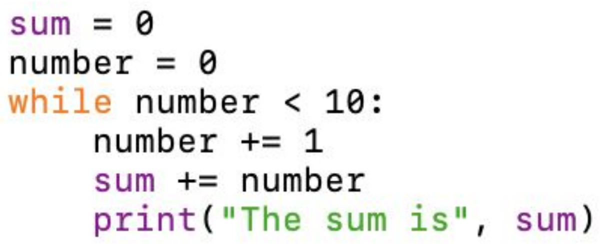
```
year = 0   # Year 0
tuition = 10000
while tuition < 20000:
    year += 1
    tuition = tuition * 1.07
```

3. “Break” and “continue” keywords



```python
sum = 0
number = 0
while number < 10:
     number += 1
     if number == 5 or number == 6:
          continue
     sum += number
     print("The sum is", sum)
```

How about this one?



```python
sum = 0
number = 0
while number < 10:
     number += 1
     sum += number
     print("The sum is", sum)
```

3. “Break” and “continue” keywords

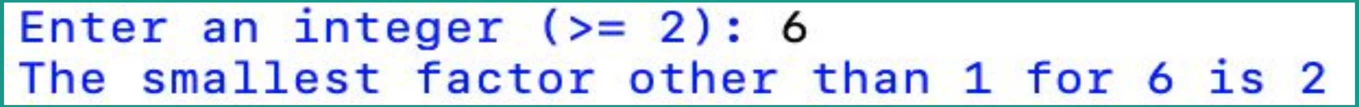
```
sum = 0
number = 0

while number < 20:
    number += 1
    sum += number
    if sum >= 100:
        break

print("The number is", number)
print("The sum is", sum)
```

How about this one?

```
sum=0
number = 0
while number < 20:
    number += 1
    sum += number
print("The number is", number)
print("The sum is", sum)
```

Example

Write a program to find the smallest factor of an integer >= 2.

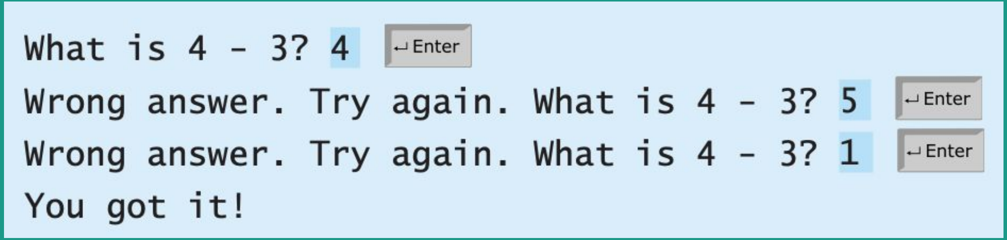


```
Enter an integer (>= 2): 6
The smallest factor other than 1 for 6 is 2
```

Example

- Write a program that prompts the user to enter an answer for a question on subtraction.

- Using a loop, you can now rewrite the program to let the user enter a new answer until it is correct, as shown below:



```
What is 4 - 3? 4 ↵Enter
Wrong answer. Try again. What is 4 - 3? 5 ↵Enter
Wrong answer. Try again. What is 4 - 3? 1 ↵Enter
You got it!
```

Function

Presented by: Bonpagna KANN

Content

Find the sum of integers from “1 to 10”, “20 to 37”, and “35 to 49”

```python
sum = 0
for i in range(1, 11):
    sum += i
print("Sum from 1 to 10 is", sum)
```

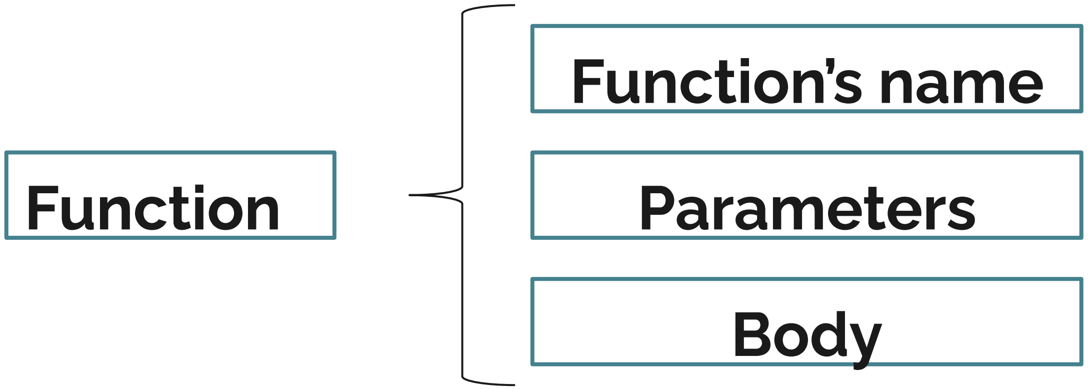
```python
sum = 0
for i in range(20, 38):
    sum += i
print("Sum from 20 to 37 is", sum)
```

```python
sum = 0
for i in range(35, 50):
    sum += i
print("Sum from 35 to 49 is", sum)
```

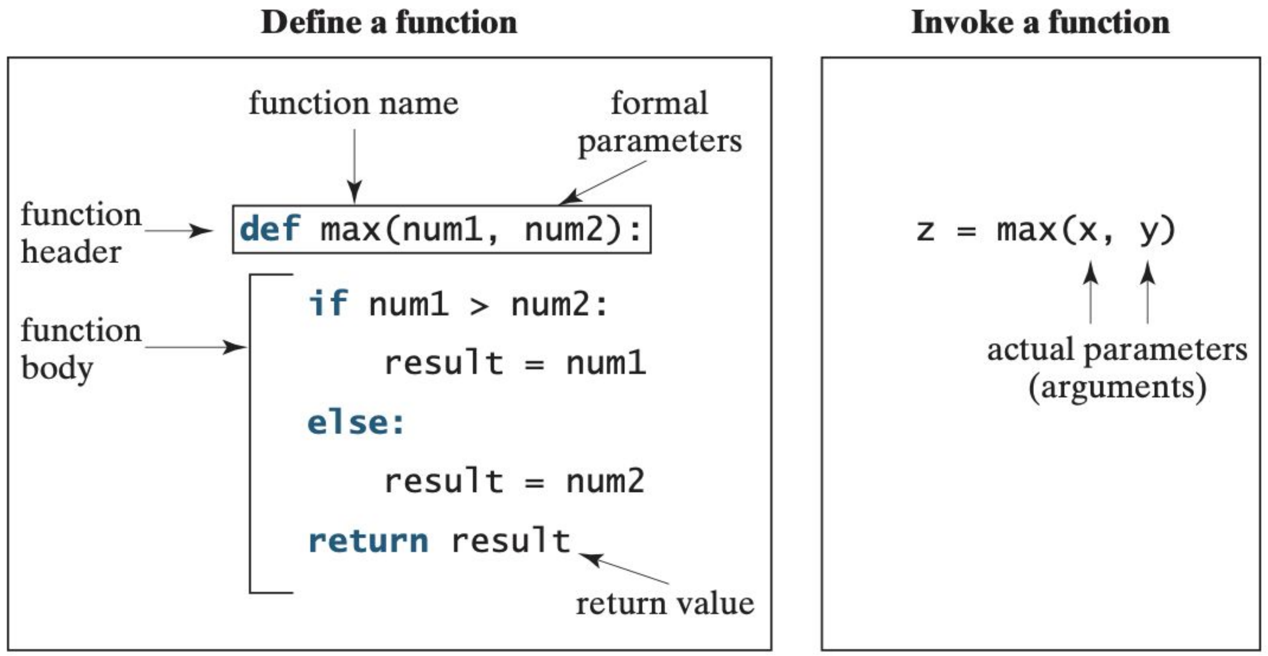
```python
def sum(i1, i2):
    result = 0
    for i in range(i1, i2 + 1):
        result += i

    return result

def main():
    print("Sum from 1 to 10 is", sum(1, 10))
    print("Sum from 20 to 37 is", sum(20, 37))
    print("Sum from 35 to 49 is", sum(35, 49))

main() # Call the main function
```

What is function?

Functions can be used to define **reusable code** and organize and simplify code.



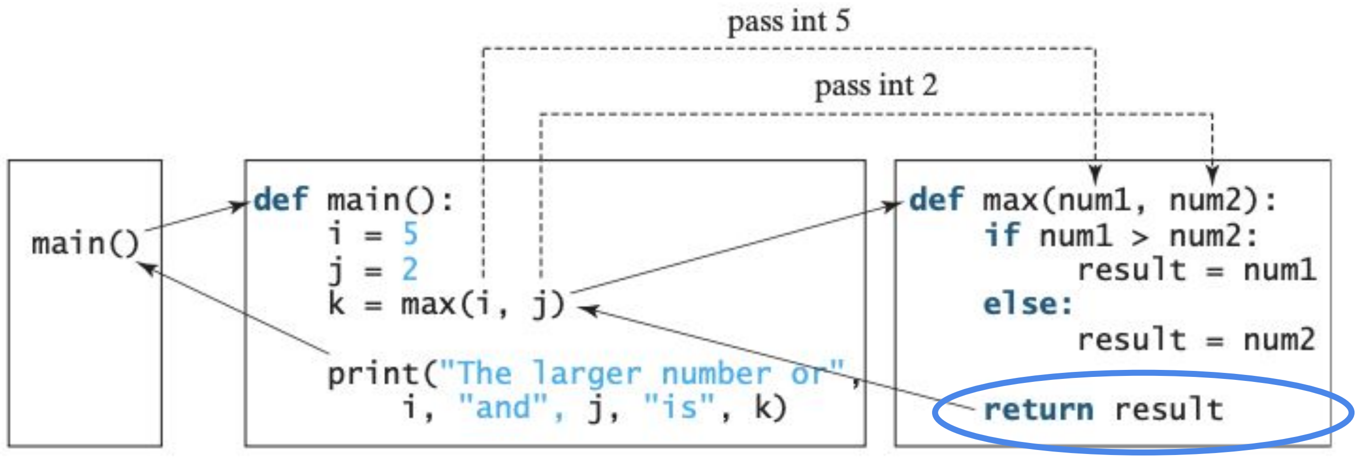
Define and Invoke function



```
def max(num1, num2):
    if num1 > num2:
        result = num1
    else:
        result = num2
    return result
```

```
z = max(x, y)
```

How function works?



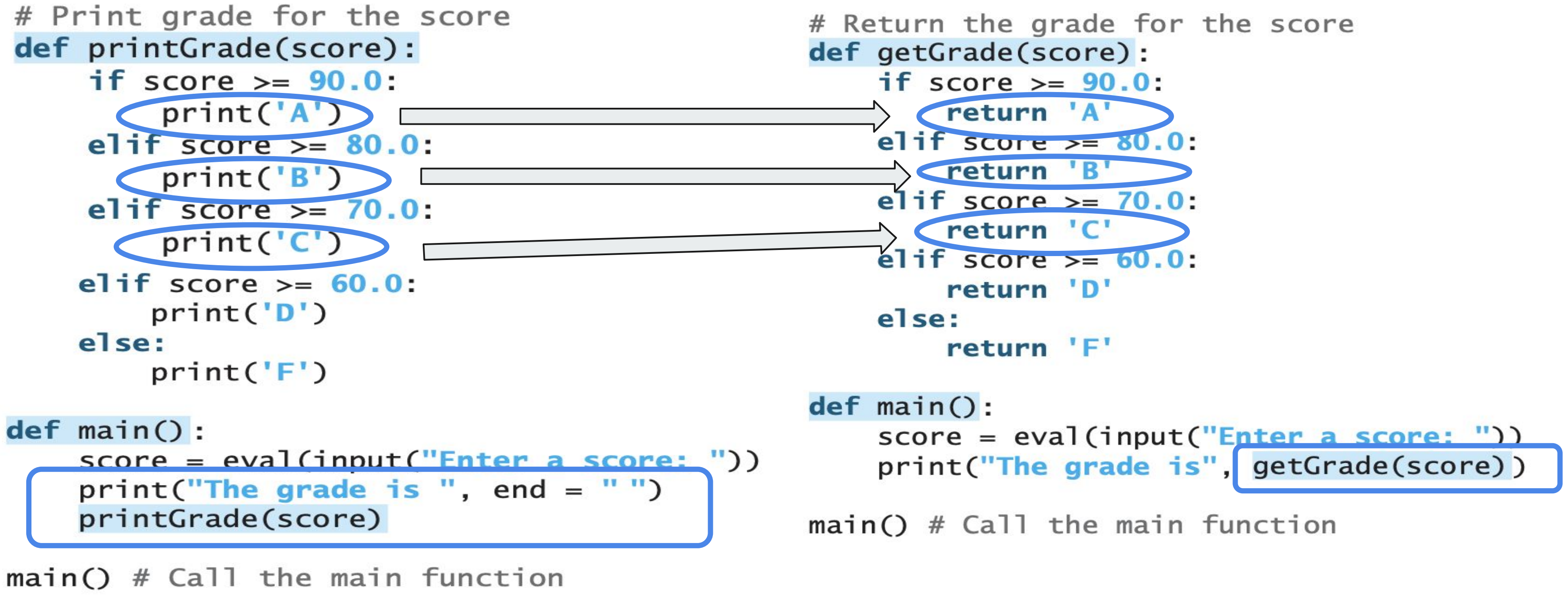
```
def main():
    i = 5
    j = 2
    k = max(i, j)

    print("The larger number of",
        i, "and", j, "is", k)
```

```
def max(num1, num2):
    if num1 > num2:
        result = num1
    else:
        result = num2

    return result
```

??

return a **value**

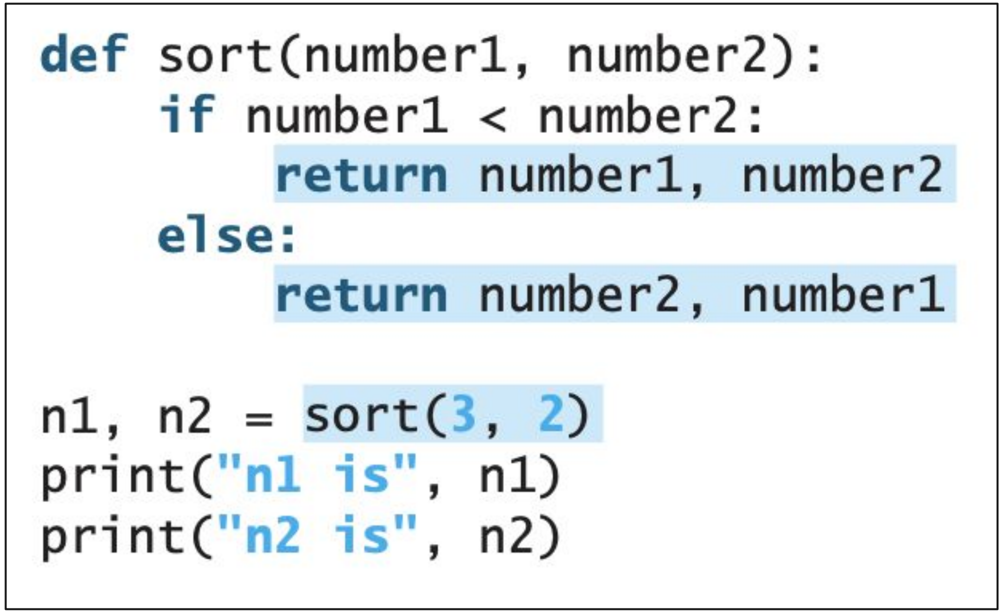
How function works?



```
# Print grade for the score
def printGrade(score):
    if score >= 90.0:
        print('A')
    elif score >= 80.0:
        print('B')
    elif score >= 70.0:
        print('C')
    elif score >= 60.0:
        print('D')
    else:
        print('F')

def main():
    score = eval(input("Enter a score: "))
    print("The grade is ", end = " ")
    printGrade(score)

main() # Call the main function
```

```
# Return the grade for the score
def getGrade(score):
    if score >= 90.0:
        return 'A'
    elif score >= 80.0:
        return 'B'
    elif score >= 70.0:
        return 'C'
    elif score >= 60.0:
        return 'D'
    else:
        return 'F'

def main():
    score = eval(input("Enter a score: "))
    print("The grade is", getGrade(score))

main() # Call the main function
```

Returning multiple values



```python
def sort(number1, number2):
    if number1 < number2:
        return number1, number2
    else:
        return number2, number1

n1, n2 = sort(3, 2)
print("n1 is", n1)
print("n2 is", n2)
```

Scope of variable

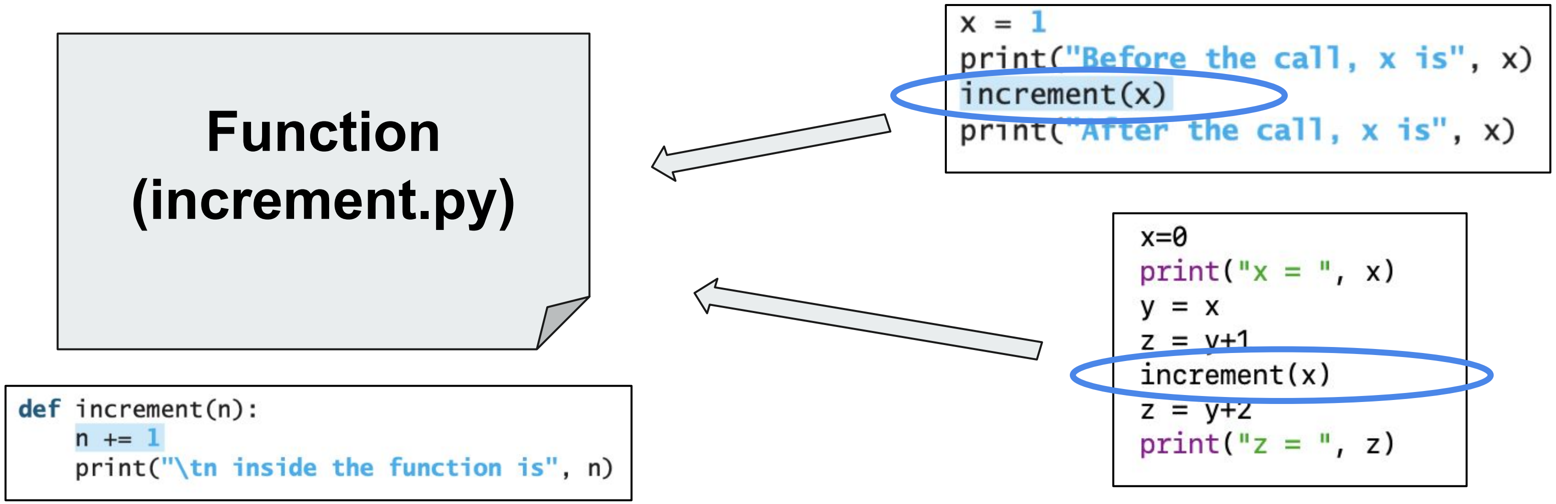
```python
def main():
    x = 1
    print("Before the call, x is", x)
    increment(x)
    print("After the call, x is", x)

def increment(n):
    n += 1
    print("\tn inside the function is", n)

main() # Call the main function
```

```python
x = 1
def increase():
    global x
    x =  x + 1
    print(x) # Displays 2

increase()
print(x) # Displays 2
```

Code Modularization

- Modularizing makes code easy to maintain and debug, and enables the code to be reused.
- Functions can be used to reduce redundant code and enable code reuse.
- Functions can also be used to modularize code and improve a

Code Modularization

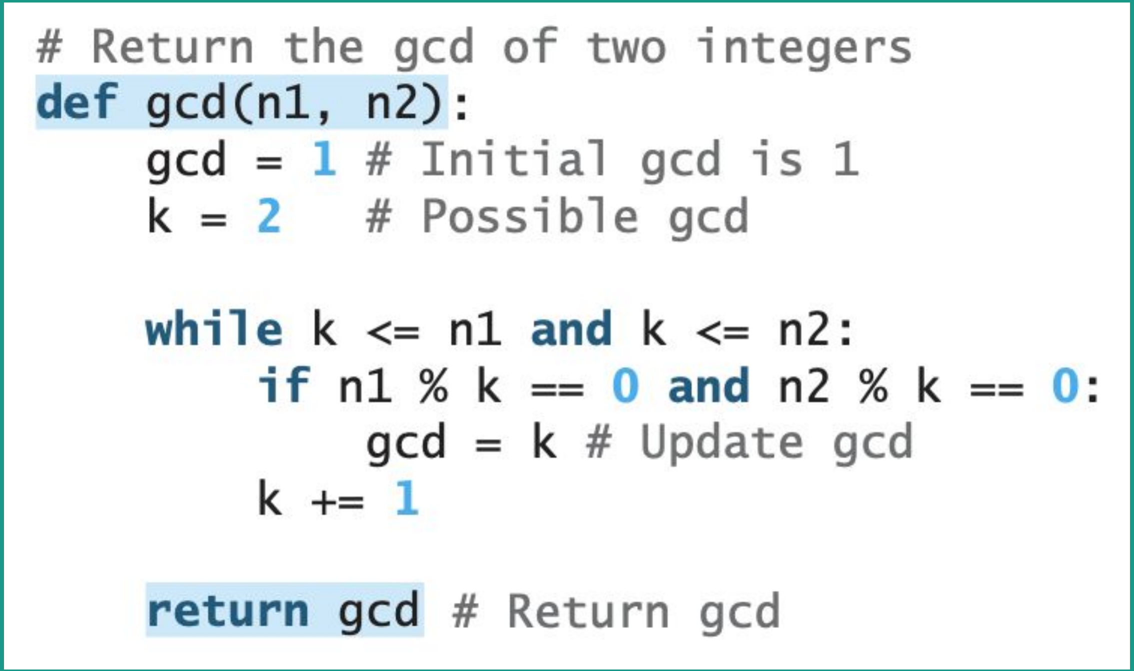


```
x = 1
print("Before the call, x is", x)
increment(x)
print("After the call, x is", x)
```

```
x=0
print("x = ", x)
y = x
z = y+1
increment(x)
z = y+2
print("z = ", z)
```

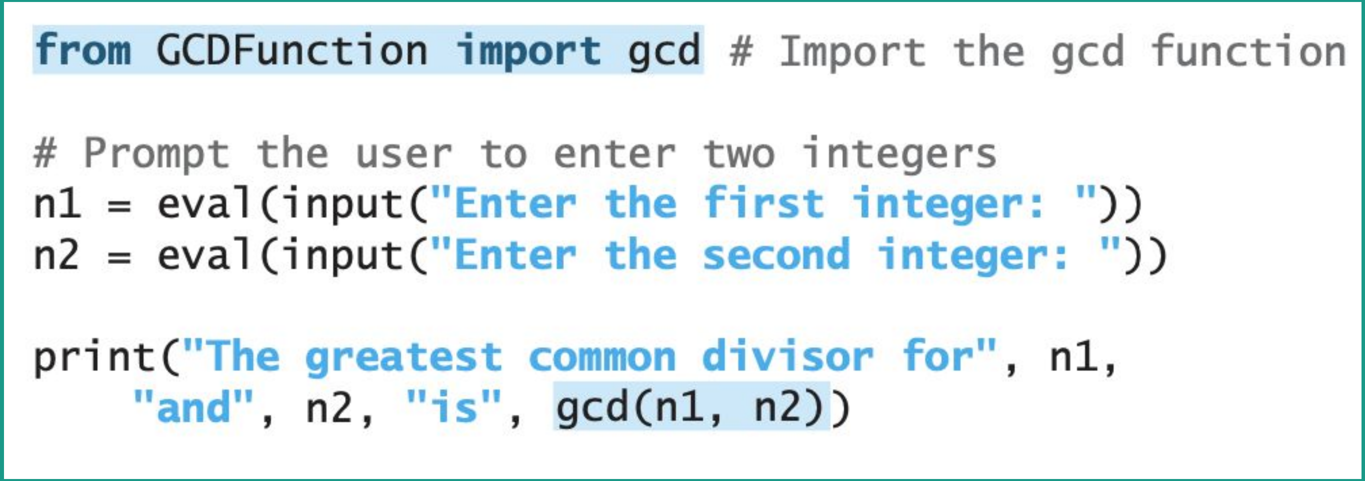
```
def increment(n):
    n += 1
    print("\tn inside the function is", n)
```

Example



```python
# Return the gcd of two integers
def gcd(n1, n2):
    gcd = 1 # Initial gcd is 1
    k = 2   # Possible gcd

    while k <= n1 and k <= n2:
        if n1 % k == 0 and n2 % k == 0:
            gcd = k # Update gcd
        k += 1

    return gcd # Return gcd
```

Example

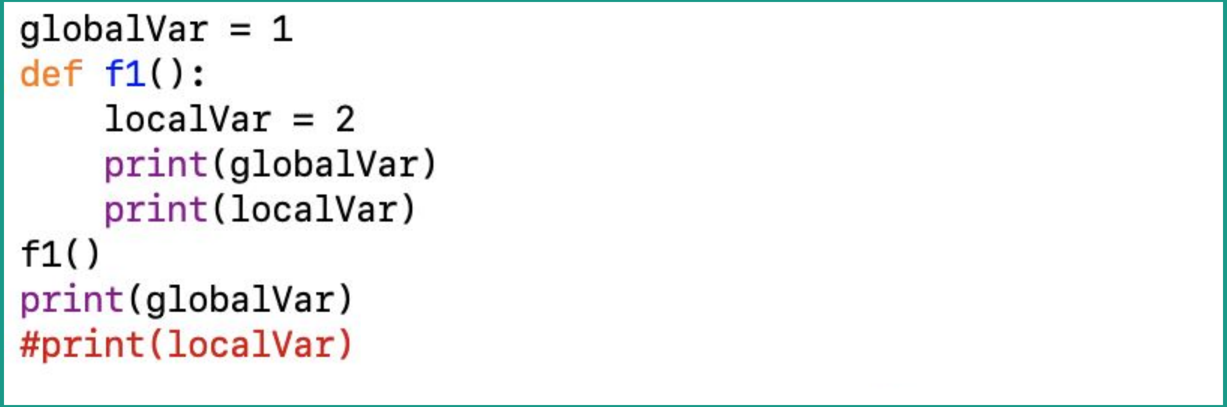


```python
from GCDFunction import gcd # Import the gcd function

# Prompt the user to enter two integers
n1 = eval(input("Enter the first integer: "))
n2 = eval(input("Enter the second integer: "))

print("The greatest common divisor for", n1,
    "and", n2, "is", gcd(n1, n2))
```

Common confusions in Function

```
def main():
    x = 1
    print("Before the call, x is", x)
    increment(x)
    print("After the call, x is", x)

def increment(n):
    n += 1
    print("\tn inside the function is", n)

main() # Call the main function
```

Example



```python
globalVar = 1
def f1():
    localVar = 2
    print(globalVar)
    print(localVar)
f1()
print(globalVar)
#print(localVar)
```

Default Arguments

```python
def printArea(width = 1, height = 2):
    area = width * height
    print("width:", width, "\theight:", height, "\tarea:", area)
```

```python
printArea() # Default arguments width = 1 and height = 2
printArea(4, 2.5) # Positional arguments width = 4 and height = 2.5
printArea(height = 5, width = 3) # Keyword arguments width
printArea(width = 1.2) # Default height = 2
printArea(height = 6.2) # Default width = 1
```

List, Tuple, Set, Dictionary

Presented by: Bonpagna KANN

Anaconda Installation
(for Jupyter Notebook)

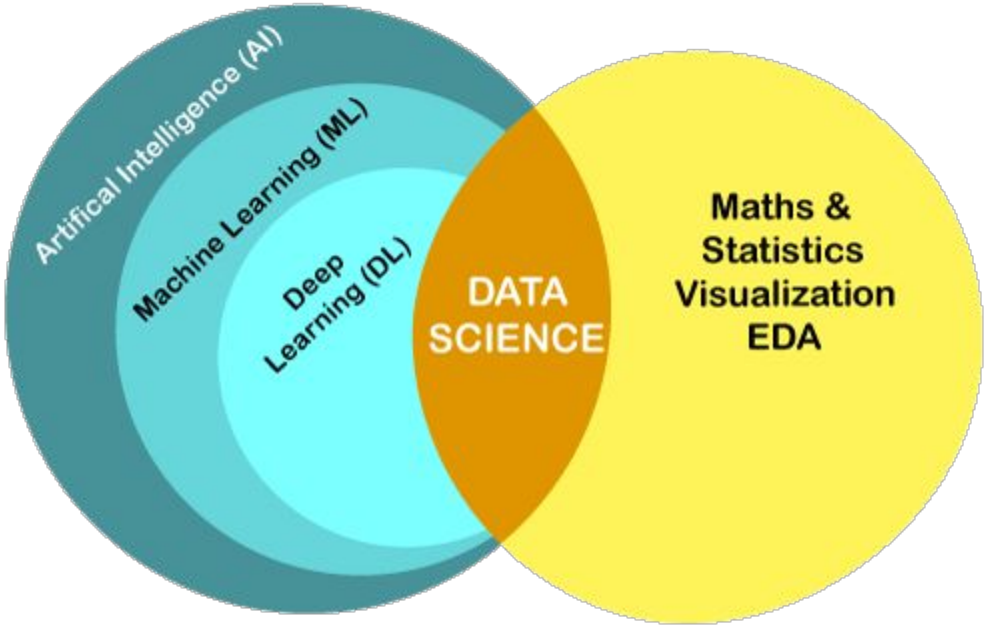
Link to the Lesson Materials and Practices

https://bit.ly/2R29Imf

Python for Data Analysis

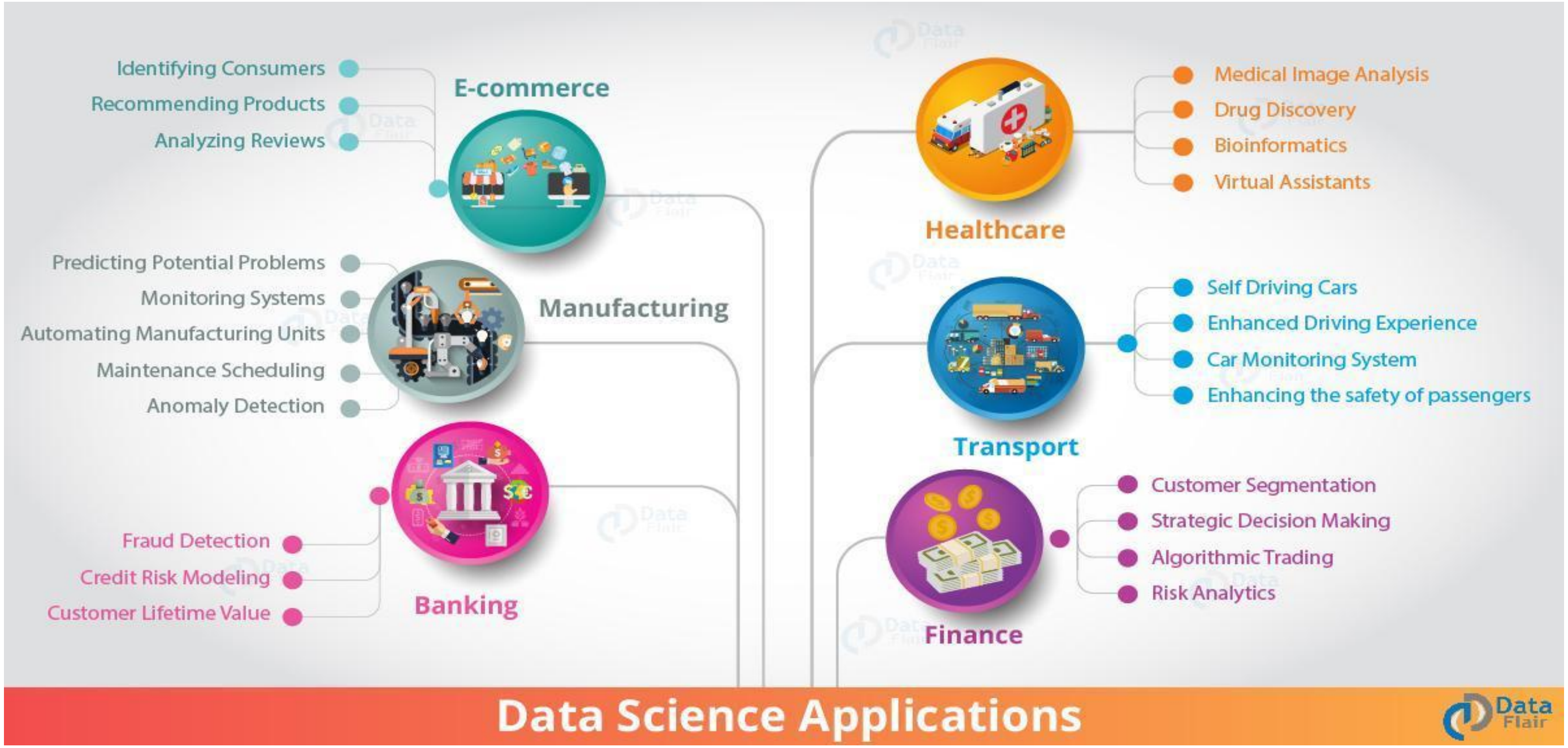
Kann Bonpagna

AI, ML, DL, and Data Science



*** EDA = Exploratory Data Analysis

Data Science Applications



Available Jobs

Data Engineer

Perform tasks of data preparation like data cleaning and organizing. They build data pipeline & architecture, and perform data transformation including cleaning, structuring and formatting the data.

Data Analyst

Managing large sets of data and scrutinizing information by using data analysis tools, curate reports of the analysis and presenting them to the management.

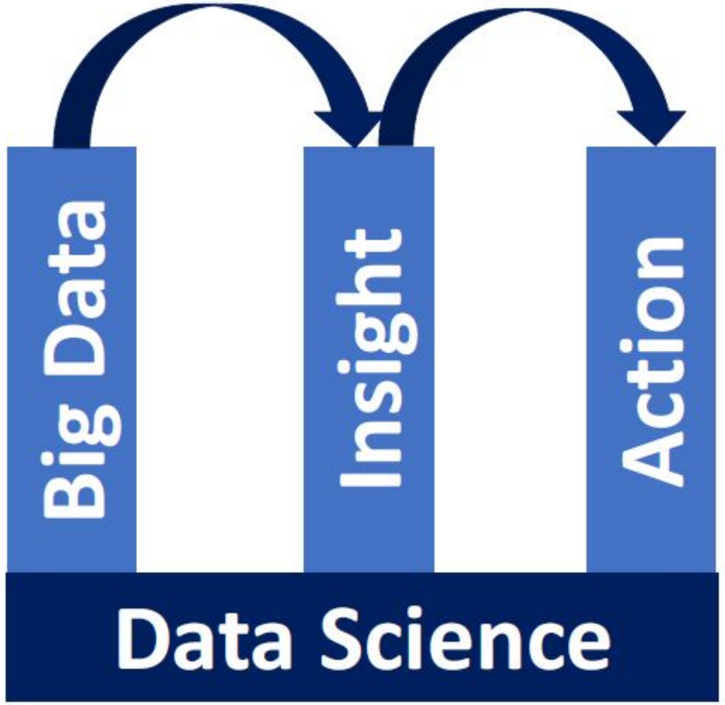
Data Science

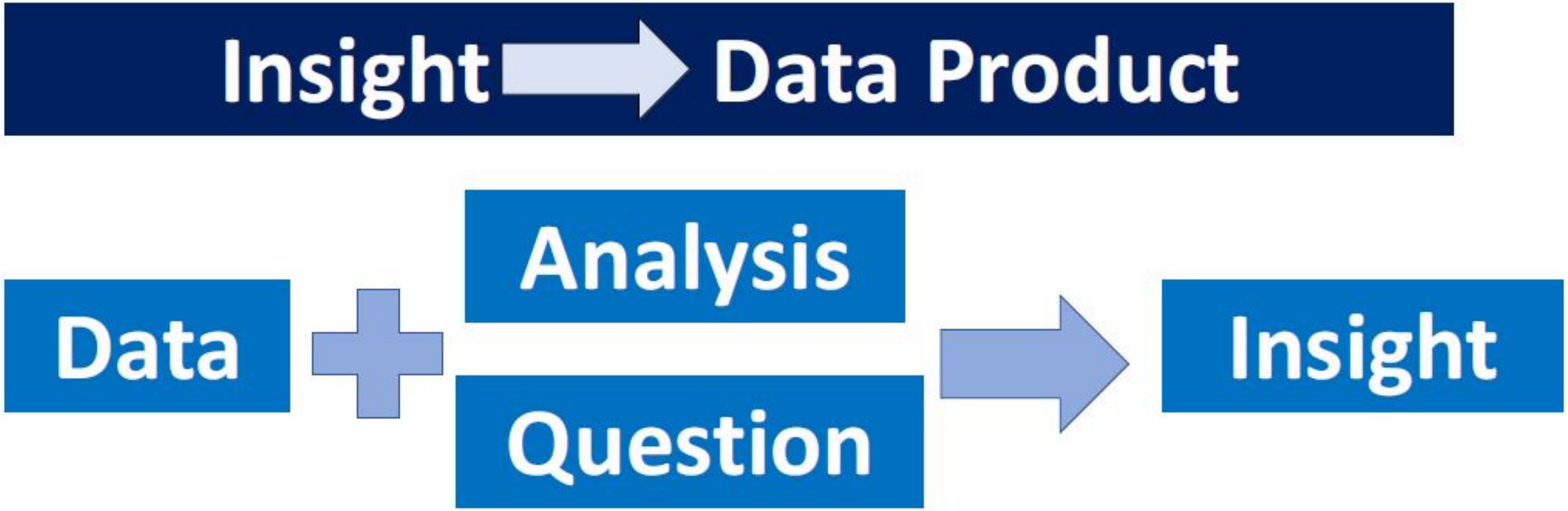
Business Analyst

formulate strategic plans for organizations, ensuring that the required information can be utilized and channelized properly. A Business Intelligence is adept in using BI tools to drive innovation in business by keeping track of and analyzing the market trends.

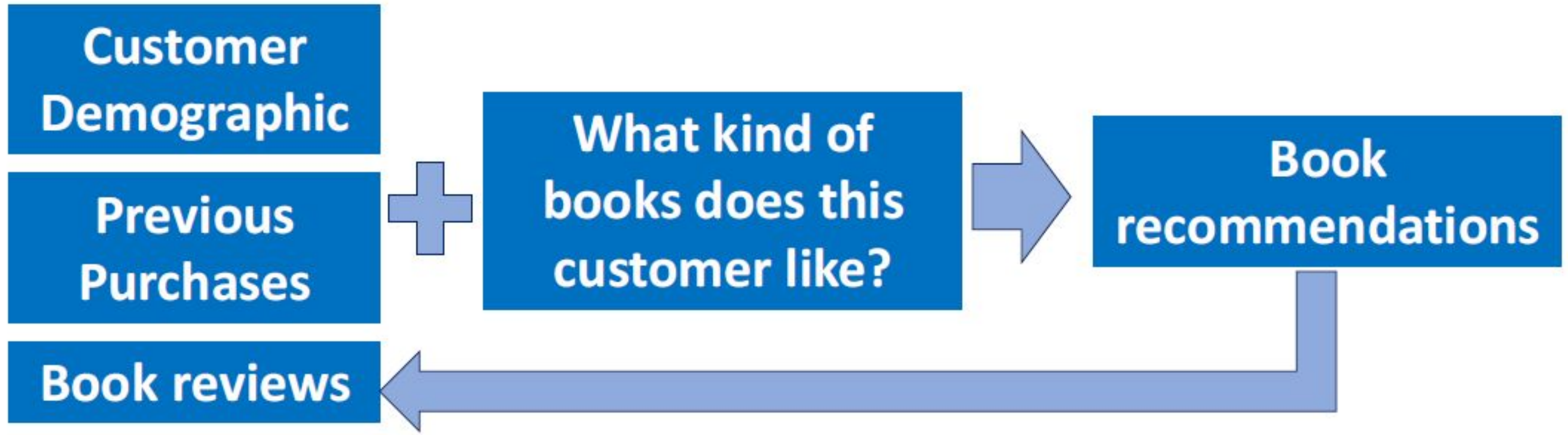
Data Scientist

Perform more technical tasks including data modeling. Data scientists are also responsible for handling huge amounts of data to extricate useful patterns and trends from the data, and build the model to solve the problems.


Big Data
Insight
Action
Data Science

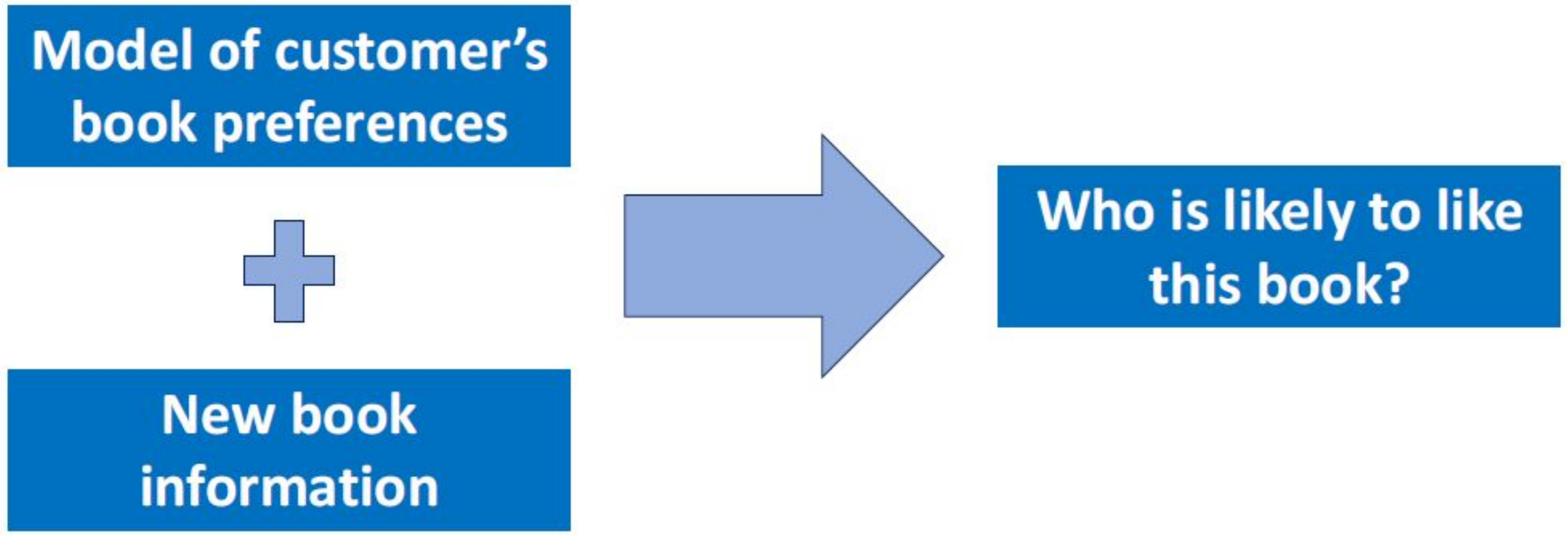

Insight
Data Product
Data
Analysis
Question
Insight

Book Recommendations



amazon.com

Find Potential Audience for a Book


Model of customer's book preferences
New book information
Who is likely to like this book?

Market a New Book

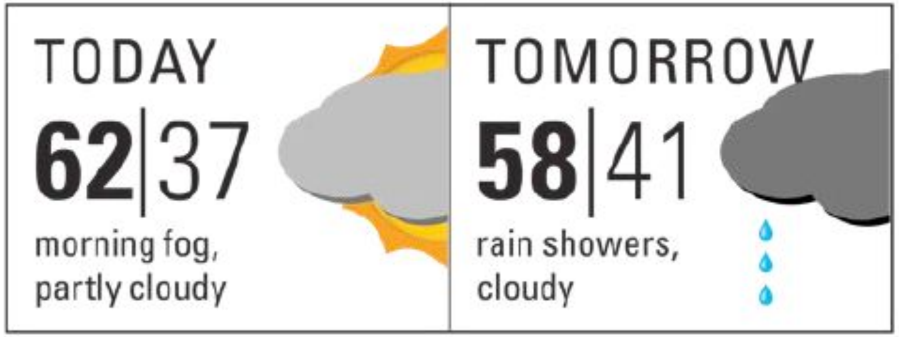
Who is likely to like this book? → Action to market the book to the right audience

Market a New Book

Who is likely to like this book? → Action to market the book to the right audience

Insight → Action

Prediction



Action

Every minute…

EMAIL

204 Million emails

facebook

200,000 photos

1.8 Million likes

YouTube

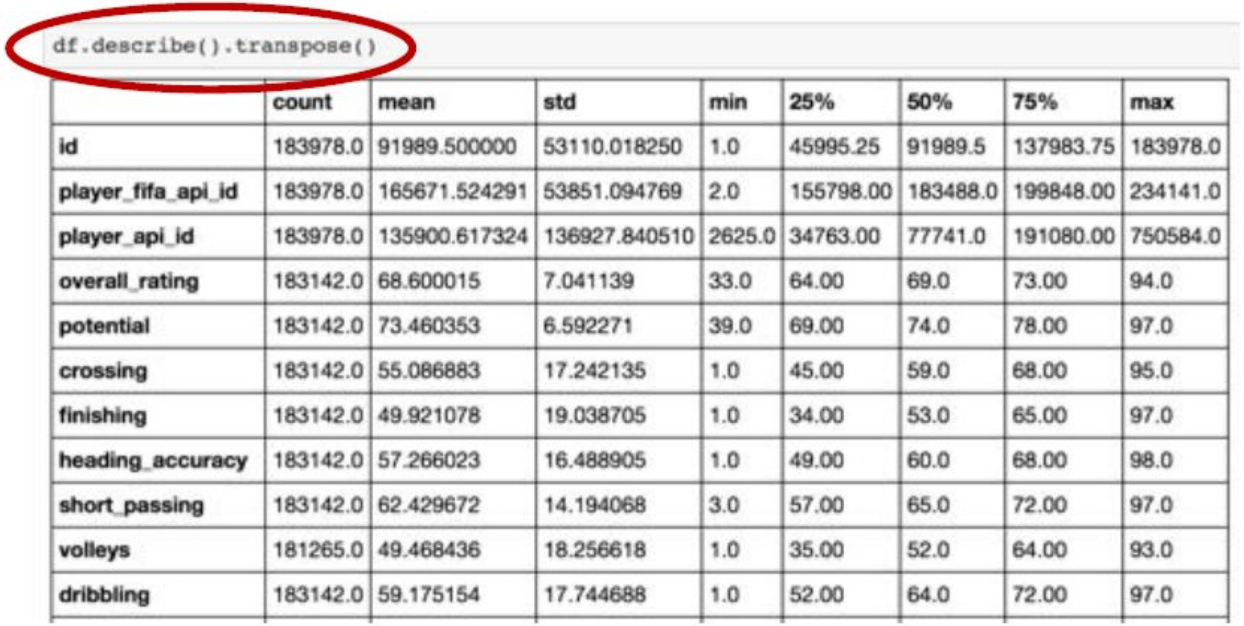
2.78 Million video views

72 hours of video uploads

Basic Steps in a Data Science Project

ACQUIRE	• Import raw dataset into your analytics platform
PREPARE	• Explore & Visualize
	• Perform Data Cleaning
ANALYZE	• Feature Selection
	• Model Selection
	• Analyze the results
REPORT	• Present your findings
ACT	• Use them

Data Preparation: Explore using Statistics

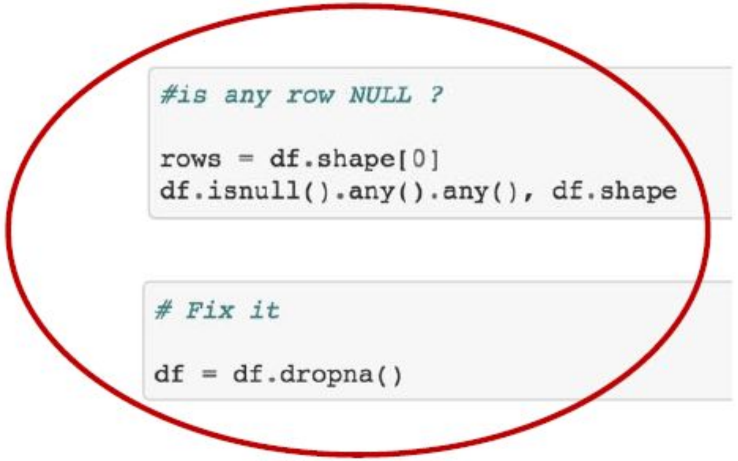


```
df.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
id	183978.0	91989.500000	53110.018250	1.0	45995.25	91989.5	137983.75	183978.0
player_fifa_api_id	183978.0	165671.524291	53851.094769	2.0	155798.00	183488.0	199848.00	234141.0
player_api_id	183978.0	135900.617324	136927.840510	2625.0	34763.00	77741.0	191080.00	750584.0
overall_rating	183142.0	68.600015	7.041139	33.0	64.00	69.0	73.00	94.0
potential	183142.0	73.460353	6.592271	39.0	69.00	74.0	78.00	97.0
crossing	183142.0	55.086883	17.242135	1.0	45.00	59.0	68.00	95.0
finishing	183142.0	49.921078	19.038705	1.0	34.00	53.0	65.00	97.0
heading_accuracy	183142.0	57.266023	16.488905	1.0	49.00	60.0	68.00	98.0
short_passing	183142.0	62.429672	14.194068	3.0	57.00	65.0	72.00	97.0
volleys	181265.0	49.468436	18.256618	1.0	35.00	52.0	64.00	93.0
dribbling	183142.0	59.175154	17.744688	1.0	52.00	64.0	72.00	97.0

Data Cleaning

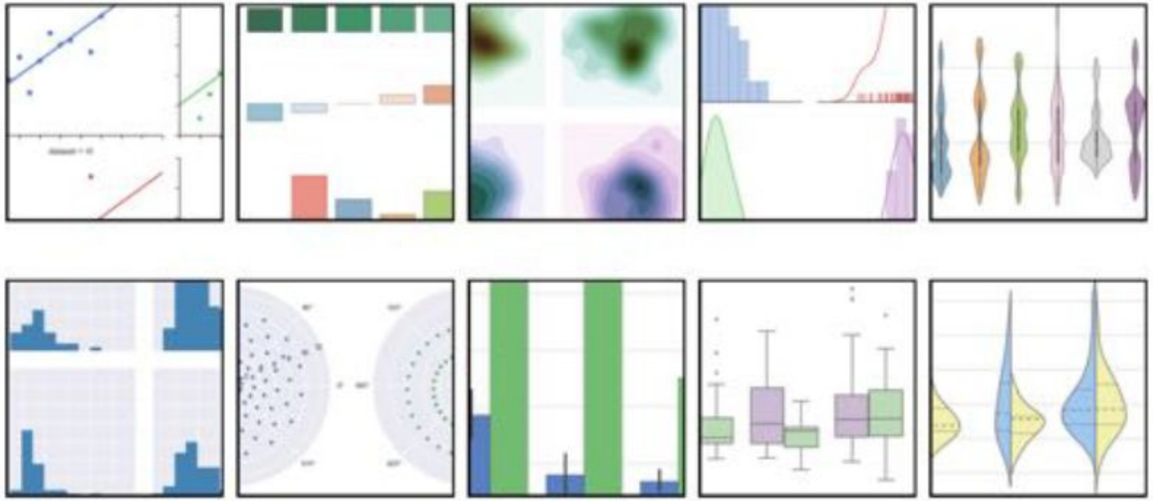
- Why do we need to clean data?
  - Missing entries
  - Garbage values
  - NULLs
- How do we clean data?
  - Remove the entries
  - Impute these entries with a counterpart
    - Ex. Average values of the column
    - Ex. Assign 0, -1, etc



```
#is any row NULL ?

rows = df.shape[0]
df.isnull().any().any(), df.shape
```

```
# Fix it

df = df.dropna()
```

Data Visualization

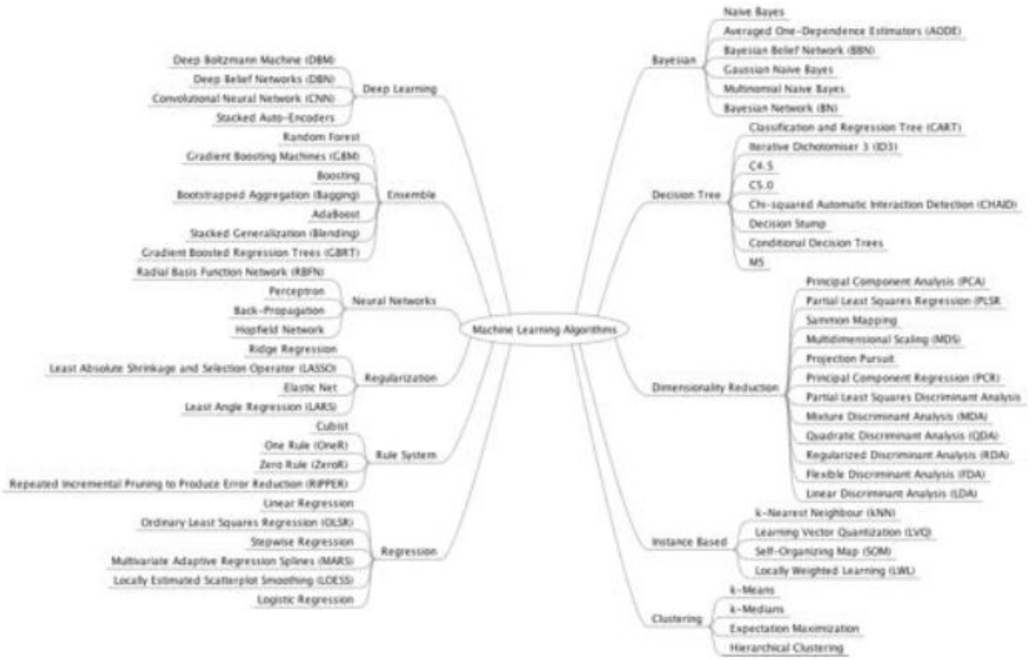


Convey more in less space and time

Use Graphs when possible

Analysis and Modeling

- Supervised Learning
- Unsupervised Learning
- Semi supervised Learning


Machine Learning Algorithms
Deep Learning
Deep Boltzmann Machine (DBM)
Deep Belief Networks (DBN)
Convolutional Neural Network (CNN)
Stacked Auto-Encoders
Ensemble
Random Forest
Gradient Boosting Machines (GBM)
Boosting
Bootstrapped Aggregation (Bagging)
AdaBoost
Stacked Generalization (Blending)
Gradient Boosted Regression Trees (GBRT)
Neural Networks
Radial Basis Function Network (RBFN)
Perceptron
Back-Propagation
Hopfield Network
Regularization
Ridge Regression
Least Absolute Shrinkage and Selection Operator (LASSO)
Elastic Net
Least Angle Regression (LARS)
Rule System
Cubist
One Rule (OneR)
Zero Rule (ZeroR)
Repeated Incremental Pruning to Produce Error Reduction (RIPPER)
Regression
Linear Regression
Ordinary Least Squares Regression (OLSR)
Stepwise Regression
Multivariate Adaptive Regression Splines (MARS)
Locally Estimated Scatterplot Smoothing (LOESS)
Logistic Regression
Bayesian
Naive Bayes
Averaged One-Dependence Estimators (AODE)
Bayesian Belief Network (BBN)
Gaussian Naive Bayes
Multinomial Naive Bayes
Bayesian Network (BN)
Decision Tree
Classification and Regression Tree (CART)
Iterative Dichotomiser 3 (ID3)
C4.5
C5.0
Chi-squared Automatic Interaction Detection (CHAID)
Decision Stump
Conditional Decision Trees
M5
Dimensionality Reduction
Principal Component Analysis (PCA)
Partial Least Squares Regression (PLSR
Sammon Mapping
Multidimensional Scaling (MDS)
Projection Pursuit
Principal Component Regression (PCR)
Partial Least Squares Discriminant Analysis
Mixture Discriminant Analysis (MDA)
Quadratic Discriminant Analysis (QDA)
Regularized Discriminant Analysis (RDA)
Flexible Discriminant Analysis (FDA)
Linear Discriminant Analysis (LDA)
Instance Based
k-Nearest Neighbour (kNN)
Learning Vector Quantization (LVQ)
Self-Organizing Map (SOM)
Locally Weighted Learning (LWL)
Clustering
k-Means
k-Medians
Expectation Maximization
Hierarchical Clustering