Academy Xi

Data Analytics Elevate

# DATA ANALYTICS

DAE2-E

**Week 1 : Topic 1: Python Essentials - Data Types II**
**Session 2: 1 September 2021**

## Welcome Everyone!

- You can download these slides and live sessions from the shared **Student folder** on Google Drive:
  - https://drive.google.com/drive/folders/15NLyoIDW-EObb5SPow7GLQsqQo0Kx3oL?usp=sharing
- The link to join each Zoom meeting on Monday and Wednesday evening is:
  - https://academyxi.zoom.us/j/2042517222
  - Every live session will be recorded on Zoom and uploaded to the Student Folder under 'Live Sessions':
- Pre-reading: each week please complete the readings and labs before class those marked with ☆
- 80% Student Attendance
- Housekeeping rules:
  - Please engage in the live session and turn on your camera so that we can see your friendly face ☺
  - Turn your mobile phone to silent
  - Please turn your mic on mute and use chat unless you are speaking

# AGENDA

1. Recap of Session 1 – Data Types Part I
2. Data Types Part 2

a) **Conditionals**

b) **Built-in Python Operators, Functions and Methods**

3. Labs
4. Project 1
5. Wrap-Up

# Learning Objectives

- Use built-in Python functions and methods
- Use comparison operators to compare objects
- Use logical operators to incorporate multiple conditions
- Use identity operators to incorporate multiple conditions
- Use identity operators to confirm the identity of an object
- Use Python conditional statements

# Our Expectations

- **You're ready to take charge of your learning experience**
- **You are curious about Python!**
- **You will dedicate between 8-10 hours each week:**
  - Labs
  - Readings
  - Projects
  - Other resources: videos, blogs, meetups, online courses, Google search, Stack Overflow,books

# The Big Picture – Career Opportunities

python.org/jobs/

Multi Media LLC — Remote, CA, United States
Looking for: Back end, Django
Posted: 20 August 2021 — Category: Developer / Engineer

NEW Senior Software Engineer (REMOTE OK) — San Francisco, CA, USA
Angaza
Looking for: Back end, Cloud, Database, Web, Python, Flask, SQLAlchemy, Celery, PostgreSQL, AWS
Posted: 19 August 2021 — Category: Developer / Engineer

NEW Senior Data Scientist (ALS) - Houston, TX, Claremore, OK or Calgary — Houston, Texas, United States
Baker Hughes
Looking for: Database, Management, Systems, Testing
Posted: 19 August 2021 — Category: Researcher / Scientist

NEW Django Developer — Birmingham, United Kingdom
The Developer Society
Looking for: Back end, Cloud, Database, Testing, Web
Posted: 19 August 2021 — Category: Developer / Engineer

NEW Senior Software Engineer — London, London, United Kingdom
nPlan
Looking for: Back end, Big Data, Cloud, Machine Learning
Posted: 18 August 2021 — Category: Developer / Engineer

NEW Lead Backend Engineer - 100% REMOTE — Slovenia, Europe
Remotesome
Looking for: Back end, Python, Node, Go
Posted: 18 August 2021 — Category: Developer / Engineer

Source: python.org/jobs

- Python programming is a sought after skill!
- Opens doors to opportunities from data science, research, AI, natural language processing and data journalism

## Highest paying cities in Australia for Python Developers

| | | | |
|---|---|---|---|
| 1 | Canberra ACT, 9 salaries reported | $206,705 per year | |
| 2 | Sydney NSW, 78 salaries reported | $153,763 per year | |
| 3 | Sydney Central Business District ..., 38 salaries reported | $139,841 per year | |
| 4 | Clovelly NSW, 5 salaries reported | $125,908 per year | |
| 5 | Brisbane QLD, 10 salaries reported | $125,257 per year | |
| 6 | Melbourne VIC, 29 salaries reported | $122,994 per year | |
| 7 | Bondi Beach NSW, 7 salaries reported | $119,585 per year | |
| 8 | Perth WA, 6 salaries reported | $115,567 per year | |

Source: Indeed

Academy Xi

# In the Previous Session:

## Variables and Data Types

1. **Numbers:**

   - Integer (int)

   - Float

2. **Booleans** (TRUE, FALSE)
3. **Sequence**

   - String (str)

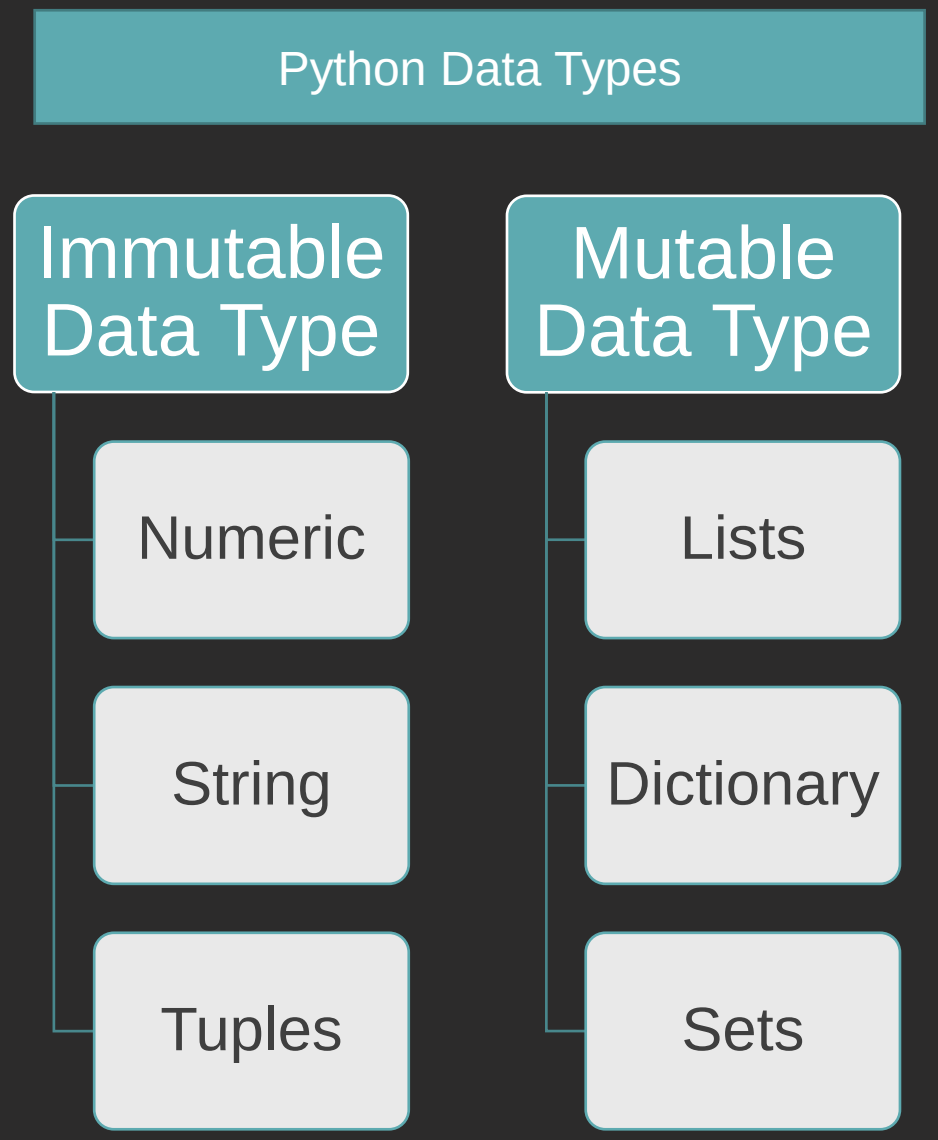
   - Lists

   - Tuple

4. **Set**
5. **Dictionary**

## Re-Cap

Data Types Part 1: Numeric vs Sequence



# Session 1 Recap

- Recording and Slides located in the Student Folder on Google Drive:

  https://drive.google.com/drive/folders/15NLyoIDW-EObb5SPow7GLQsqQo0Kx3oL?usp=sharing

# What is OO?

**Object-oriented (OO):** In Python we focus on objects that contain data (attributes)

- Everything is an **object** in Python programming
- Variables can store different data types
- Variables are objects of these classes

# Variable Assignment

- Python variable names are **case-sensitive**
- Variable names can only use letters, underscore and numbers
- Variable name start with a letter or an underscore
- Variable names cannot use Python's **reserved words** (https://realpython.com/lessons/reserved-keywords/)

Left hand-side assigns the variable to the Right hand-side. In Python = is variable assignment

In [1]:
```
# Assign a variable name
```

In [2]:
```
my_height = 183
```

In [3]:
```
print(my_height)      # print() allows you to view the output of the input variable and also useful for debugging
```

```
183
```

Check the data type:

type(my_height)  # integer

# Calculator: Arithmetic Operators

| Operator | Meaning | Example |
|---|---|---|
| + | Add two operands or unary plus | x + y+ 2 |
| - | Subtract right operand from the left or unary minus | x - y- 2 |
| * | Multiply two operands | x * y |
| / | Divide left operand by the right one (always results into float) | x / y |
| % | Modulus - remainder of the division of left operand by the right | x % y (remainder of x/y) |
| // | Floor division - division that results into whole number adjusted to the left in the number line | x // y |
| ** | Exponent - left operand raised to the power of right | x**y (x to the power y) |

Source: https://www.programiz.com/python-programming/operators

# Python Data Types

- float - real numbers (fractions and decimals)
- int - integer numbers
- str - string, text, characters
- bool - True, False
- list - List
- tuple - Tuple
- set - Set

# Data Type – Sequence - Lists [ ]

Updated slide

- **Ordered collection** of objects in Python
- **Mutable** or flexible
- Lists can be altered after they are created
- Lists can include **different data types** such as numbers, strings, other lists, tuples
- In lists you access individual values with consecutive integers calls **indices** or **index values**

```
# Create a list
b = [“ Academy”, “Xi”, “Data”, “Analytics”]

# Print the last element of the list
print(nums[-1])

“Analytics”

b[2] = 6                    # Replace an element in a list

‘Academy’, ‘Xi’,’6’,’Analytics’

b.append(“UX”)              # Adds an element to a list

‘Academy’, ‘Xi’,’6’,’Analytics’, ‘UX’

b.remove(“UX”)              # Removes an element of a list
```

# Accessing elements within a List [ ]

Updated slide

```
# Create a list
z = [“ Academy”, “Xi”, “Data”, “Analytics”, “cloud”, “UX”, “CX”, “Digital”, “Customer Journey”]
z
```

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | | | | | | | -2 | -1 |

Count from left to right

Count from right to left

```
# Locate the position of the third element using indexing (where index starts from 0)
List[3]
```

“Analytics”

```
# Locate the second last element within the list with negative indexing
List[-2]
```

“Digital”

# Accessing elements in a List [ ]

Updated slide

```
# Create a list
z = ['Academy', 'Xi', 'Data', 'Analytics', 'cloud', 'UX', 'CX', 'Digital', 'Customer Journey']
z
```

0 1 2 3 4 5 6 7 8

← Count from left to right →

**# Accessing multiple elements within a list**

z[3:5] (start from position 3)

['Analytics', 'cloud']

**# Accessing multiple elements within a list**

z[1:4]

['Xi', 'Data', 'Analytics']

[start : end]

include exclude

# Accessing elements in a List [ ]

Updated slide

```
# Create a list
z = ['Academy', 'Xi', 'Data', 'Analytics', 'cloud', 'UX', 'CX', 'Digital', 'Customer Journey']
z
```

0 1 2 3 4 5 6 7 8

← Count from left to right →

```
# Everything up to but not including index 4
z[ :4]
```

'Academy', 'Xi', 'Data', 'Analytics'

```
# Select items at index 1 and 2
z[1:3]
```

'Xi', 'Data'

```
# Select items after index 4
z[ 4:]
```

'cloud', 'UX', 'CX', 'Digital', 'Customer Journey'

[start : end]

include exclude

# Sequence Data Types

| Sequence Data Types | | |
|---|---|---|
| String (str) | A string (or text) is a collection of one or more characters put in single quote or double quote | “Welcome to session 2” |
| List (list) | Ordered collection of objects in Python | List = [“ Academy”, “Xi”, “Data”, “Analytics”] |
| Tuple (tuple) | Similar to a list except a tuple cannot be modified or changed after it’s created | List = (“ Academy”, “Xi”, “Data”, “Analytics”) |

# Data Type: Booleans

- Boolean have two possible values TRUE or FALSE are called logical
- Bool
- TRUE or FALSE must be uppercase
- Boolean arithmetic operators:
  - and
  - or
  - not
  - ==
  - !=

```
breakfast_list = [ “milk”, “ scrambled egg”, “toast”, “cereal”, “coffee”, “yoghurt”, “french toast”, “toast”]
dinner_list = [“steak”, “broccoli”, “pasta”, “salmon”]

# find out if milk is in the breakfast or dinner list
print(“milk” in breakfast_list or dinner_list)

True
```

# Logical Operators: AND, OR, NOT

| Operator | Meaning | Example |
|---|---|---|
| and | True if both the operands are true | x and y |
| or | True if either of the operands is true | x or y |
| not | True if operand is false (complements the operand) | not x |

Source: https://www.programiz.com/python-programming/operators

# Data Type - Dictionary { }

- Dictionaries combine **keys** with **values** in paired with a unique identifier
- Sequence data is stored in **unordered** manner
- A dictionary itself is mutable, but each of its individual keys are **immutable**
- Like in a dictionary, you can search the keys to obtain their corresponding value
- In dictionaries you access individual values using integers, strings or other python objects called **keys**
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([])
- Remove an element with del() or pop()

Example: covid exposure areas

```
epo_dict = {‘canterbury-bankstown": 25.0, “liverpool": 30.0, “bondi":55.0}
print(type(epo_dict ))
print(epo_dict)

bondi_expo = epo_dict[“bondi”]
```

# Data Type - Set

- An unorderedcollection of string or integer with no specific order or index, like a list
- Automatically sorts values in alphabetical order
- Has **no duplicate** elements
- Is iterable and mutable
- There is no indexing because elements are unordered
- Built-in set function
- Used in a for loop for looping over set items

```
# Obtain unique values from breakfast list
breakfast_list = [ “milk”, “ scrambled egg”, “toast”, “cereal”, “coffee”, “yoghurt”, “french toast”, “toast”]
print(set(breakfast_list))

[ “milk”, “ scrambled egg”, “toast”, “cereal”, “coffee”, “yoghurt”, “french toast”]
```

# This Session:

## Python Data Types Part II

1. **Conditionals and Control Flow**

2. **Built-in operators and functions**

Academy Xi

# 1. Comparison Operators

| Operator | Meaning | Example |
|---|---|---|
| > | Greater than - True if left operand is greater than the right | x > y |
| < | Less than - True if left operand is less than the right | x < y |
| == | Equal to - True if both operands are equal | x == y |
| != | Not equal to - True if operands are not equal | x != y |
| >= | Greater than or equal to - True if left operand is greater than or equal to the right | x >= y |
| <= | Less than or equal to - True if left operand is less than or equal to the right | x <= y |

Source: https://www.programiz.com/python-programming/operators

# 2. Boolean Operators

- Boolean arithmetic operators:
  - and
  - or
  - not

# 3. Conditional Statements

- Conditional Statements:
  - if
  - else
  - elif

# Control Flow

Control Flow:

- Provides business logic in programs
- Can assist with **decision-making** such as using If Statements such as conditionals

- Python control flow elements include:

  - if-else
  - if-elif-else
  - range function
  - for loops (looping over collections)
  - while loops

# Important: Assignment Operators

| Operator | Example | Equivalent to |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| *= | x *= 5 | x = x * 5 |
| /= | x /= 5 | x = x / 5 |
| %= | x %= 5 | x = x % 5 |
| //= | x //= 5 | x = x // 5 |
| **= | x **= 5 | x = x ** 5 |
| &= | x &= 5 | x = x & 5 |
| \|= | x \|= 5 | x = x \| 5 |
| ^= | x ^= 5 | x = x ^ 5 |
| >>= | x >>= 5 | x = x >> 5 |
| <<= | x <<= 5 | x = x << 5 |

Source: https://www.programiz.com/python-programming/operators

# Conditionals – If statements

**If statements:**

- "If this then do that, else do something else"

Example: Automatically top up your Opal card balance

```
opal_balance = 10

if opal_balance < 25 :
   opal_balance += 30

40
```

Check the condition

If condition is TRUE, then top up the opal balance to $40 (i.e. 10 +30)

If condition is FALSE the opal balance is $50 there is no top up

Increment the variable with +=

# Conditionals – If statements

**If statements:**

- "If this then do that, else do something else"

Example: Using Python to illustrate an if statement

```
i= 10
if (i > 20):                                  # check if the variable i is greater than 20
    print("10 is less than 20")
Print(" I am Not in if")                      # this print statement does not  belong to the if block of code
```

indentation

```
I am Not in if
```

# Conditionals – If-else statements

**If-else statements:**

- "If this then do that, else do something else"

Example: Automatically top up your Opal card balance

```
opal_balance = 50

if opal_balance < 25 :
   opal_balance += 30
else:
  message = " do not top up opal card"

do not top up opal card
```

Check the condition

If condition is FALSE, then there is no Opal card top up and the balance remains $50

# Conditionals – If-else statements

In Excel: if(A1 < 15, 1,0)

In Excel: if(A1 < 15, True, False)

**If-else statements:**

- "If this then do that, else do something else"

Example: Using Python to illustrate an if-else statement

```
i= 20
if (i < 15):                                    # check if the variable i is less than 15
    print("i is smaller than 15")
    print(" I am in the if block")
else:                                           # do the following if  is is NOT less than 15
   print(" i is greater than 15")
   print("i am in the else block")
print(" i'm not in if  and not in the else block")
```

```
i is greater than 15
i am in the else block
i'm not in if  and not in the else block
```

# Conditionals – if-elif-else statements

**If-elif-else statements:**

- elif is another if statement

Example: Automatically top up your Opal card balance

```
opal_balance = 50
bank_balance = 80

if opal_balance > bank_balance:
   print("opal balance is larger than bank balance")
elif opal_balance == bank_balance:
   print("opal balance is equal to  bank balance")
else:
  print("opal balance is smaller than bank balance")
```

opal balance is smaller than bank balance

# Conditionals – if-elif-else statements

**If-elif-else statements:**

- elif:  is another if statement
- else:  do something if the other conditions are not met or is FALSE

Example:

```
i= 20
if (i == 10):                          # check if the variable is equal to 10
   print("i is 10")
elif (i == 15):                        # if i is not equal to 10, check if i equals 15
   print("i is15")
elif (i == 20):                        # if i is not equal to neither  10 or 15, check if it equals 20
   print("i is 20")
else:
   print("i is not present")
```

i is 20

# Lab

# Python Built-In Methods

## List Method `.remove()`

The `.remove()` method in Python is used to remove an element from a list by passing in the value of the element to be removed as an argument. In the case where two or more elements in the list have the same value, the first occurrence of the element is removed.

```
# Create a list
shopping_line = ["Cole", "Kip", "Chris", "Sylvana",
"Chris"]

# Removes the first occurance of "Chris"
shopping_line.remove("Chris")
print(shopping_line)

# Output
# ["Cole", "Kip", "Sylvana", "Chris"]
```

## List Method `.pop()`

The `.pop()` method allows us to remove an element from a list while also returning it. It accepts one optional input which is the index of the element to remove. If no index is provided, then the last element in the list will be removed and returned.

```
cs_topics = ["Python", "Data Structures", "Balloon
Making", "Algorithms", "Clowns 101"]

# Pop the last element
removed_element = cs_topics.pop()

print(cs_topics)
print(removed_element)

# Output:
# ['Python', 'Data Structures', 'Balloon Making',
'Algorithms']
# 'Clowns 101'
```

Check Python's directory for all built in methods for a data type:

dir()

e.g.

- dir(str) for string
- dir(dict) for dictionary
- dir(int) for integer
- dir(float) for float

# Changing Data with Python Built In String Methods

- .upper()

```
txt = "Hello my friends"

x = txt.upper()

print(x)
```

```
HELLO MY FRIENDS
```

- .title()

Make the first letter in each word upper case:

```
txt = "Welcome to my world"

x = txt.title()

print(x)
```

```
Welcome To My World
```

- .capitalize()

```
sentence = "woW WE LOVE cOdInG and strINGS!".capitalize()
sentence
```

```
'Wow we love coding and strings!'
```

https://www.w3schools.com/python/python_ref_string.asp

Academy Xi

# Python Identity Operators

| Operator | Description | Example |
|---|---|---|
| is | Returns true if both variables are the same object | x is y |
| is not | Returns true if both variables are not the same object | x is not y |

# Python Logical Operators (i.e. Boolean Operators)

| Operator | Description | Example |
|---|---|---|
| and | Returns True if both statements are true | x < 5 and  x < 10 |
| or | Returns True if one of the statements is true | x < 5 or x < 4 |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) |

https://www.w3schools.com/python/gloss_python_logical_operators.asp

# Python Comparison Operators

- Tests the equality of two elements

| Operator | Name | Example |
|---|---|---|
| == | Equal | x == y |
| != | Not equal | x != y |
| > | Greater than | x > y |
| < | Less than | x < y |
| >= | Greater than or equal to | x >= y |
| <= | Less than or equal to | x <= y |

# Lab

# Project 1

# Optional Sessions for this week:

- Recap: Data Types Part I: Thursday 6-8pm ✓
- Recap: Data Types Part II: Friday7-9pm (a minimum of 3 people by Thursday)

Sessions will be recorded so you may watch later ☺

# Re-Cap

Session 2: Data Types Part 2

- Control Flow: Conditionals
- Built-In Python Operators, Functions and Methods

# Resources

- https://www.pythoncentral.io/top-5-free-python-resources/
- https://www.freecodecamp.org/news/python-if-else-statement-conditional-statements-explained/
- Built-in methods:https://www.codecademy.com/learn/learn-python-3/modules/learn-python3-lists/cheatsheet
- Comparison Operators: https://www.codecademy.com/learn/learn-python-3/modules/learn-python3-control-flow/cheatsheet

# Homework – Week 1 Labs

- Introduction to Variables: Variable Assignment – Lab
- Introduction to Variables: Strings - Lab
- Working with Lists – Lab
- Working with Dictionaries – Lab
- Built-in Python Operators, Functions and Methods – Lab
- Control Flow: Conditionals – Lab

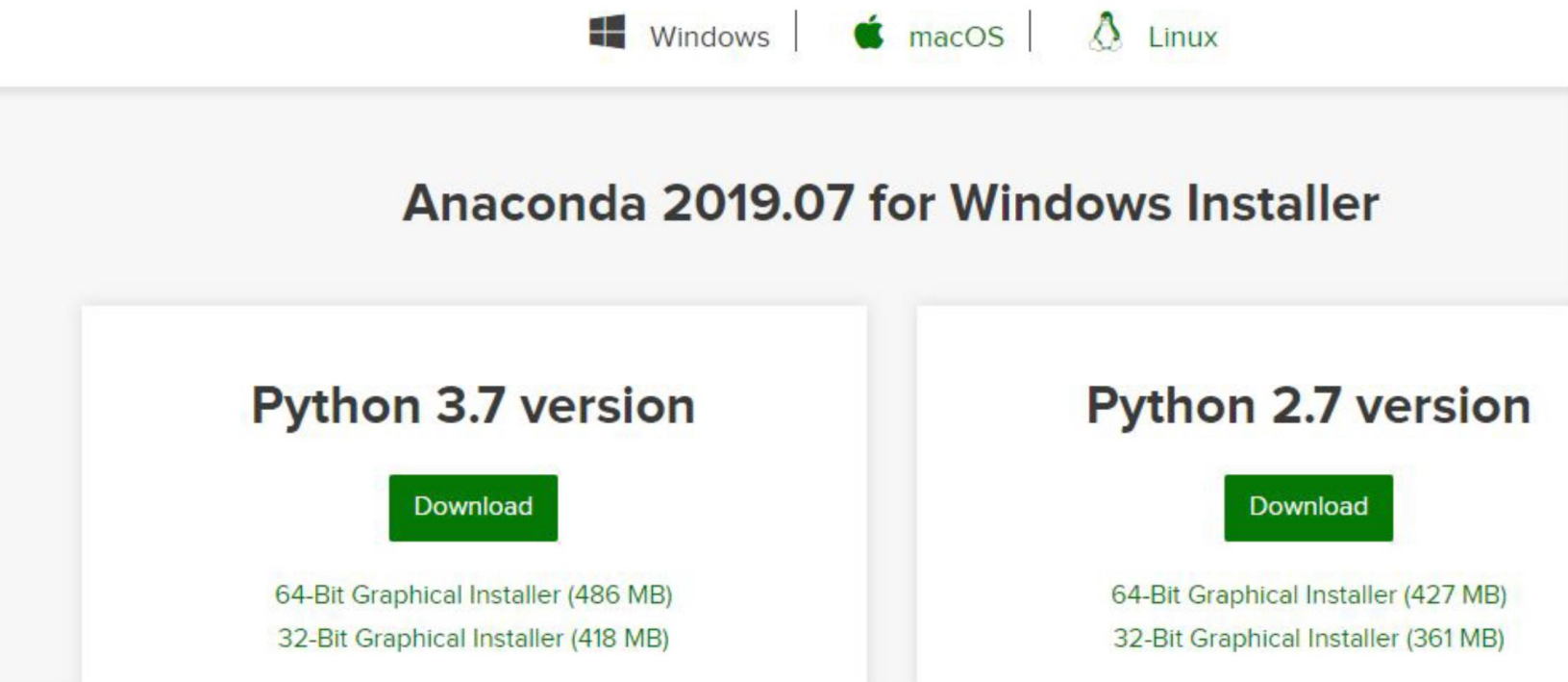
Pre-Reading Week 2:

- Loops
- Functions
- Labs
- Optional: Install: Python 3.8 and above onto your laptop from Anaconda.(We will go through this in Week 2)

# Install latest version of Python 3

## Download Python 3.7 version

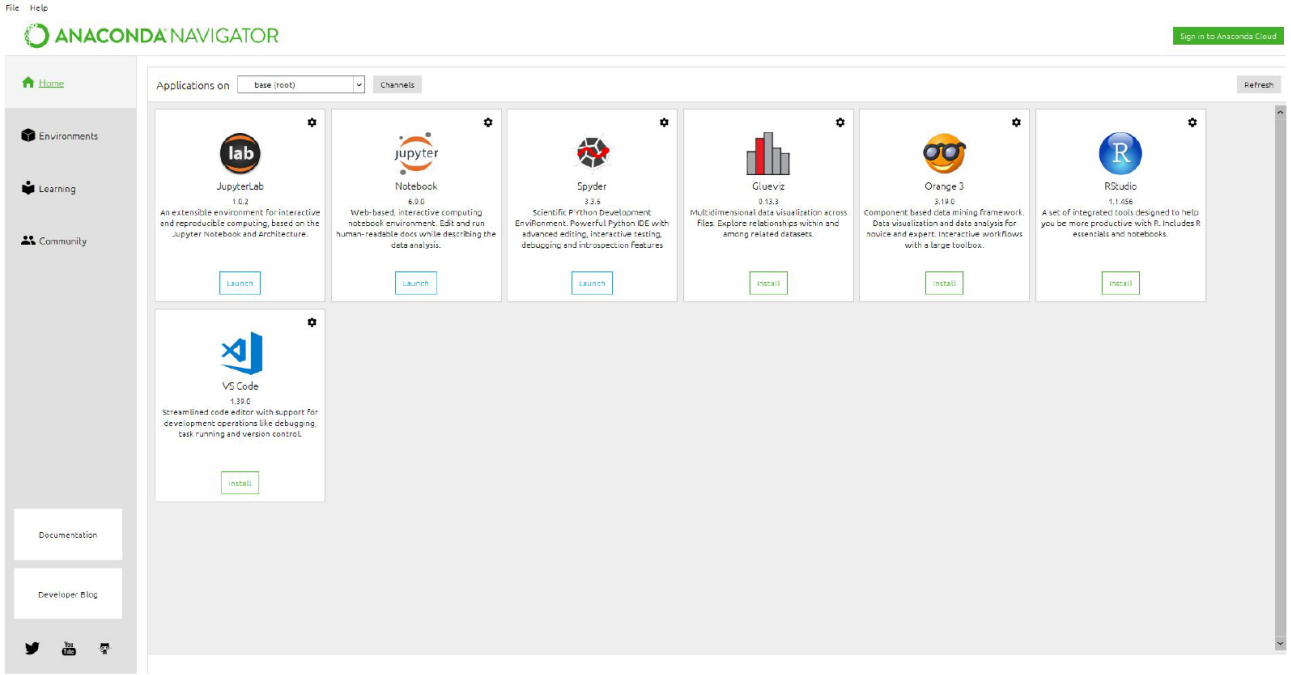
https://www.anaconda.com/distribution/

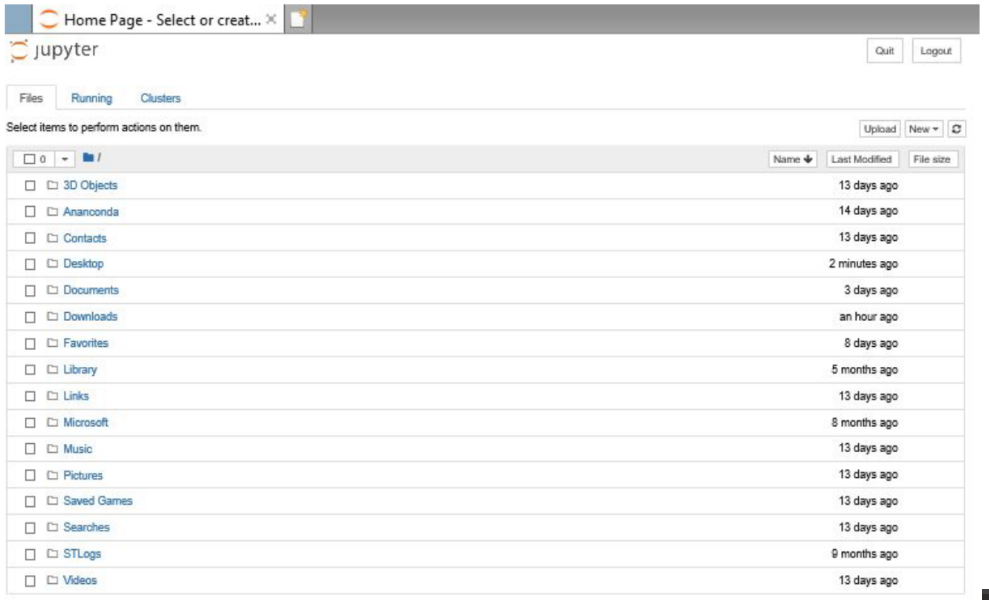


# Launch Python

## Launch Python 3.7 from Windows

1. Create a **new folder** e.g. Python Training from your Desktop.
2. From your **Start Menu**, type **Anaconda Navigator (Anaconda)**.
3. Click **Ok** on the pop up message.
4. Click **Launch** from **Jupyter Notebook.**



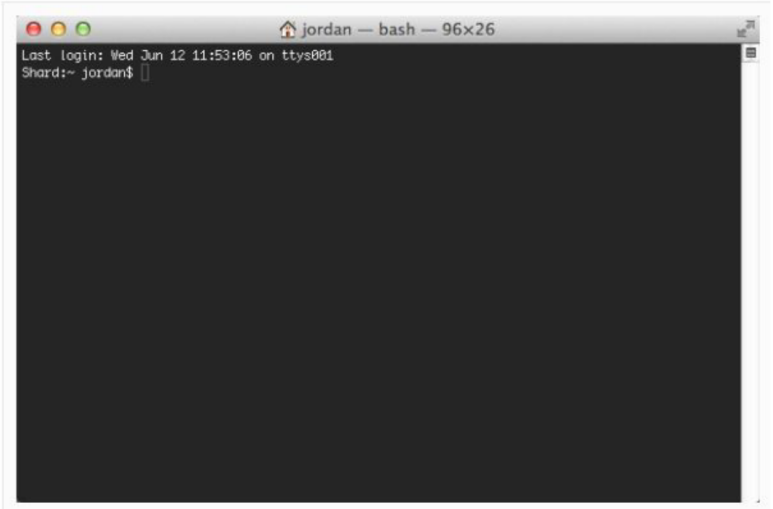



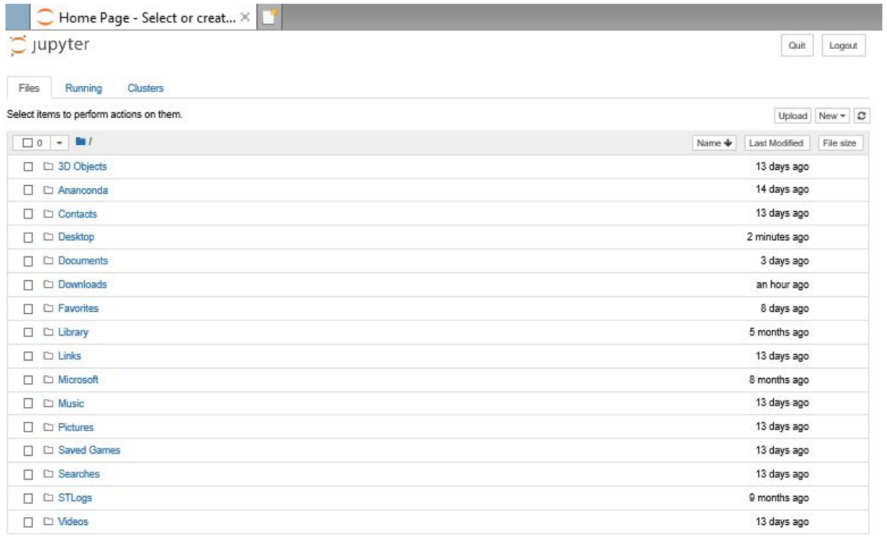
# Launch Python

## Launch Python 3.7 from Mac

1. Create a **new folder** e.g. Python Training from your Desktop.
2. In **Terminal**, type **Jupyter Notebook.**

### Entering Terminal



## Session 3: Python Functions and Loops

- Re-cap: Python Essentials Session 2
- Control Flow: Writing Functions
  - With and without arguments
- Control Flow: Loops
  - For Loops
  - While Loops
  - Nested Loops
  - Looping over collections

- Mentoring: You may book mentoring sessions in my calendar from Week 1:

  https://calendly.com/da_mentor2

# Thank You

# Stay Connected

**Join Slack channel:** da-e-2.slack.com

# general
# questions
# random