



WELCOME TO PYTHON PROGRAMMING!

A PYTHON BOOTCAMP

Center for Continuing Education AND ViSER LLC

April 27 - 30, 2020



(C) ViSER LLC

BOOTCAMP AGENDA



	BACKGROUND AND INSTALLATION
APRIL 27	INTRO TO PYTHON BASICS Data types, operators, type conversion etc STRING PARSING AND FORMATTING
APRIL 28	DEFINING FUNCTIONS USING LOOPS AND CONDITIONS: <ul style="list-style-type: none"> • If-then-else (CONDITIONS) • for AND while (LOOPS)
APRIL 29	DATA STRUCTURES Lists and its functions, Dictionary, Numpy array
APRIL 30	DRAWING GRAPHS <ul style="list-style-type: none"> • Using Matplotlib package: Scatter plot, Line chart, Bar chart, Histogram • Intro to other plotting packages

(C) ViSER LLC



INTRODUCTION



- Created by *Guido van Rossum* and first released in 1991
- Meant to be an easily readable
 - often uses English keywords
- Top programming language (IEEE – 2018 and 2019)
 - simple to use
 - vast application in Data Analysis and other fields (Machine Learning)
 - salary and job openings in 2019 (codeplatoon.org)
- Interpreted language
 - easy to debug
- Extensive support libraries
 - many programs are already embedded in libraries and reduce the length of the code
- Open source language
 - freely available for the programmers to download and distribute for commercial use
- Cross-platform
 - supports all the major platforms such as windows, Linux, Macintosh

(C) ViSER LLC



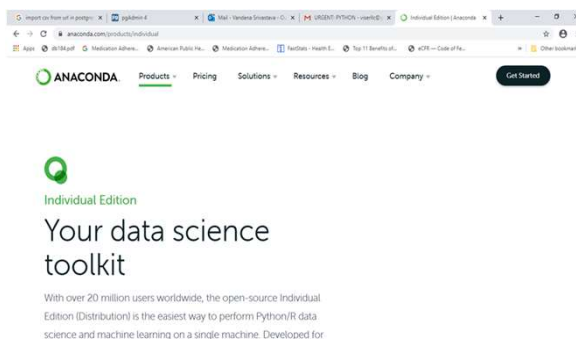
INSTALLATION (ANY ONE OF THE TWO)



- Integrated development environment (IDE):
 - environment for writing, editing, debugging and running Python programs
 - IDLE, PyCharm, Eclipse with Pydev, Sublime Text 3 etc

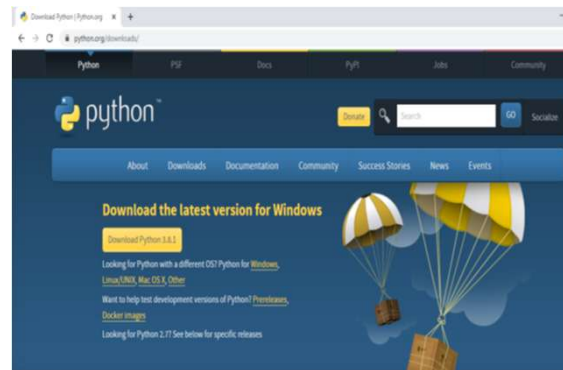
Anaconda Distribution

(<https://www.anaconda.com/products/individual>)



(C) ViSER LLC

(<https://www.python.org/downloads/>) for windows , Linux/UNIX, Mac OS X,



- Type "jupyter notebook" in the **windows search bar** at the bottom of the screen to launch Jupyter



PYTHON BASICS



- Hash(#) is used to write a comment in Python
 - Example -- #This is my first program
- Python ignores everything after the hash mark and up to the end of the line
- Comments can be inserted anywhere in the code, even inline with other code
 - `print("This will be printed.") # This won't run`
- For multiline comments, triple quotes are used


```
""" This comment span
many lines and it can be done
using triple quotes
"""
```
- One of the distinctive features of Python is its use of indentation to highlight the blocks of code

(C) VISOER LLC



PYTHON BASICS -- DATA TYPES



Integer(int), Decimal (float), Text or string (str), Boolean(bool)

```
>>> type(1)
```

```
<class 'int'>
```

```
>>> type(1.0)
```

```
<class 'float'>
```

```
>>> type("1")
```

```
<class 'str'>
```

```
>>> type(True)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#4>", line 1, in
<module>
```

```
    type(True)
```

```
NameError: name 'True' is not
defined
```

```
>>> type("True")
```

```
<class 'str'>
```

```
>>> type(True)
```

```
<class 'bool'>
```

```
>>> float(10)
```

```
10.0
```

```
>>> float(10.5)
```

```
10.5
```

```
>>> int(10)
```

```
10
```

```
>>> int(-10)
```

```
-10
```

```
>>> int(10.5)
```

```
10
```

```
>>> str("10")
```

```
'10'
```

```
>>> str(10)
```

```
'10'
```

```
>>> str(-10.5)
```

```
'-10.5'
```

```
>>> bool(0)
```

```
False
```

```
>>> bool(-5)
```

```
True
```

```
>>> bool(5)
```

```
True
```

```
>>> a="python"
```

```
>>> bool(a)
```

```
True
```

The empty string "" returns as False. All other strings convert to True.

```
>>> name = "python"
```

```
>>> bool(name)
```

```
True
```

```
>>> text=""
```

```
>>> bool(text)
```

```
False
```



PYTHON BASICS -- *VARIABLES AND OPERATORS*



variable_name = value

- must start with a letter
- **can only contain** letters, numbers and the underscore character _
- **can not contain** spaces or punctuation
- **not enclosed** in quotes or brackets

```
>>> a = 5
>>> print(a)
5
>>> print("The value of a is:", a)
The value of a is: 5
```

(C) VISER LLC

ARITHMETIC OPERATOR	DESCRIPTION	SYNTAX
+, -	Addition / Subtraction	x + y, x - y
*	Multiplication	x * y
/	Division (float)	x / y
//	Division (floor)	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
RELATIONAL OPERATOR	DESCRIPTION	SYNTAX
>	Greater than	x > y
<	Less than	x < y
==	Equal to	x == y
!=	Not equal to	x != y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y
LOGICAL OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x



PYTHON BASICS -- *DATA TYPE CONVERSION*



▪ Implicit type conversion

- automatically converts one data type to another data type
- doesn't need any user involvement
- Example 1: conversion of lower datatype (integer) to higher data type (float) to avoid data loss

```
>>> num_int = 123
>>> num_flo = 1.23
>>> num_new = num_int + num_flo
>>> print(num_new)
124.23
```

- Example 2: Addition of string (higher) data type and integer (lower) datatype

```
>>> num_int = 123
>>> num_str = "456"
>>> print(num_int + num_str)
```

Traceback (most recent call last):

```
File "<pyshell#2>", line 1, in <module>
    print(num_int + num_str)
```

TypeError: unsupported operand type(s) for +: 'int' and 'str'

(C) VISER LLC

▪ Explicit type conversion

- Addition of string and integer using explicit conversion

```
>>> num_int = 123
>>> num_str = "456"
>>> num_str = int(num_str)
>>> num_sum = num_int + num_str
>>> print(num_sum)
```

579

```
>>> a = input("Enter the value of a:\n")
Enter the value of a:
5
>>> print(a)
5
>>> b = 7
>>> print(a+b) # Error
>>> a = int(input("Enter the value of a:\n"))
Enter the value of a:
5
>>> b = 7
>>> print(a+b)
12
```



PYTHON LOOPS -- "WHILE"



*while expression:
statement(s)*

when the condition becomes false, the line immediately after the loop in program is executed

```
>>> count = 0
>>> while (count < 3):
    count = count + 1
    print("Hello World")
Hello World
Hello World
Hello World
```

*while condition:
execute these statements
else:
execute these statements*

```
count = 0
while (count < 3):
    count = count + 1
    print("Hello World")
else:
    print("This block will execute")

Hello World
Hello World
Hello World
This block will execute
```

Single statement while block:

```
count = 0
while (count == 0): print("Hello World")
```

Infinite loop

suggested **not to use** this type of loops

(C) VISER LLC



PYTHON LOOPS -- "FOR"



*for index_var in sequence:
statements(s)*

```
count = 0
for i in range (1, 4):
    count = count + i
    print(count)
1
3
6
```

nested for loops in Python

```
for i in range(1, 5):
    for j in range(i):
        print(i, end=' ')
    print()

1
2 2
3 3 3
4 4 4 4
```

*for index_var in sequence:
execute these statements
else:
execute these statements*

```
for index in range(0,3):
    print (index)
else:
    print ("Inside Else Block")
0
1
2
Inside Else Block
```

break is used to exit a for loop

continue is used to skip the current block, and return to the "for" statement.

The **for** statement iterates through a collection or iterable object or generator function

The **while** statement simply loops until a condition is False.

(C) VISER LLC



PYTHON - CONDITIONAL STATEMENTS



IF conditions (single)

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

```
if condition 1 is true:
    # execute these statements
elif condition 2 is true:
    # execute these statements
elif condition 3 is true:
    # execute these statements
.
.
.
else:
    # execute these statements
```

IF conditions (multiple)

AND (example- if `a > b` and `c > a` :)

OR (example- if `a > b` or `a > c` :)

Nested If

if statements inside if statements
is called *nested* if statements.

(C) ViSER LLC



PYTHON FUNCTIONS



Creating a Function

In Python a function is defined using the **def** keyword

```
def my_function():
    print("Hello from a function")
```

```
def sum(x,y):
    add = x+y
    print(add)
```

Calling a Function

To call a function, use the function name- **my_function()**:

Parameters

- Information can be passed to functions as parameter
- can add as many parameters as needed, separating them with a comma

(C) ViSER LLC



PRACTICE EXERCISES -- LOOPS/CONDITIONS/FUNCTIONS



- Write a Python program that:
 - prints all the letters of a given text
 - prints all the numbers from 0 to 6 except 3 and 6
 - accepts a word from the user and reverse it
 - will return TRUE if the two given integer values are equal and FALSE if their sum or difference is 5
 - counts the number of even and odd numbers from a series of numbers
- Write a function that:
 - To add the numbers from 0 to n where n is a given number
 - returns the sum of multiples of 3 and 5 between 0 and limit (parameter).

(C) ViSER LLC



STRING PARSING



Parsing is the process of analyzing the string of characters

capitalize()	
count()	Returns the number of times a specified value occurs in a string
endswith()	Returns true if the string ends with the specified value, else false is returned
find()	Searches the string for a specified value and returns the position of where it was found
index()	Searches the string for a specified value and returns the position of where it was found

islower() /isupper()	Returns True if all characters in the string are lower case / upper case
isnumeric()	Returns True if all characters in the string are numeric
isupper()	Returns True if all characters in the string are upper case
split()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value

(C) ViSER LLC



PYTHON DATA STRUCTURES



List

- Can be used for any type of object, from numbers and strings to more lists
- Simple to use and they're variable length, i.e. they grow and shrink
- Can store heterogeneous data type

Dictionary

- It consists of key value pairs. The value can be accessed by unique key in the dictionary.
- Keys are unique & immutable objects.
- Syntax: dictionary = {"key name": value}

Tuple

- Usually written inside parentheses to distinguish them from lists (which use square brackets), but parentheses aren't always necessary
- Tuples are immutable, their length is fixed. To grow or shrink a tuple, a new tuple must be created.

Others

- Sets, Arrays etc



PYTHON DATA STRUCTURES



```
# Iterating over a list
print("List Iteration")
l = ["I", "love", "python"]
for i in l:
    print(i)
```

```
# Iterating over dictionary
print("\nDictionary Iteration")
d = dict()
d['xyz'] = 123
d['abc'] = 345
for i in d:
    print("%s %d" %(i, d[i]))
```

```
# Iterating over a tuple
(immutable)
print("\nTuple Iteration")
t = ("I", "love", "python")
for i in t:
    print(i)
```



LISTS VS DICTIONARY



A list is an ordered sequence of objects

items in list are accessed via their position

Dictionaries are unordered sets.

items in dictionaries are accessed via keys and not via their position

dictionary with cities located in the US and Canada and their corresponding population.

If we want to get the population of one of those cities, all we have to do is to use the name of the city as an index:

(C) VISER LLC



REFERENCES / RESOURCES



- <https://yourstory.com/mystory/interesting-facts-about-python-language>
- <https://www.geeksforgeeks.org/loops-in-python/>
- <https://www.programiz.com/python-programming/regex>
- <https://www.w3resource.com/python-exercises/>
- Online community
 - Stack overflow

(C) VISER LLC

