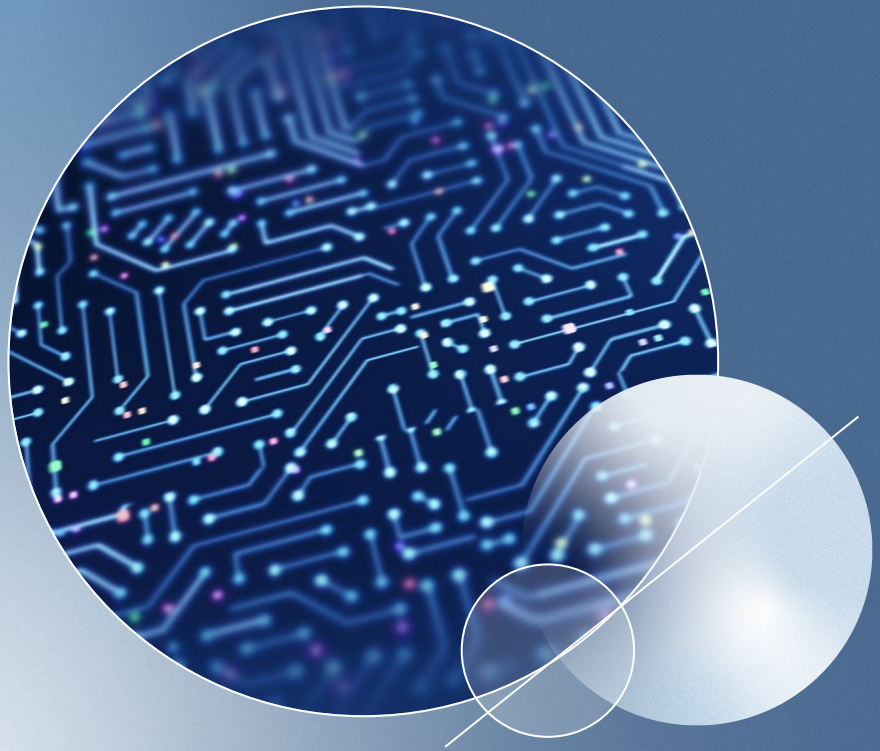
MATPLOTLIB AND NUMPY

MODULE 3

Learning Objectives

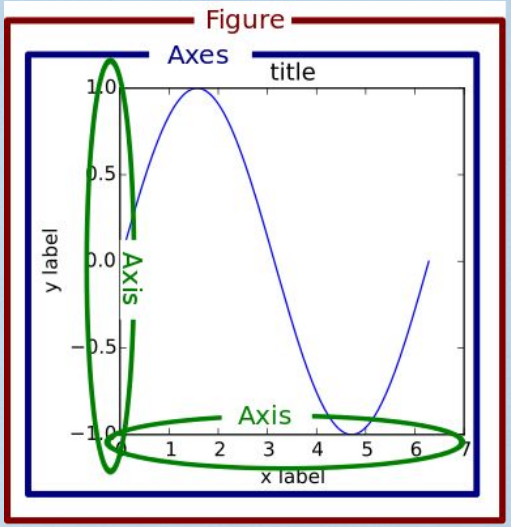


By the end of this week, students should be able to:

1. Know key concepts of matplotlib's design
2. Understand plt.subplots()
3. Visualize arrays with matplotlib

The Matplotlib Object Hierarchy

```
plt.plot([1, 2, 3])
```



- Widely used python data visualization library
- 2D and 3D plots

One important big-picture matplotlib concept is its object hierarchy.

If you've worked through any introductory matplotlib tutorial, you've probably called something like plt.plot([1, 2, 3]). This one-liner hides the fact that a plot is really a hierarchy of nested Python objects. A "hierarchy" here means that there is a tree-like structure of matplotlib objects underlying each plot.

You can think of the Figure object as a box-like container holding one or more Axes (actual plots). Below the Axes in the hierarchy are smaller objects such as tick marks, individual lines, legends, and text boxes. Almost every "element" of a chart is its own manipulable Python object, all the way down to the ticks and labels,

Matplotlib object is a tree-like structure

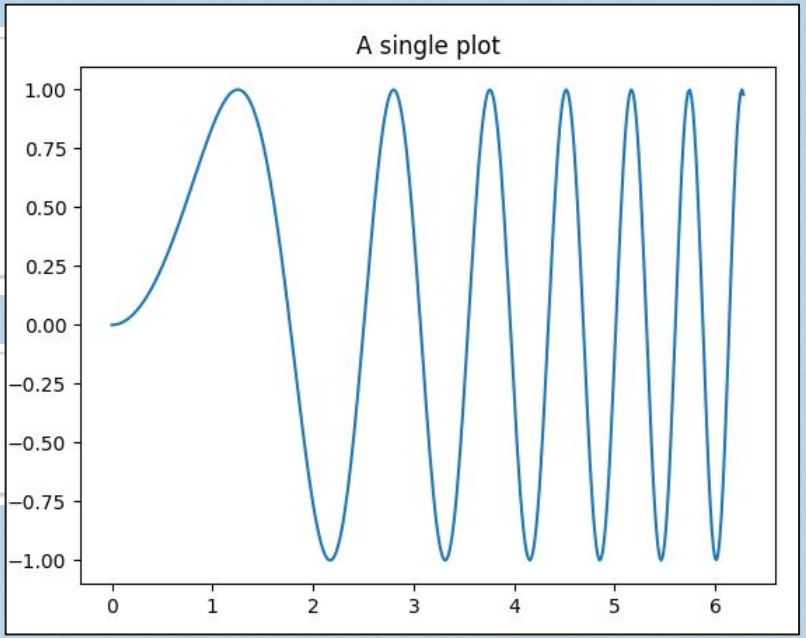
Figure: top of the tree-like structure (container)

Plots are called “axes” (the figure can have more than one)

Using plt.subplots for one subplot

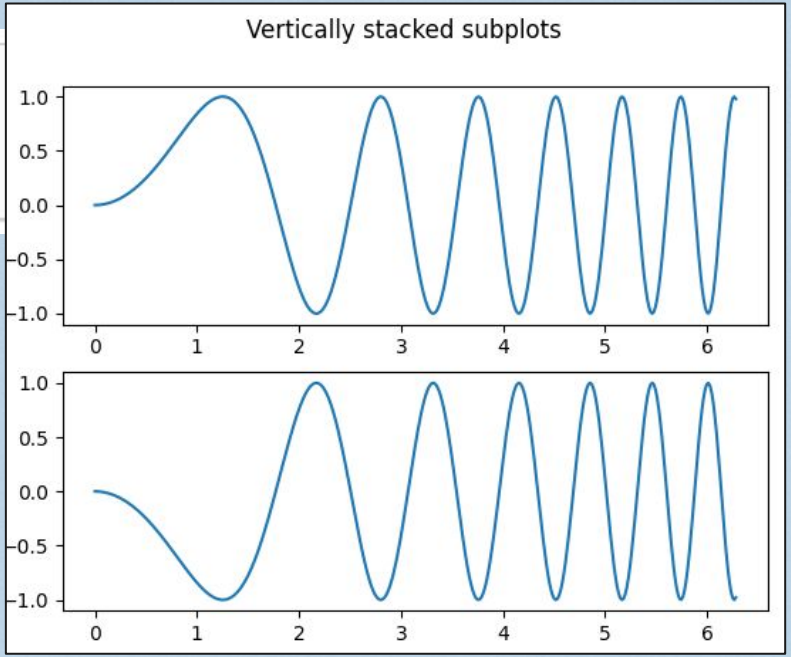
```
import matplotlib.pyplot as plt
import numpy as np

# Some example data to display
x = np.linspace(0, 2 * np.pi, 400)
y = np.sin(x ** 2)
```

```
fig, ax = plt.subplots()
ax.plot(x, y)
ax.set_title('A single plot')
```



Using plt.subplots for stacking subplots in one direction

```
fig, axs = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
axs[0].plot(x, y)
axs[1].plot(x, -y)
```

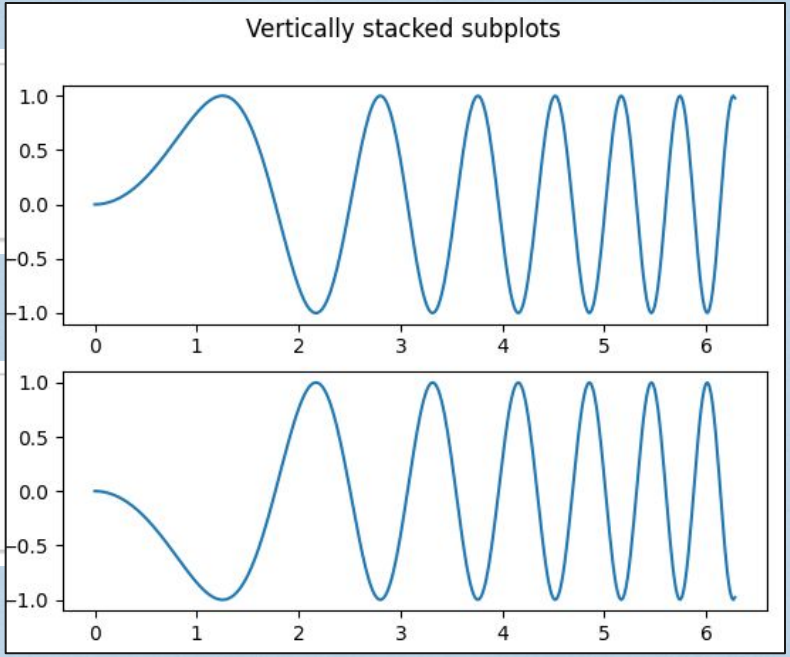


Using plt.subplots for stacking subplots in one direction

```
fig, axs = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
axs[0].plot(x, y)
axs[1].plot(x, -y)
```

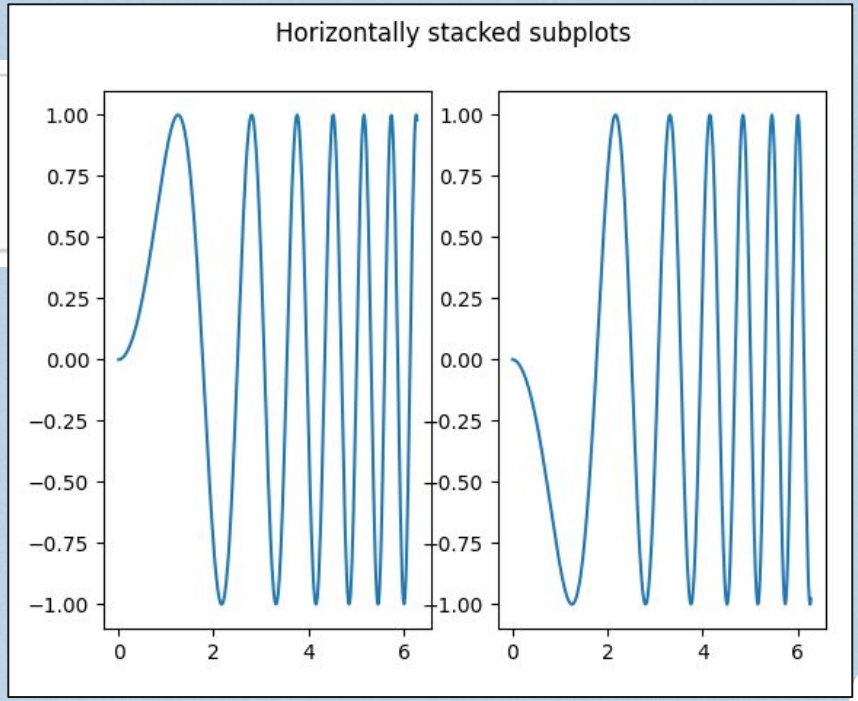
or

```
fig, (ax1, ax2) = plt.subplots(2)
fig.suptitle('Vertically stacked subplots')
ax1.plot(x, y)
ax2.plot(x, -y)
```



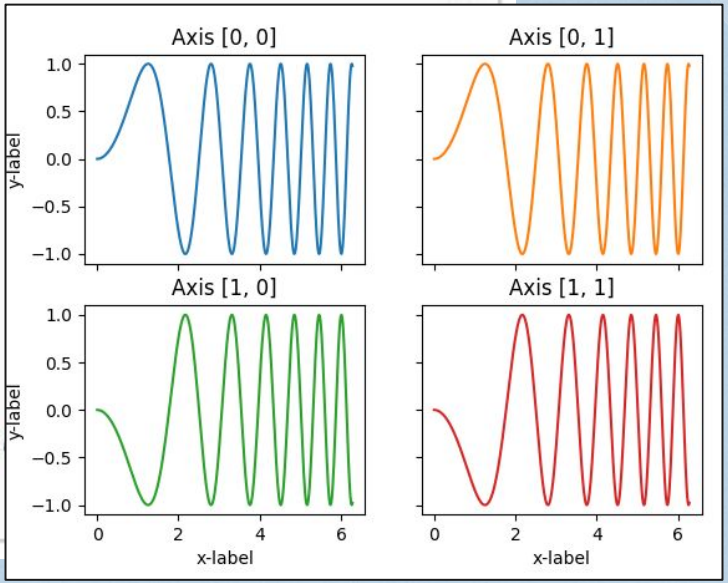
Horizontally stacked subplots

```
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Horizontally stacked subplots')
ax1.plot(x, y)
ax2.plot(x, -y)
```



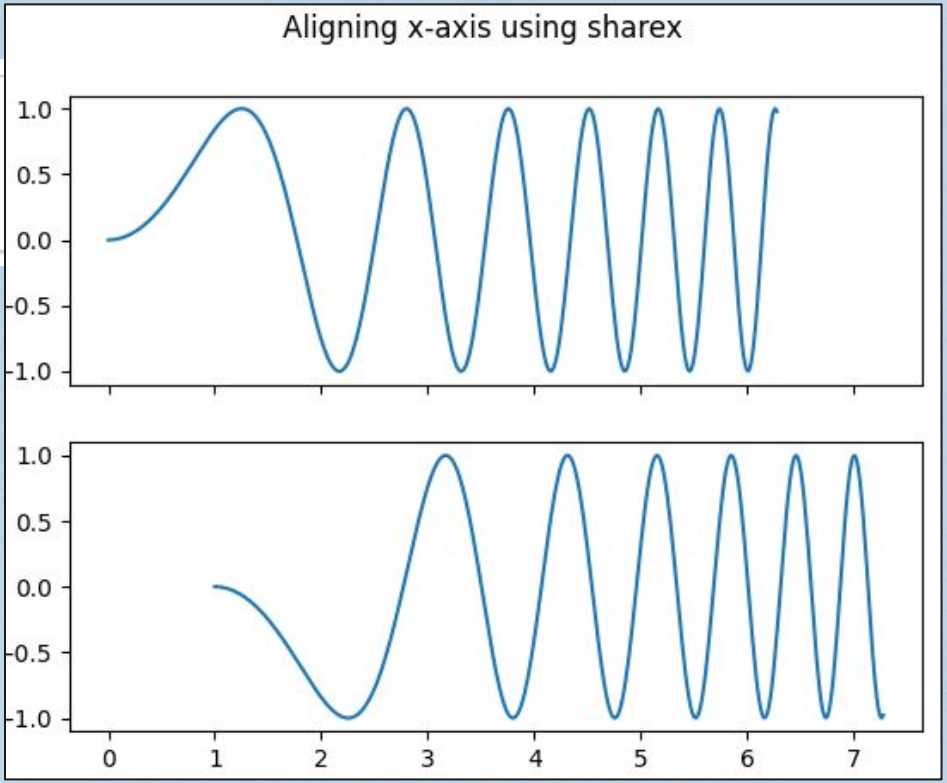
Stacking subplots in two directions

```
fig, axs = plt.subplots(2, 2)
axs[0, 0].plot(x, y)
axs[0, 0].set_title('Axis [0, 0]')
axs[0, 1].plot(x, y, 'tab:orange')
axs[0, 1].set_title('Axis [0, 1]')
axs[1, 0].plot(x, -y, 'tab:green')
axs[1, 0].set_title('Axis [1, 0]')
axs[1, 1].plot(x, -y, 'tab:red')
axs[1, 1].set_title('Axis [1, 1]')

for ax in axs.flat:
    ax.set(xlabel='x-label', ylabel='y-label')

# Hide x labels and tick labels for top plots and y ticks for right
for ax in axs.flat:
    ax.label_outer()
```



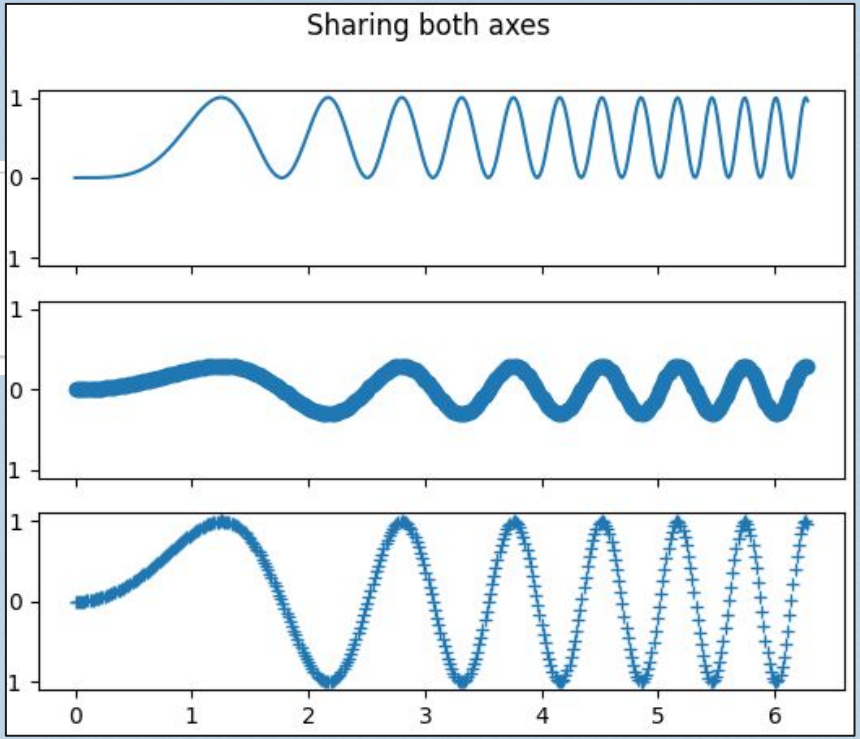
Sharing axes in subplots

```
fig, (ax1, ax2) = plt.subplots(2, sharex=True)
fig.suptitle('Aligning x-axis using sharex')
ax1.plot(x, y)
ax2.plot(x + 1, -y)
```



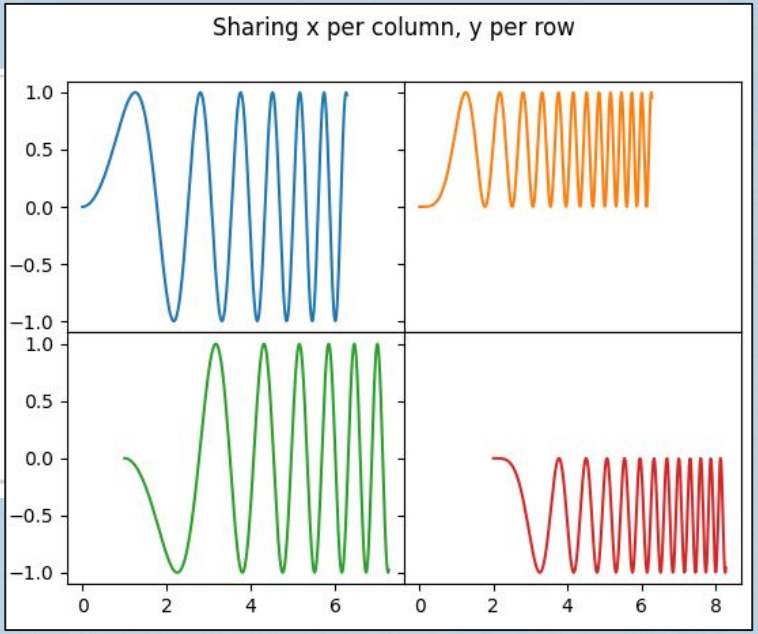
Sharing both axes

```
fig, axs = plt.subplots(3, sharex=True, sharey=True)
fig.suptitle('Sharing both axes')
axs[0].plot(x, y ** 2)
axs[1].plot(x, 0.3 * y, 'o')
axs[2].plot(x, y, '+')
```



Sharing *x* per column, *y* per row

```
fig = plt.figure()
gs = fig.add_gridspec(2, 2, hspace=0, wspace=0)
(ax1, ax2), (ax3, ax4) = gs.subplots(sharex='col', sharey='row')
fig.suptitle('Sharing x per column, y per row')
ax1.plot(x, y)
ax2.plot(x, y**2, 'tab:orange')
ax3.plot(x + 1, -y, 'tab:green')
ax4.plot(x + 2, -y**2, 'tab:red')

for ax in axs.flat:
    ax.label_outer()
```



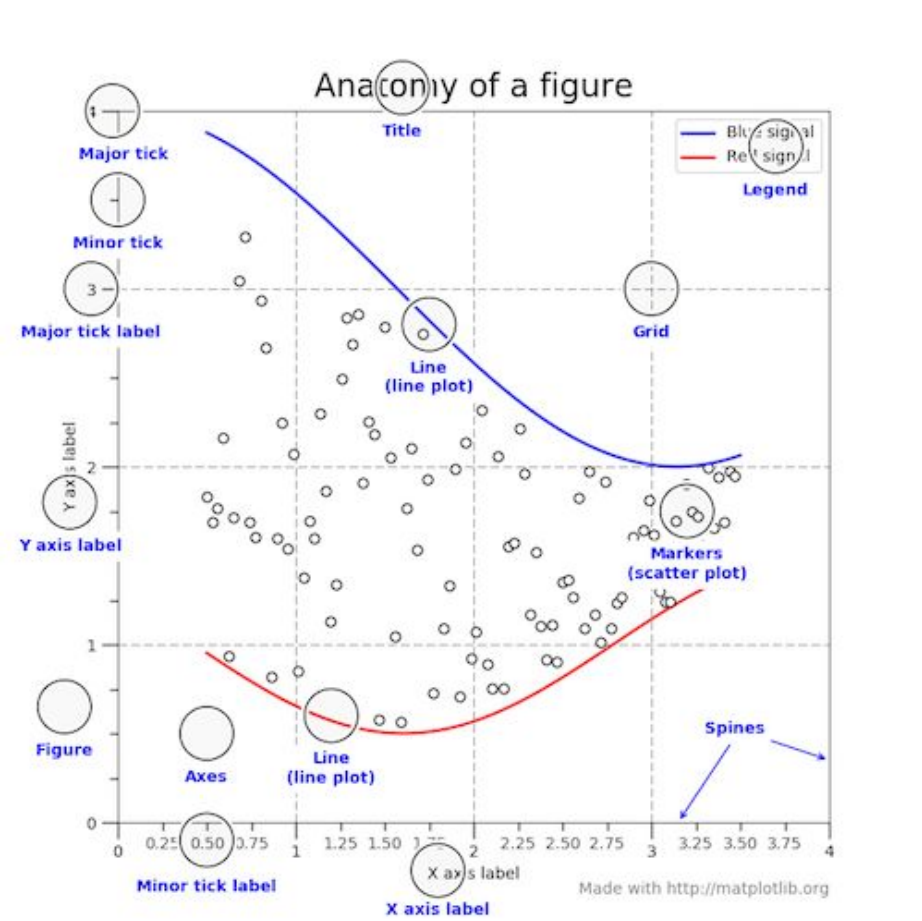


fig (a Figure class instance) has multiple Axes (a list, for which we take the first element). Each axes has a yaxis and xaxis, each of which have a collection of “major ticks,”

ARRAYS (LISTS)

An array is a collection of values of the same type saved under the same name.

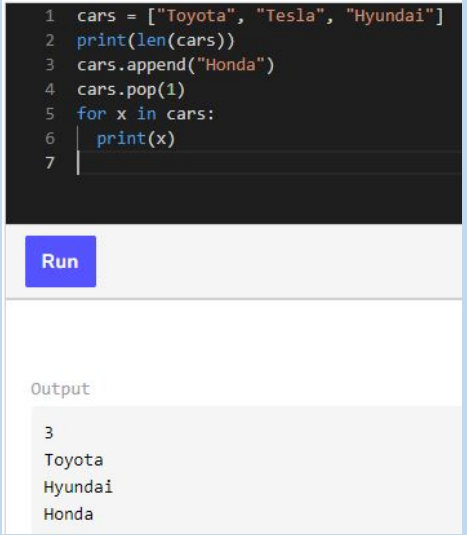
- Each value in the array is called an “element” and has an index
- value = array[index]
- The length of an array: the len() method
- Python array automatically scale up and down when elements are added/subtracted

Advantages:

- Simple to create and use data sequence
- Automatically scale to meet changing size requirement
- Used to create more complex data structures

Disadvantages:

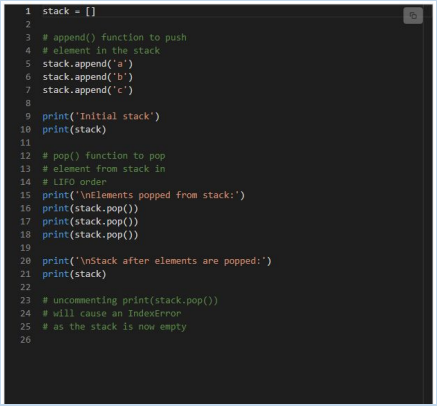
- Not optimized for scientific data (unlike Numpy array)



STACKS (LISTS)

A stack is a linear data structure that follows a particular order in which the operations are performed.

- Follows Last-in, First-out" (LIFO)(different version of queue)
- Insert new element at the top of the stack
- To access a middle element, you must first remove enough element to make it to the top
- Adding elements - push
    - Adding elements is known as a push, and removing elements is known as a pop. You can implement stacks in Python using the built-in list structure. With list implementation, push operations use the append() method, and pop operations use pop().
- Removing elements - pop



```
stack = []

# append() function to push
# element in the stack
stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)

# pop() function to pop
# element from stack in
# LIFO order
print('\nElements popped from stack:')
print(stack.pop())
print(stack.pop())
print(stack.pop())

print('\nStack after elements are popped:')
print(stack)

# uncommenting print(stack.pop())
# will cause an IndexError
# as the stack is now empty

```

GRAPHS (DICTIONARY)

Graphs are data structure used to represent a visual of relationships between data vertices (the node of a graph). The links that connect vertices together are called edges.

- Excellent for modeling networks or web-like structures
- Used to model social network sites like Facebook

Advantages:

- Quickly convey visual information through code
- Usable for modeling a wide range of real world problems
- Simple to learn syntax

Disadvantages:

- Vertex links are difficult to understand in large graphs
- Time expensive to parse data from a graph

When written in plain text, graphs have a list of vertices and edges:

```
V = {a, b, c, d, e}
E = {ab, ac, bd, cd, de}
```

In Python, graphs are best implemented using a dictionary with the name of each vertex as a key and the edges list as the values.

```python
# Create the dictionary with graph elements
graph = { "a" : ["b","c"],
          "b" : ["a", "d"],
          "c" : ["a", "d"],
          "d" : ["e"],
          "e" : ["d"]
        }

# Print the graph
print(graph)
```

QUEUES (COLLECTIONS.DEQUE)

An queue is a linear data structure that stores data in a "first in, first out" (FIFO) order.

- You can only pull the oldest element
- Great for order-sensitive tasks like online order processing and voicemail storage
- Use a Python list with append() and pop() methods to implement a queue. However, this is inefficient because lists must shift all elements by one index whenever you add a new element to the beginning.
    - Instead, it's best practice to use the deque class from Python's collections module.
- Deques are a generalization of stacks and queues (the name is pronounced "deck" and is short for "double-ended queue").
- Deques are optimized for the append and pop operations. The deque implementation also allows you to create double-ended queues, which can access both sides of the queue through the popleft() and popright() methods.

Advantages:

- Automatically orders data chronologically
- Scales to meet size requirements
- Time efficient with deque class

Disadvantages:

- Can only access data on the ends

```python
from collections import deque

# Initializing a queue
q = deque()

# Adding elements to a queue
q.append('a')
q.append('b')
q.append('c')

print("Initial queue")
print(q)

# Removing elements from a queue
print("\nElements dequeued from the queue")
print(q.popleft())
print(q.popleft())
print(q.popleft())

print("\nQueue after removing elements")
print(q)

# Uncommenting q.popleft()
# will raise an IndexError
# as queue is now empty
```

```
Output                                                    0.95s

Initial queue
deque(['a', 'b', 'c'])

Elements dequeued from the queue
a
b
c

Queue after removing elements
```

A linked list is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.

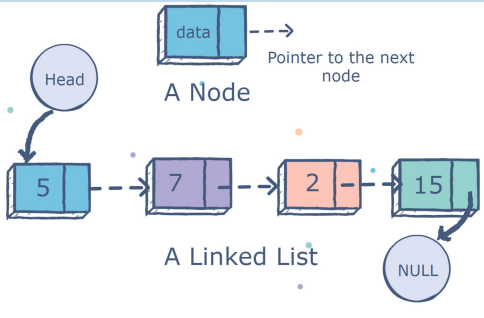
- Sequential collection of data - use relational pointers on each data node to link to the next node in the list
- Individual links only have a connection to their immediate neighbors
- All the links together form a larger structure
- Unlike arrays, linked lists do not have objective positions in the list. Instead, they have relational positions based on their surrounding nodes.
- The first node in a linked list is called the head node, and the final is called the tail node, which has a null pointer.
- Arrays allow random access and require less memory per element (do not need space for pointers) while lacking efficiency for insertion/deletion operations and memory allocation. On the contrary, linked lists are dynamic and have faster insertion/deletion time complexities.

Advantages:

- Efficient insertion and deletion
- Simpler to reorganize than arrays
- Useful as a starting point for advanced data structures like graphs

Disadvantages:

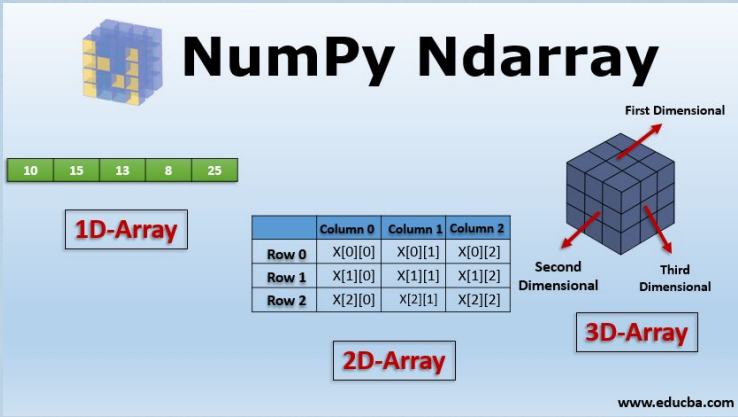
- Storage of pointers with each data point increases memory usage
- Must always traverse the linked list from head node to find a specific element



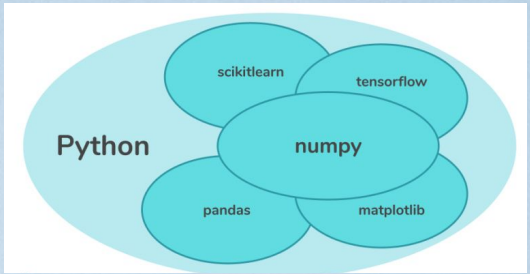
LINKED LISTS (CLASS)

INTRO: NUMPY ARRAYS

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.



At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.



CREATE A NUMPY ARRAY FROM PYTHON LIST

```
>>> import numpy as np
>>> a = np.array([2, 3, 4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

Numpy Arrays versus Python Lists

What is the difference?

1. Size - Numpy data structures take up less space
2. Performance - Numpy array runs faster
3. Functionality - Numpy has optimized functions such as built-in linear algebra operations

```
>>> a = np.array([1, 2, 3, 4, 5, 6])
>>> b = a[:2]
>>> b += 1
>>> print('a =', a, '; b =', b)
a = [2 3 3 4 5 6] ; b = [2 3]
```

```
>>> a = np.array([1, 2, 3, 4])
>>> b = a[:2].copy()
>>> b += 1
>>> print('a = ', a, 'b = ', b)
a =  [1 2 3 4] b =  [2 3]
```

Numpy Arrays *versus* Python Lists

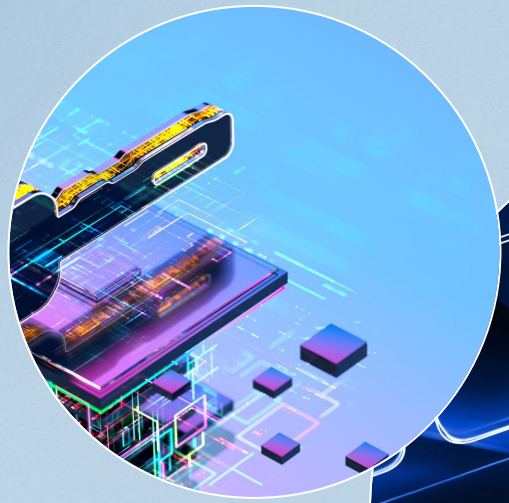
What is the difference?

- ***Size*** - Numpy data structures take up less space
- ***Performance*** - Numpy array runs faster
- ***Functionality*** - Numpy has optimized functions such as built-in linear algebra operations

"**Computer-based** visualization systems provide **visual representations** of datasets designed to help **people** carry out tasks more effectively."

Three key questions:

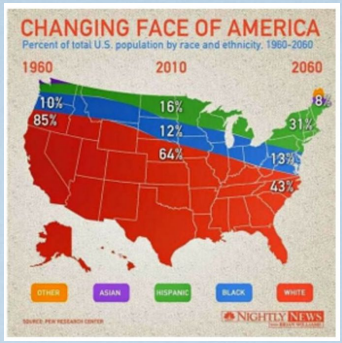
- **Why computers?** Computers have a large amount of memory storage and can finish more tasks much faster than humans can manually.
- **Why people?** While certain tasks can be fully automated, analysis questions are often ill-specified. People are needed to ask the correct question, detect patterns, conduct analyses, debug, and check for errors to verify and build trust in the model.
- **Why visual?** Our eyes are drawn to colors and patterns and we can see trends and outliers quickly. Thus, we can build upon pre existing hypotheses. It is storytelling with a purpose.



WHY IS EFFECTIVENESS IMPORTANT?

Even when we have the best intentions in communicating data through visuals, they don't always turn out the way we want. Sometimes it's the result of a mislabeled axis or a poor choice of color. Other times, we may choose the wrong type of chart entirely. Whatever the case, a bad data visualization can derail the message we want to communicate to the audience — or lead them to draw inaccurate conclusions.

KEY THINGS TO NOTE..



don't use misleading graphs

While interesting to look at, using a map doesn't help you understand what's going on here. For example, the fact that Oklahoma is red or that other states are partially different colors doesn't really mean anything. In addition, the racial and ethnic differences have no correlation to the states.

don't over-graph

This chart is too difficult to read. It should be broken up into a few separate visuals. Only visualize necessary and important variables and don't be afraid to make multiple graphs.

TABLE OF CONTENTS

JANE DOE

+34 654 321 432 | jane@freepik.com

"I always strive for **excellence**. I believe in the power of **collaboration**, and I love to build **meaningful relationships** with clients throughout the **creative process**"

EXPERIENCE

NOW-20XX — *POSITION A* — Place A
- Describe your work tasks here
- Describe your work tasks here

20XX-20XX — *POSITION B* — Place B
- Describe your work tasks here
- Describe your work tasks here

20XX-20XX — *POSITION C* — Place C
- Describe your work tasks here
- Describe your work tasks here

EDUCATION

20XX-20XX — *STUDIES 1*
- Competence
- Competence
- Competence

20XX-20XX — *STUDIES 2*
- Competence
- Competence
- Competence

20XX-20XX — *STUDIES 3*
- Competence
- Competence
- Competence

PHOTO SHOWCASE

This photo showcase is a collection of stunning images, each capturing the essence of my work. Let them inspire you!

WHAT PEOPLE SAY ABOUT ME

EMPLOYER 1

“I have been impressed by the quality, attention to detail and creative approach brought to every project made”

EMPLOYER 2

"This person's dedication and attention to detail significantly contributed to project success. Their creativity and ability to meet tight deadlines set them apart"

EMPLOYER 3

“This person has a strong technical understanding in order to develop successful strategies. Highly recommended for any kind of project or task”

EMPLOYER 4

"Working with this person has been valuable. Their strong work ethic and leadership qualities are instrumental in project success, combining a keen understanding of the bigger picture”

MY SKILLS AND QUALIFICATIONS

LEADERSHIP

"I possess leadership abilities that have been sharpened through past roles where I was responsible for the teams"

COMMUNICATIONS

"With excellent verbal and written skills, I can effectively convey complex messages to a variety of audiences"

TECHNICAL SKILL

"I am knowledgeable in several technical areas that support my ambition in my field"

PROBLEM-SOLVING

"My problem-solving capabilities have enabled me to solve difficult issues in the past, using creative and innovative solutions"

ORGANIZATION

"My proactivity allows me to plan ahead and prioritize tasks according to objectives and different deadlines"

INTERPERSONAL SKILL

"Teamwork is one of the essential part of my daily job, and I am sensitive towards different cultures"

200

The number of projects I have worked on so far

A CHART OF THE AREAS I WORK IN

10%

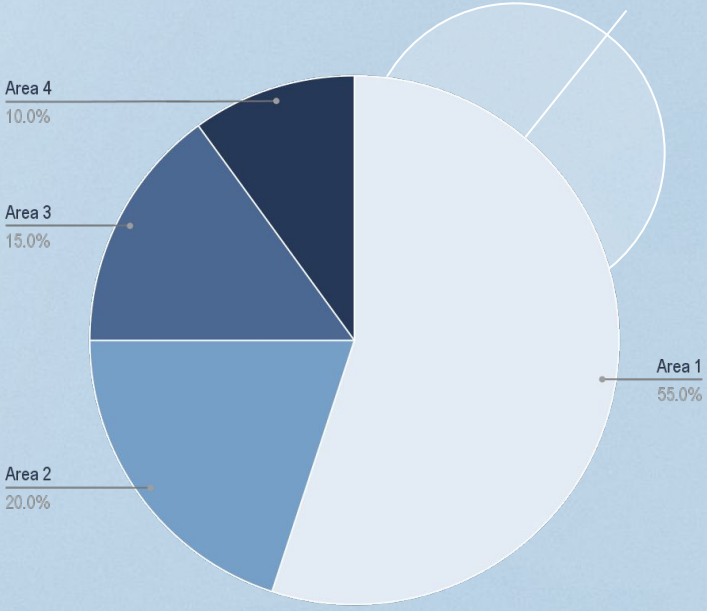
AREA 4

Describe the area here

15%

AREA 3

Describe the area here



55%

AREA 1

Describe the area here

20%

AREA 2

Describe the area here

Follow the link in the graph to modify its data and then paste the new one here. **For more info, click here**

TABLE

PROJECT TITLE	SKILLS	DATE	REF	CATEGORY
PROJECT 1	Describe the skills here	Jan 20XX	[Link or reference]	Write the category
PROJECT 2	Describe the skills here	Feb 20XX	[Link or reference]	Write the category
PROJECT 3	Describe the skills here	Mar 20XX	[Link or reference]	Write the category
PROJECT 4	Describe the skills here	Apr 20XX	[Link or reference]	Write the category
PROJECT 5	Describe the skills here	May 20XX	[Link or reference]	Write the category

HOW DID I ACHIEVE SUCCESS?

STEP 1

Identifying objectives

STEP 2

Research and preparation

STEP 3

Developing strategies

STEP 4

Plans of action

STEP 5

Measuring progress

STEP 6

Continuous improvement

STEP 7

Analyzing outcomes

STEP 8

Focusing on the future

ROADMAP OF MY FUTURE PROJECTS

INITIATIVE	*OBJECTIVE*	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Briefly describe the initiative 1	Describe the goal you want to achieve during the initiative	●	●	●									
Briefly describe the initiative 2	Describe the goal you want to achieve during the initiative		●	●	●	●	●	●					
Briefly describe the initiative 3	Describe the goal you want to achieve during the initiative					●	●	●					
Briefly describe the initiative 4	Describe the goal you want to achieve during the initiative			●	●					●			
Briefly describe the initiative 5	Describe the goal you want to achieve during the initiative								●	●	●		
Briefly describe the initiative 6	Describe the goal you want to achieve during the initiative							●				●	●

THANKS!

Do you have any questions?

youremail@freepik.com
+34 654 321 432
yourwebsite.com

ALTERNATIVE RESOURCES

Here's an assortment of alternative resources whose style fits that of this template:

VECTORS

- Gradient grainy gradient texture

RESOURCES

Did you like the resources used in this template? Get them on these websites:

VECTORS

- Gradient grainy gradient texture

PHOTOS

- Female artist in front of canvas
- People directing a new movie together
- Female artist painting
- Man holding sculpture side view
- Front view hand holding chisel
- Medium shot woman holding crossed arms
- Side view woman with photo camera
- Side view smiley girl with hat
- Young woman using a camera in her vacation

Instructions for use

If you have a free account, in order to use this template, you must credit **Slidesgo** by keeping the **Thanks** slide. Please refer to the next slide to read the instructions for premium users.

As a Free user, you are allowed to:

- Modify this template.
- Use it for both personal and commercial projects.

You are not allowed to:

- Sublicense, sell or rent any of Slidesgo Content (or a modified version of Slidesgo Content).
- Distribute Slidesgo Content unless it has been expressly authorized by Slidesgo.
- Include Slidesgo Content in an online or offline database or file.
- Offer Slidesgo templates (or modified versions of Slidesgo templates) for download.
- Acquire the copyright of Slidesgo Content.

For more information about editing slides, please read our FAQs or visit our blog:
https://slidesgo.com/faqs and **https://slidesgo.com/slidesgo-school**

Instructions for use (premium users)

As a Premium user, you can use this template without attributing **Slidesgo** or keeping the "**Thanks**" slide.

You are allowed to:

- Modify this template.
- Use it for both personal and commercial purposes.
- Hide or delete the “Thanks” slide and the mention to Slidesgo in the credits.
- Share this template in an editable format with people who are not part of your team.

You are not allowed to:

- Sublicense, sell or rent this Slidesgo Template (or a modified version of this Slidesgo Template).
- Distribute this Slidesgo Template (or a modified version of this Slidesgo Template) or include it in a database or in any other product or service that offers downloadable images, icons or presentations that may be subject to distribution or resale.
- Use any of the elements that are part of this Slidesgo Template in an isolated and separated way from this Template.
- Register any of the elements that are part of this template as a trademark or logo, or register it as a work in an intellectual property registry or similar.

For more information about editing slides, please read our FAQs or visit our blog:
https://slidesgo.com/faqs and **https://slidesgo.com/slidesgo-school**

Fonts & colors used

This presentation has been made using the following fonts:

Aboreto
(https://fonts.google.com/specimen/Aboreto)

Barlow
(https://fonts.google.com/specimen/Barlow)

#263857	#c2d8ec	#4a6891
#759fc7	#e3ecf5	#fdfeff

Storyset

Create your Story with our illustrated concepts. Choose the style you like the most, edit its colors, pick the background and layers you want to show and bring them to life with the animator panel! It will boost your presentation. Check out **how it works**.

Pana

Amico

Bro

Rafiki

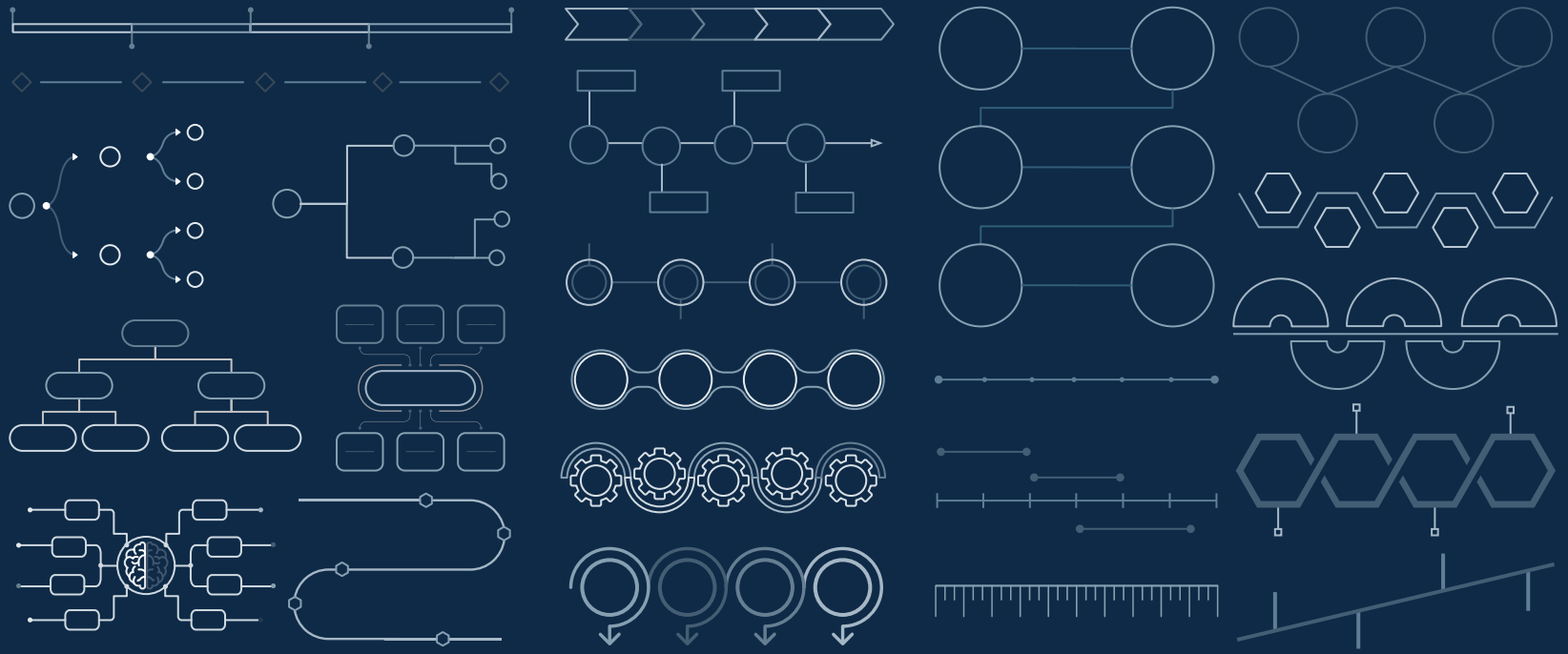
Cuate

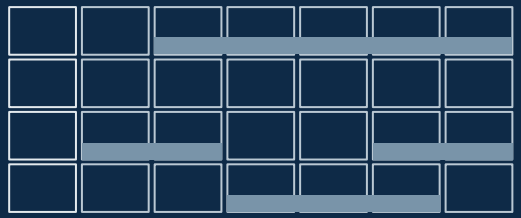
Use our editable graphic resources...

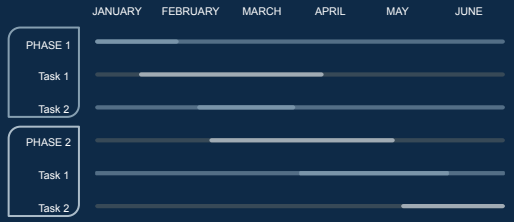
You can easily **resize** these resources without losing quality. To **change the color**, just ungroup the resource and click on the object you want to change. Then, click on the paint bucket and select the color you want. Group the resource again when you're done. You can also look for more **infographics** on Slidesgo.

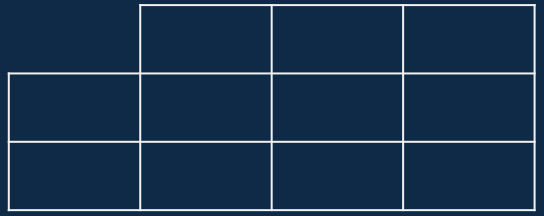


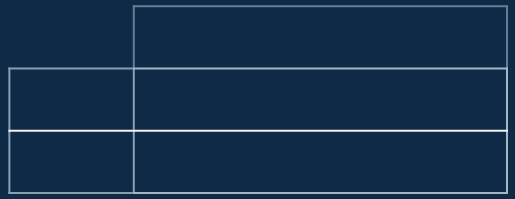


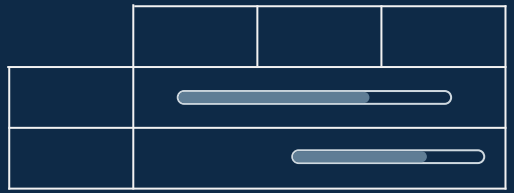



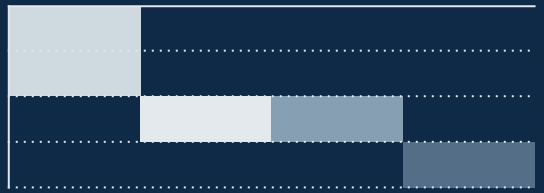


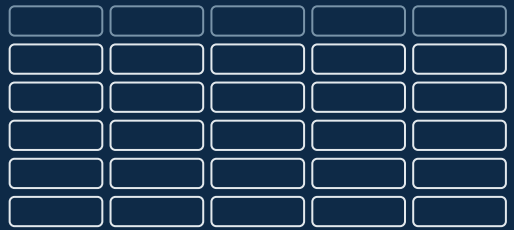

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

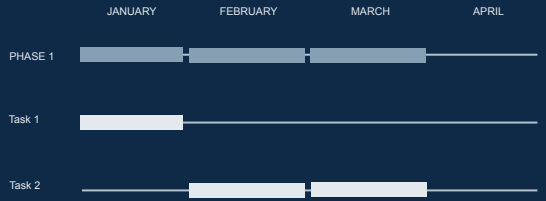


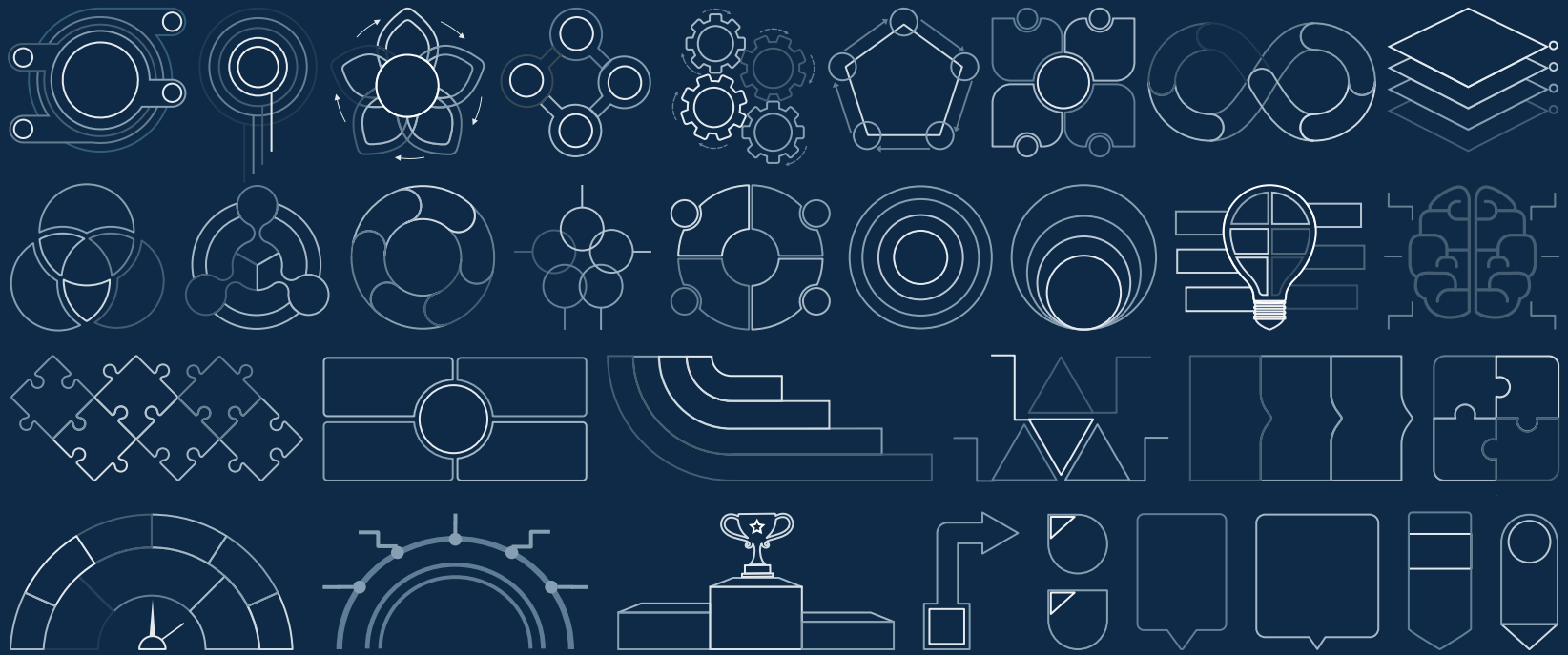


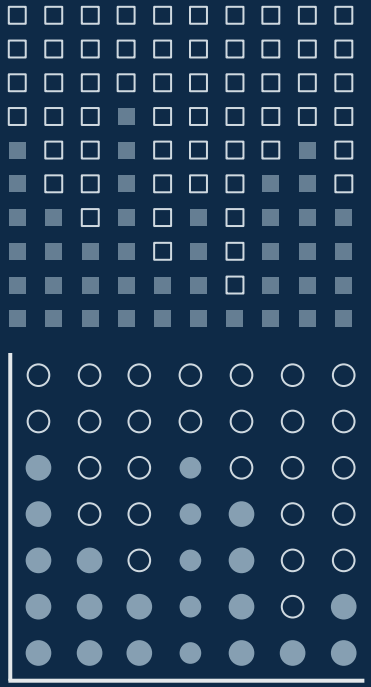
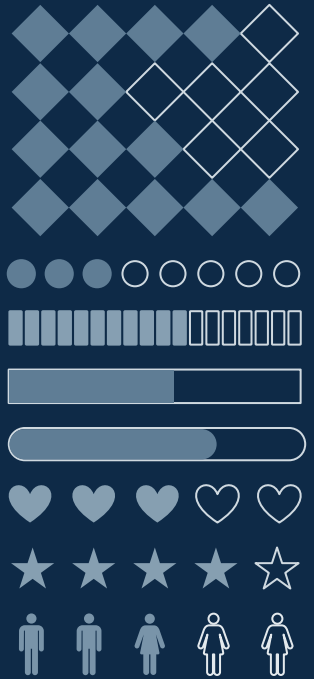
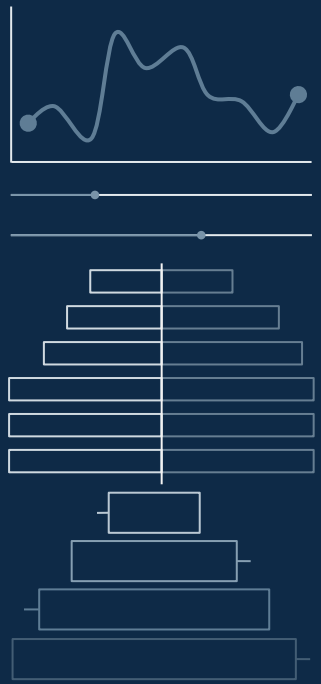
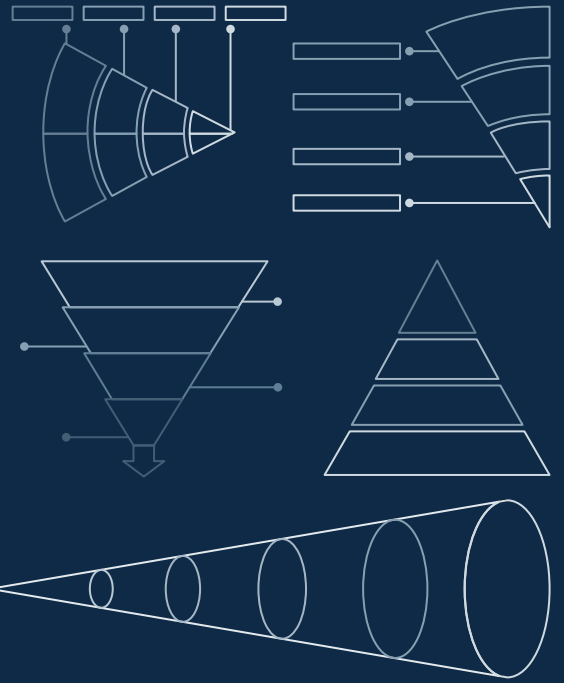







JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2



...and our sets of editable icons

You can **resize** these icons without losing quality.
You can **change the stroke and fill color**; just select the icon and click on the **paint bucket/pen**.
In Google Slides, you can also use **Flaticon's extension**, allowing you to customize and add even more icons.

Educational Icons

Medical Icons

Business Icons

Teamwork Icons

Help & Support Icons

Avatar Icons

Creative Process Icons

Performing Arts Icons

Nature Icons

SEO & Marketing Icons


slidesgo