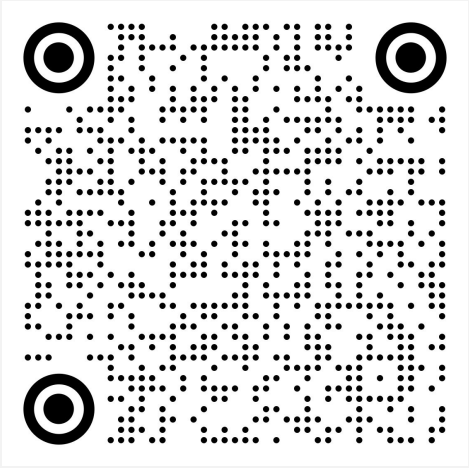
Unlock Xtreme 14.0
workshop series

Lecture one:
Fundamentals of
Python programming

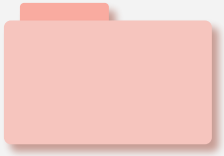
Unlock Xtreme 14.0

Lecture one:



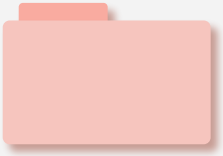
IEEE Student Branch
UNIVERSITY OF MELBOURNE

Overview



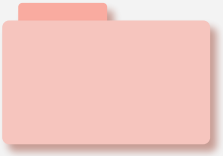
01.

Fundamentals of Python programming



02.

Live coding session to solve simple programming problems in an online platform



03.

Simulate what a live coding quiz feels like

Introduction

Python

A popular programming language used in a variety of applications (web-development, data analysis, machine learning, etc.).

Introduction

Python can...

Be used as a web-server. Run scripts for automation and create workflows. Connect to & manage databases. Read and modify files. Handle big data and perform complex math. Perform data analyses. Rapid software prototyping.

Introduction

Why Python?

Works on different platforms. Simple syntax similar to english. Lets the programmer write programs with fewer lines of code. Quick prototyping platform. Flexible language, i.e. can be used in many ways.

Introduction

Which Python?

We'll use Python3.

You can use it online for this tutorial...

An IDE is recommended for constant use: PyCharm, Thonny, Eclipse, ...

Introduction

Online Python:

https://repl.it/languages/python3

1. Hello, World!

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

2. Read user input

```
print(input())
```

2. Read user input

```
username = input("Enter username:")
print("Username is: " + username)
```

3. Add comments

```
# store username from user input
username = input("Enter username:")

# print the stored username
print("Username is: " + username)
```

3. Add comments

```
# store username from user input
username = input("Enter username:")

"""
you can have
comments in multiple lines
print the stored username
"""
print("Username is: " + username)
```

4. Python variables

```
# store username from user input
username = input("Enter username:")
```

4. Python variables

```
# variable assignment using =

x = 5
y = "Sina"
print(x)
print(y)
```

5. Python Data Types

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

5. Python Data Types

Text Type:	`str`	x = "Sina"
Numeric Types:	`int`, `float`, `complex`	x = 5, x = 5.5
Sequence Types:	`list`, `tuple`, `range`	
Mapping Type:	`dict`	
Set Types:	`set`, `frozenset`	
Boolean Type:	`bool`	x = True, x = False
Binary Types:	`bytes`, `bytearray`, `memoryview`	

5. Python Data Types

```
# checking a variable's data type:

x = 5
print(type(x))
```

6. Casting

To specify type of a variable:

```
int(), float(), str()
```

6. Casting

```
# casting a variable to int:

x = int(1)       # x will be 1
y = int(2.8)     # y will be 2
z = int("3")     # z will be 3
```

6. Casting

```
# casting a variable to float:

x = float(1)          # x will be 1.0
y = float(2.8)        # y will be 2.8
z = float("3")        # z will be 3.0
w = float("4.2")      # w will be 4.2
```

6. Casting

```
# casting a variable to string:

x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

6. Casting

```
# sample application of casting:
# ask a user their birth year, tell them their age!

birth_year = int(input("What year were you born?"))

age = 2020 - birth_year

print("You should be " + str(age) + " now!")
```

7. Strings

```
# strings store an ordered list of characters:

print("Hello")
print('Hello')
```

7. Strings

```
# strings store an ordered list of characters:

a = "Hello"
print(a)
```

7. Strings

```
# strings store an ordered list of characters:

a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

7. Strings

```
# strings store an ordered list of characters:

a = "Hello, World!"
print(a[1])
```

7. Strings

```
# strings store an ordered list of characters:

b = "Hello, World!"

print(b[2:5])

print(b[-5:-2])

print(len(b))
```

7. Strings

```
# string methods:

a = "   Hello, World!   "

print(a.strip())              # returns "Hello, World!"

print(a.lower())              # returns "   hello, world!   "

print(a.upper())              # returns "   HELLO, WORLD!   "

print(a.replace("l", "J"))    # returns "   HeJJo, WorJd!   "
```

7. Strings

```
# string methods:

a = "Hello, World!"

print(a.split(","))          # returns ['Hello', ' World!']

x = 'ello' in a              # x is True

x = 'allo' in a              # x is False
```

7. Strings

```
# string concatenation:

a = "Hello"
b = "World"
c = a + b               # c is 'HelloWorld!'
c = a + " " + b         # c is 'Hello World!'
```

7. Strings

```
# string formatting (inserting numerals):

# inserting with casting:
print("I have " + str(5) + " apples")

# inserting with string format method:
print("I have {} apples".format(5))
print("I have {} apples and {} oranges".format(5, 10))
```

7. Strings

Special characters (character escaping):

```
\’    \”    \\    \n    \t
```

8. Boolean

In programming we often use **expressions** which are like yes/no questions:

```
True / False
```

8. Boolean

```
a = 10
b = 20

# is equal:
a == b          # False

# is lesser:
a < b           # True

# is greater:
a > b           # False
```

8. Boolean

```
a = 10
b = 20

# is not equal:
a != b          # True

# is lesser or equal:
a <= b          # True

# is greater or equal:
a >= b          # False
```

9. Operators

Arithmetic operators (for numeral data):

+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y

9. Operators

Arithmetic operators (for numeral data):

%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor Division	x // y

9. Operators

Assignment operators:

=	**x = 10**	
+=	x += 10	x = x + 10
-=	x -= 10	x = x - 10
*=	x *= 10	x = x * 10
\=	x \= 10	x = x \ 10

9. Operators

Comparison operators:

`==`	is equal
`!=`	is no equal
`>=`	is greater or equal
`<=`	is lesser or equal
`>`	is greater
`<`	is lesser

9. Operators

Logical operators (used for Boolean data):

`and`	True if both are True	a `and` b
`or`	True if at least one is True	a `or` b
`not`	Reverse	`not` a

9. Operators

Other operators:

`is , is not`	a `==` b , a `!=` b
`in , not in`	used to check if an element is in a list

Other Data types

a collection of items

list:	`[]`	**ordered, changeable, duplicates allowed**
Tuple:	`()`	**ordered, unchangeable, duplicates allowed**
set:	`{}`	**unordered, unindexed, no duplicates**
list:	`[]`	**unordered, changeable, indexed, no duplicates**

10. Lists

```
# create a list
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

10. Lists

```
# index in a list
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

# second item: (0: first)
print(thislist[1])

# last item: (-1: last)
print(thislist[-1])

# select a slice:
print(thislist[2:5])          print(thislist[2:])          print(thislist[:5])          print(thislist[:-1])
```

10. Lists

```
# change a value
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

# second item: (0: first)
thislist[1] = "blackcurrant"
print(thislist)
```

10. Lists

```
# check for a value
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

"cherry" in thislist          # True

"pear" in thislist            # False
```

10. Lists

```
# check length
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(len(thislist))                # prints 7
```

10. Lists

```
# add item to list
thislist = ["apple", "banana"]

thislist.append("orange")

print(thislist)                    # prints [“apple”, “banana”, “orange”]
```

10. Lists

```
# insert item to list
thislist = ["apple", "banana"]

thislist.insert(1, "orange")

print(thislist)                    # prints [“apple”, “orange”, “banana”]
```

10. Lists

```
# remove item from list
thislist = ["apple", "banana", "orange", "cherry"]

thislist.remove("orange")          # removes only the first occurence, gives error if not existing

print(thislist)                    # prints [“apple”, “banana”, “cherry”]

# other deletion methods:
thislist.pop()                     # removes the last item
del thislist[0]                    # remove by index
thislist.clear()                   # remove all items
```

10. Lists

```
# copy a list
list2 = list1                           # WRONG!

thislist = ["apple", "banana", "orange", "cherry"]

mylist = thislist.copy()
print(mylist)                           # mylist will be a copied replica.
                                        # (changing the copy will not change the original)

# alternative:
mylist = list(thislist)
```

10. Lists

```
# join 2 lists:
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]

list3 = list1 + list2
print(list3)
```

11. Tuples

```
thistuple = ("apple", "banana", "cherry")     # create a tuple

print(thistuple[1])                           # access element

"apple" in thistuple                          # True

len(thistuple)                                # 3

thistuple[3] = "orange"                       # This will raise an error

tuple3 = tuple1 + tuple2                      # joins tuples
```

11. Tuples

```
# where and why use tuples:

# multiple assignments
a, b = 3, 5

# swap in one line
a, b = b, a

# can be used in a method/function to return multiple values
return (a, b)
```

12. Sets

```
# unordered, unindexed, changeable, no duplicates

thisset = set()                          # create empty set

thisset = {"apple", "banana", "cherry"}  # create a set with values

"banana" in thisset                      # check for item (very efficient)
```

12. Sets

```
# unordered, unindexed, changeable, no duplicates

# change items

thisset.add("orange")                          # add an item

thisset.update(["orange", "mango", "grapes"])  # add multiple items

len(thisset)                                   # get length
```

12. Sets

```
# unordered, unindexed, changeable, no duplicates

thisset.remove("banana")          # remove an item (raises error if missing)

thisset.discard("banana")         # remove an item (no error if missing)

thisset.clear()                   # remove all items

set3 = set1.union(set2)           # join/combine 2 sets

set1.update(set2)                 # in place union
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys

thisdict = {}                          # create empty dictionary

thisdict = {                           # create a dictionary with key-value pairs
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys


x = thisdict["model"]                    # access value of a key (error if missing)

x = thisdict.get("model")                # access value of a key (no error if missing, None instead)

x = thisdict.get("model", "unknown")     # return desired value if missing
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys

thisdict["year"] = 2018                # update/change a value

thisdict.keys()                        # get a list of all keys

thisdict.values()                      # get a list of all values

"model" in thisdict                    # check if a key is existing

len(thisdict)                          # get number of keys in a dictionary
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys

thisdict["color"] = "red"          # add an item

thisdict.pop("model")              # remove an item

del thisdict["model"]              # remove an item (error if missing key)

thisdict.clear()                   # remove all keys
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys


mydict = thisdict.copy()                # copy a replica

mydict = dict(thisdict)                 # copy a replica
```

12. Dictionaries

```
# key-value pairs, unordered, indexed (by key), changeable, no duplicate keys

# nested dictionaries:
myfamily = {
  "child1" : {"name" : "Emil", "year" : 2004},
  "child2" : {"name" : "Tobias", "year" : 2007},
  "child3" : {"name" : "Linus", "year" : 2011}
}

myfamily["child2"]["name"]
```

13. Execution

Code execution by line by line interpretation:

1. **Read every line and execute**
2.
3. **Note: in python indents are important**
4.

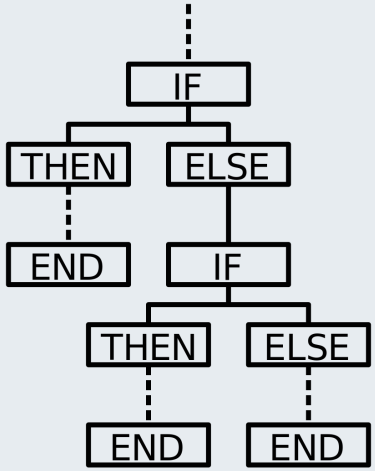
14. Conditionals

Other operators:

`if , elif, else`

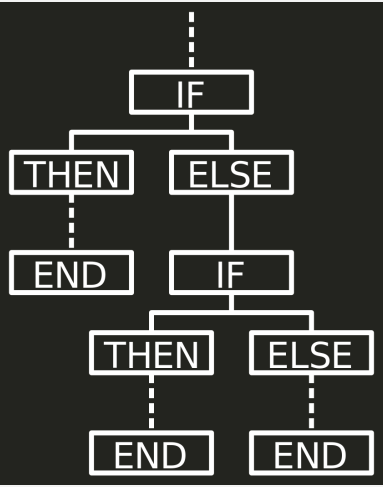
Syntax:

```
if:
        indented lines...
```



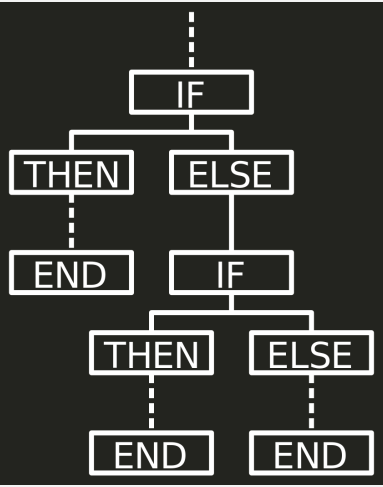
14. Conditionals

```
# How to use conditionals:

a = 33
b = 200
if b > a:
  print("b is greater than a")
```



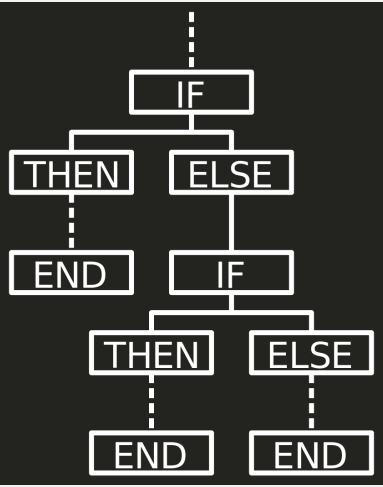
14. Conditionals

```
# How to use conditionals:

a = 23
b = 23
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```



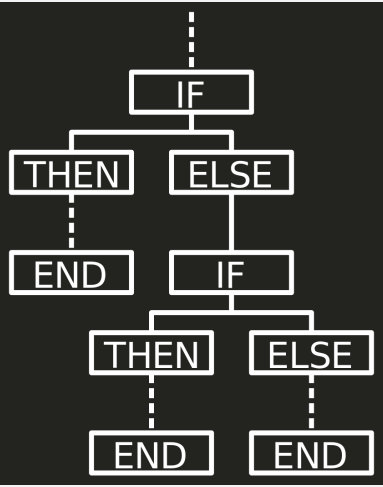
14. Conditionals

```
# How to use conditionals:

a = 330
b = 200
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```



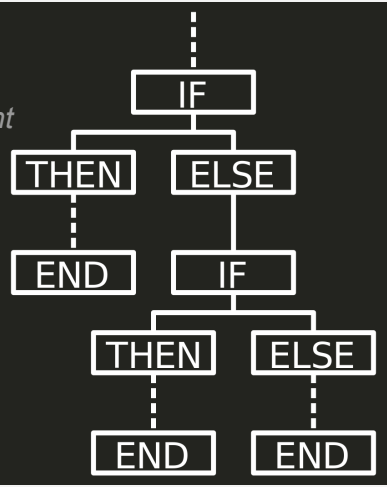
14. Conditionals

```
# How to use conditionals:

a = 330
b = 200
if b > a:
  print("b is greater than a")
else:
  print("a is not smaller than b")
```



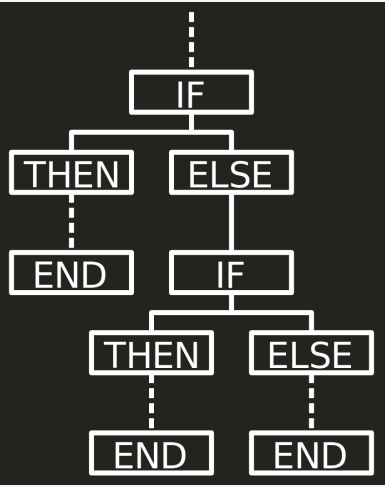
14. Conditionals

```
# How to use conditionals in one line:

if a > b: print("a is greater than b")                          # one line if statement

print("A") if a > b else print("B")                             # one line if + else

print("A") if a > b else print("=") if a == b else print("B")   # can be extended...
```



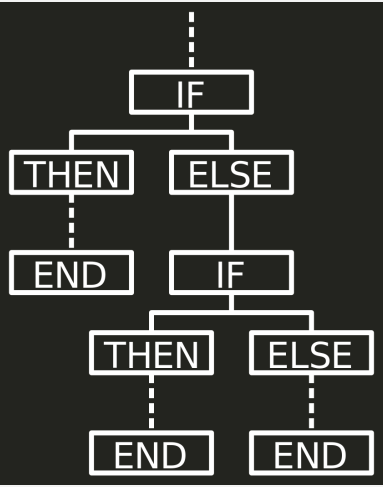
14. Conditionals

```
# How to combine conditionals with logical operators


if a > b and c > a:
  print("Both conditions are True")


if a > b or a > c:
  print("At least one of the conditions is True")
```



14. Conditionals

```
# nested conditionals:

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")

# small note: use pass to avoid getting an error
if b > a:
  pass
```

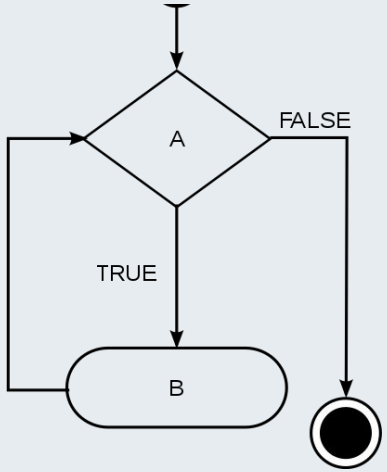


Loops

What if you wanted to run
a part of your code
multiple times?

Loops

What if you wanted to run a part of your code multiple times?



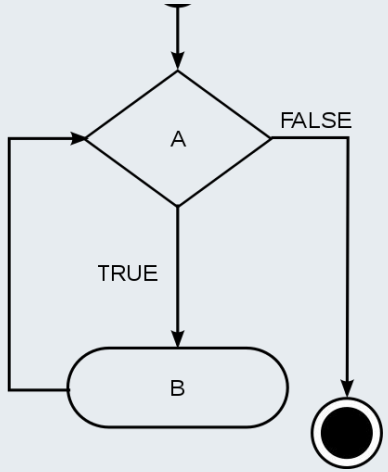
15. While loops

Python while loop:

`while, break, continue`

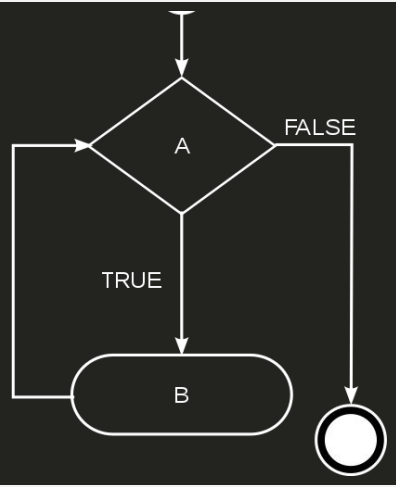
Syntax:

```
while check:
        indented lines...
```



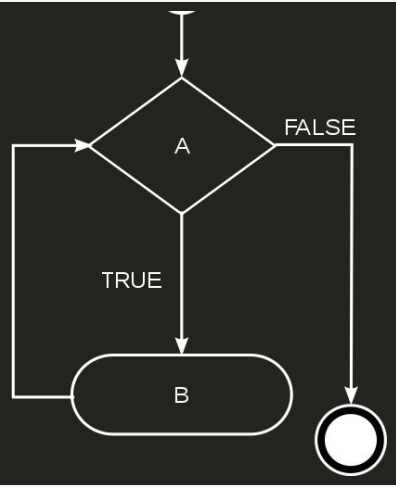
15. While loops

```
# simple while loop:


i = 1
while i < 6:
  print(i)
  i += 1
```



15. While loops

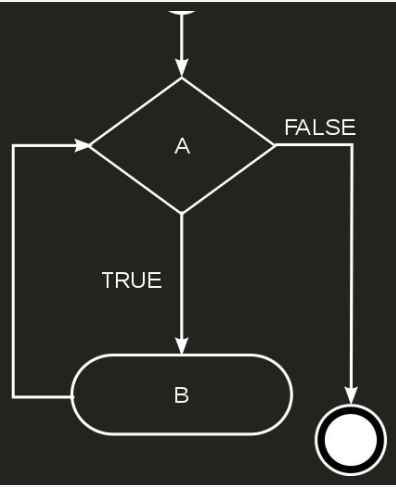
```
# break statement is used to end the loop before meeting condition:


i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```



15. While loops

*# **continue statement** is used to skip the current iteration of the loop*

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```



16. For loops

Python for loop: to iterate over an iterable class (list, dictionary, …)

Run something for each item in a collection

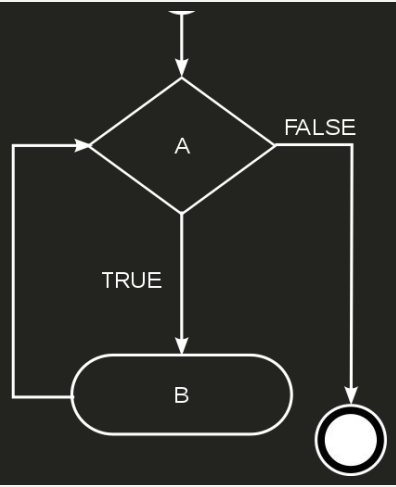
`for, in, break, continue`

Syntax:

```
for item in collection:
        indented lines...
```

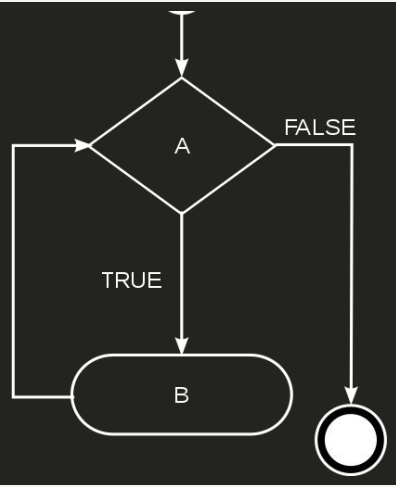
16. For loops

```
# iterate over a list with for loop


fruits = ["apple", "banana", "cherry"]

for x in fruits:
  print(x)
```



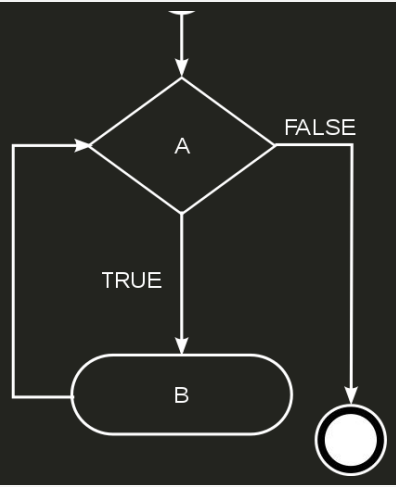
16. For loops

```
# loop through a string


for x in "banana":
  print(x)
```



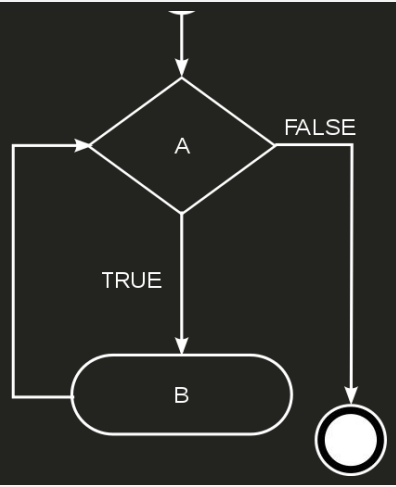
16. For loops

```
# loop through a dictionary

# create a dictionary with key-value pairs
thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

# loop through all items
for x in this dict:
  print("{} is {}".format(x, dict[x]))

for x, y in thisdict.items():
  print("{} is {}".format(x, y))
```



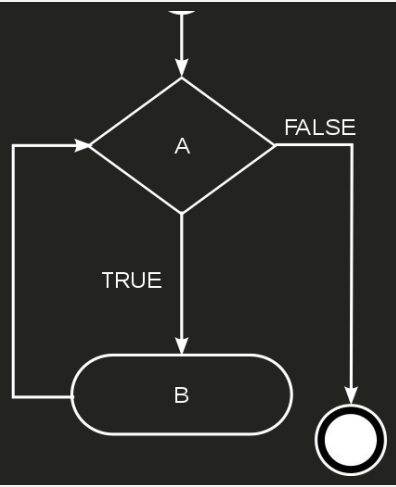
16. For loops

```
# break statement to break out of a loop

fruits = ["apple", "banana", "cherry"]

for x in fruits:
  print(x)
  if x == "banana":
    break
```



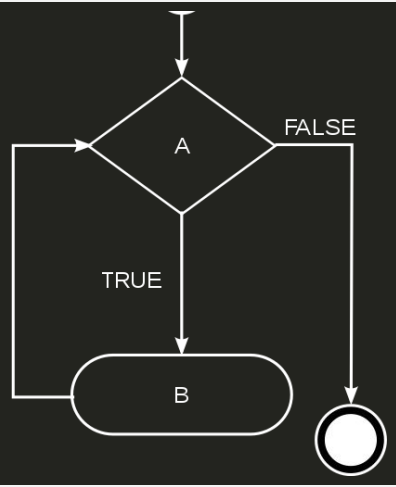
16. For loops

```
# continue statement to skip over an iteration step

fruits = ["apple", "banana", "cherry"]

for x in fruits:
  if x == "banana":
    continue
  print(x)
```



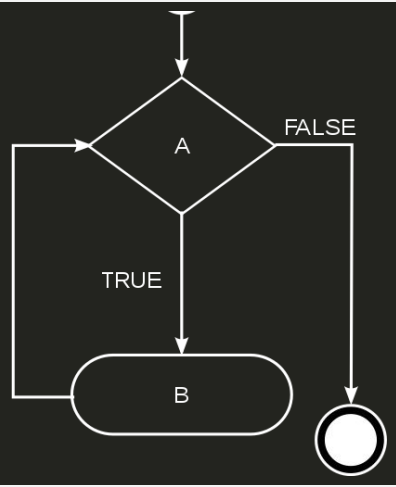
16. For loops

```
# range function to repeat something

for x in range(6):                  # 0, 1, 2, 3, 4, 5
  print(x)

for x in range(2, 6):               # start from 2: 2, 3, 4, 5
  print(x)

for x in range(2, 30, 3):           # step size of 3: 2, 5, 8, ...
  print(x)
```



16. For loops

```
# nested for loops

adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)

# to avoid errors without indented content
for x in [0, 1, 2]:
  pass
```



17. Functions

Python function/method: to create you own blackbox
(get a set of defined inputs, generate desired output)

`def, return`

Syntax:

```
def function_name(inputs):
    indented lines...
```

17. Functions

```
# define a simple function


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


# call the function
my_function()
```

17. Functions

```
# a function with one input argument/parameter


def my_function(fname):
  print(fname + " Jonas")

my_function("Kevin")
my_function("Joe")
my_function("Nick")
```

17. Functions

```
# a function with two input arguments/parameters


def my_function(fname, lname):
  print(fname + " " + lname)

my_function("Kevin", "Jonas")
```

17. Functions

```
# multiple inputs as a list


def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Kevin", "Joe", "Nick")
```

17. Functions

```
# inputs by keyword arguments


def my_function(child3, child2, child1):
  print("The youngest child is " + child3)


my_function(child1 = "Kevin", child2 = "Joe", child3 = "Nick")
```

17. Functions

```
# arbitrary keyword arguments


def my_function(**kid):
  print("His last name is " + kid["lname"])


my_function(fname = "Kevin", lname = "Jonas")
```

17. Functions

```
# default parameter for a function


def my_function(city = "Melbourne"):
  print("I am in " + city)


my_function("Perth")
my_function()
```

17. Functions

```
# function with a return value


def my_function(x):
  return 5 * x


print(my_function(3))
a = my_function(5)
```

17. Functions

```
# recursion (calling a function within itself)

def my_function(x):
  if (x > 0):
    print(x)
    my_function(x-1)
  else:
    return

my_function(5)
```

IEEE IEEE Student Branch
UNIVERSITY OF MELBOURNE

Live Coding

https://www.hackerrank.com

Live Coding: task 1, hello world

https://www.hackerrank.com/challenges/py-hello-world

Live Coding: task 2, conditionals

https://www.hackerrank.com/challenges/py-if-else

Live Coding: task 3, loops

https://www.hackerrank.com/challenges/python-loops

Live Coding: task 4, puzzle

https://www.hackerrank.com/challenges/find-second-maximum-number-in-a-list

Live Coding: bonus task, functions

https://www.hackerrank.com/challenges/write-a-function

IEEE IEEE Student Branch
UNIVERSITY OF MELBOURNE

Workshop 1: Programming contest

https://www.hackerrank.com/workshop-1-unlock-xtreme-14-0-melbourne


POSTERA CRESCAM LAUDE
THE UNIVERSITY OF
MELBOURNE


IEEE Women in Engineering
We


IEEE
Computational
Intelligence
Society


IEEE
WOMEN IN
ENGINEERING
The University of Melbourne


IEEE
computer
society


IEEE Victorian Section


IEEE
Region 10

Thanks!

facebook.com/ieeeunimelb

youtube.com/channel/UCxvk512F22jTdSWfEQVuCTw/

ieee.unimelb@gmail.com
edu.ieee.org/au-unimelb


slidesgo

10. Loop through a list

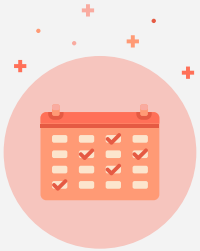
```
# legos
```

Fonts & colors used

This presentation has been made using the following fonts:

Hepta Slab
(https://fonts.google.com/specimen/Hepta+Slab)

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



OS Weekly Planner for Teachers

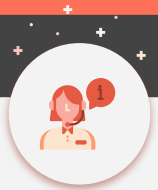
Start now!

Contents of This Template

Here's what you'll find in this **Slidesgo** template:

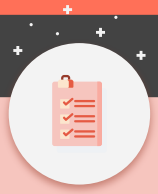
1. A slide structure based on a multi-purpose presentation, which you can easily adapt to your needs. For more info on how to edit the template, please visit **Slidesgo School** or read our **FAQs**.
2. An assortment of illustrations that are suitable for use in the presentation can be found in the **alternative resources slide**.
3. A **thanks slide**, which you must keep so that proper credits for our design are given.
4. A **resources slide**, where you'll find links to all the elements used in the template.
5. **Instructions for use**.
6. Final slides with:
    1. The **fonts and colors** used in the template.
    2. More **infographic resources**, whose size and color can be edited.
    3. Sets of **customizable icons** of the following themes: general, business, avatar, creative process, education, help & support, medical, nature, performing arts, SEO & marketing, and teamwork.

You can delete this slide when you're done editing the presentation.



Contact Me

Here you could describe the topic of the section

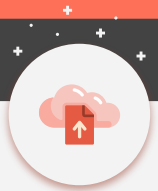


Our Week

Here you could describe the topic of the section

Homework

Here you could describe the topic of the section



Upload

Here you could describe the topic of the section

How to Contact Me

658-956-87

youremail@freepik.com

"This is a quote, words full of wisdom that someone important said and can make the reader get inspired."

—Someone Famous

Contact

Video Conference

Monday

Tuesday

Wednesday

Thursday

Friday

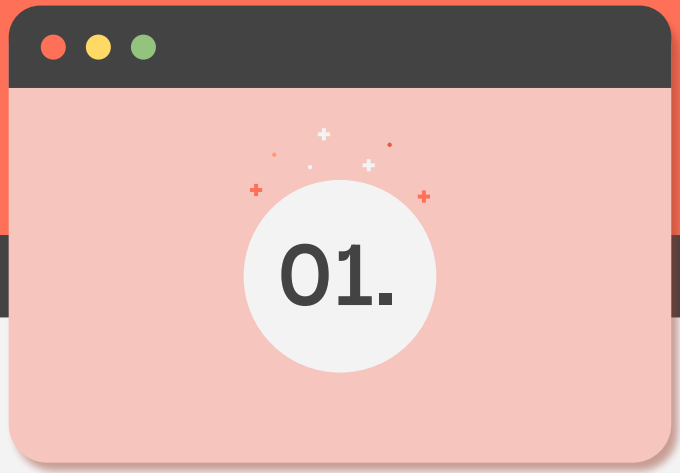
Homework

Our Week

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon

Videoconference Hours

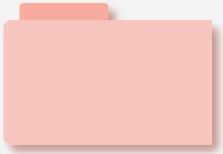
Monday	Tuesday	Wednesday	Thursday	Friday
10:45 - 11:45	10:45 - 11:45	10:45 - 11:45	10:45 - 11:45	10:45 - 11:45



Our Week

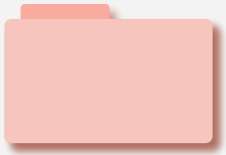
You could enter a subtitle here if you need it

Monday



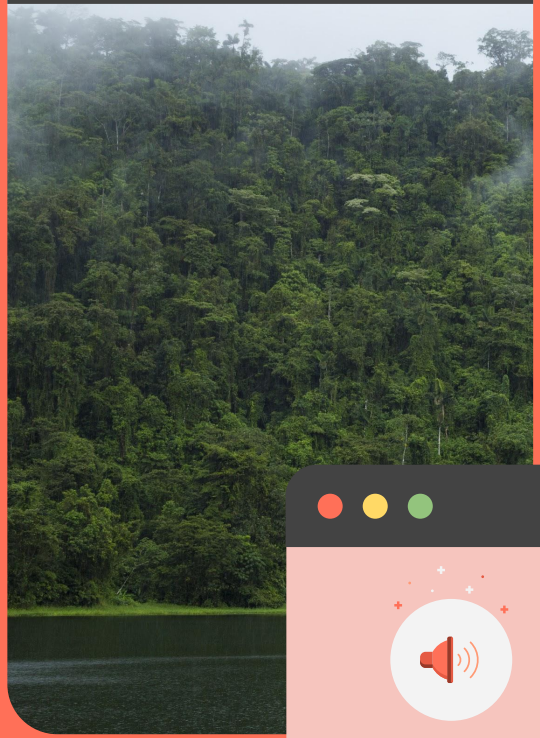
01.

Mercury is the closest planet to the Sun and also the smallest one in the Solar System



02.

Jupiter is the biggest planet in the Solar System and the fourth-brightest object in the sky



Natural Science

a. Jupiter is a gas giant and the biggest planet in the Solar System

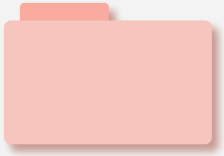
b. Saturn is composed mostly of hydrogen and helium

c. Neptune is the fourth-largest planet in the Solar System

d. Venus has a beautiful name and is the second planet from the Sun

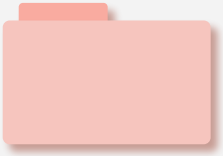
A Picture Is Worth a Thousand Words

Tuesday



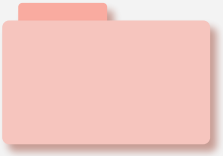
01.

Neptune is the fourth-largest planet in the Solar System



02.

Jupiter is a gas giant and the biggest planet in the Solar System



03.

Venus has a beautiful name and is the second planet from the Sun

Photo 01

Photo 02

Photo 01.

Venus has a beautiful name and is the second planet from the Sun. It's terribly hot—even hotter than Mercury

Photo 02.

Mercury is the closest planet to the Sun and the smallest one in the Solar System—it's only a bit larger than the Moon

History

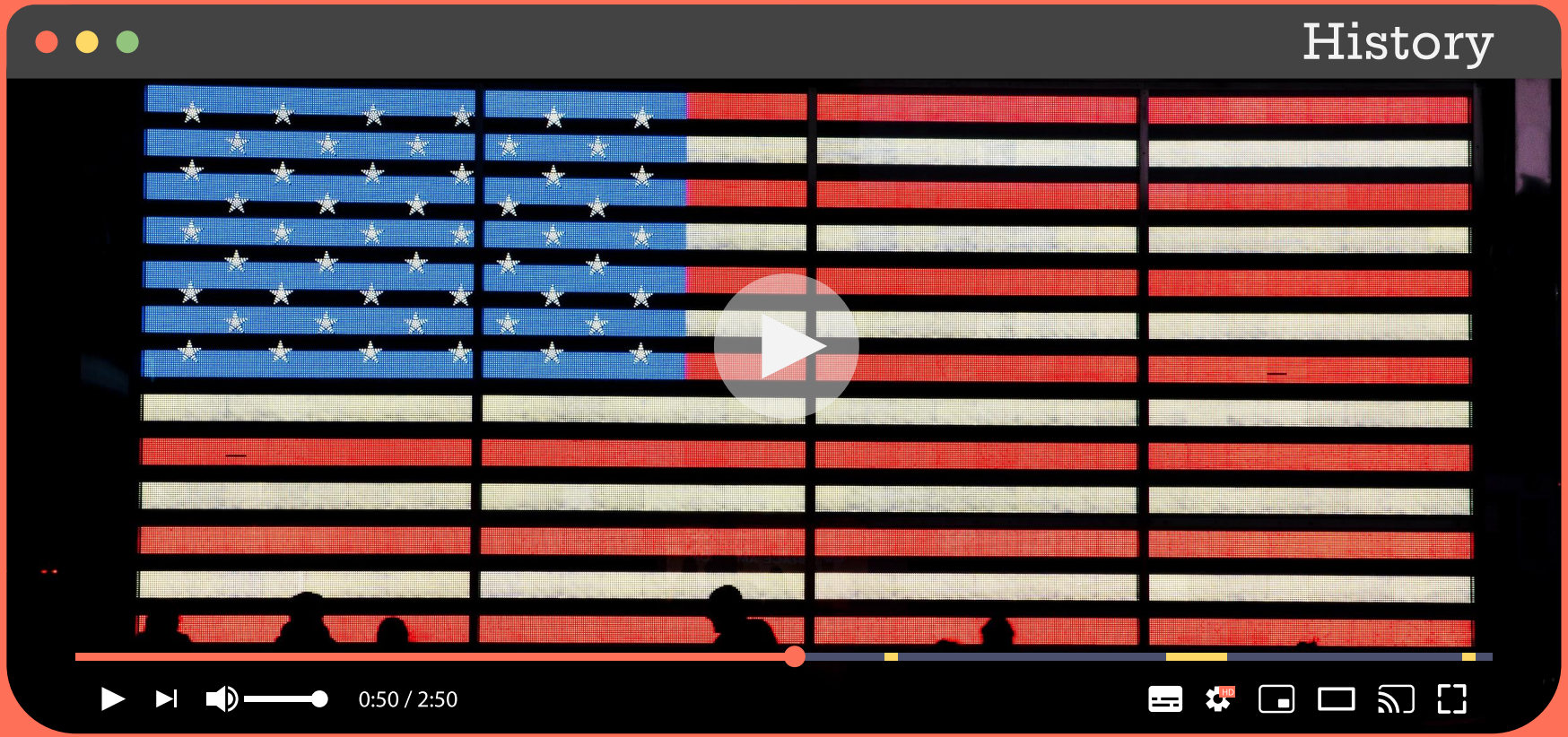
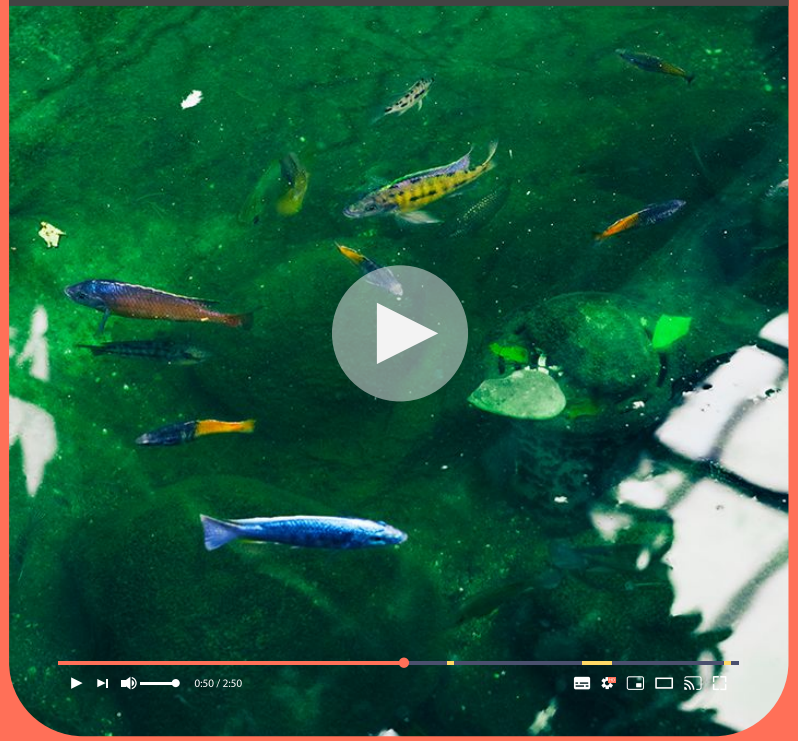
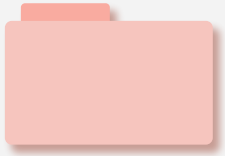


Photo 01

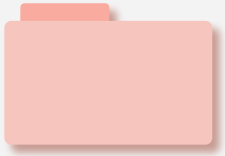
Photo 02



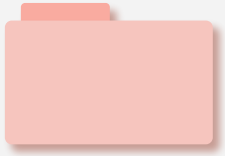
Wednesday



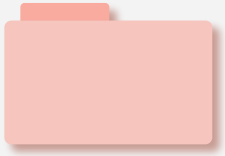
Despite being red, Mars is actually a cold place



Mercury is the smallest planet in the Solar System

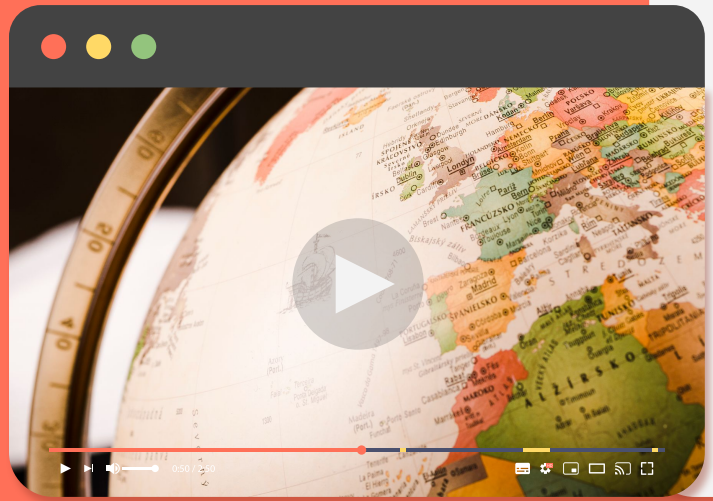


Venus has a beautiful name, but it's very hot



Neptune is the farthest planet from the Sun

Geography

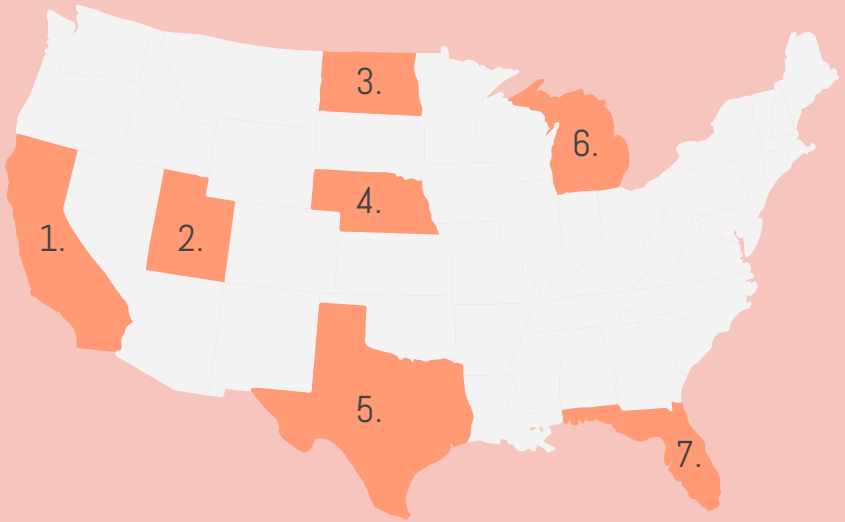


☑	Venus has a beautiful name
☒	Despite being yellow, Mars is actually a cold place
☐	Mercury was named after a Roman god
☐	Saturn is composed mostly of hydrogen and helium
☐	Neptune is the farthest planet from the Sun
☐	Jupiter is a gas giant and the smallest planet

Neptune is the farthest planet from the Sun:

1.
2.
3.
4.
5.
6.
7.

Geography

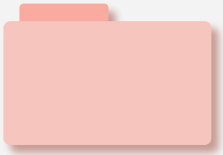


Name 5 countries

It's a gas giant and the biggest planet in the Solar System, which planet is it?

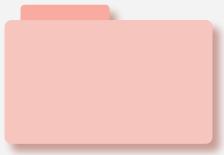
It's also a gas giant, the sixth planet from the Sun and has rings around it, what's its name?

Thursday



Exercises

Mercury is the closest planet to the Sun and also the smallest one in the Solar System

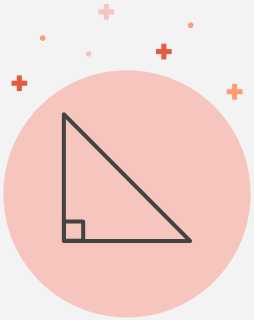


Theorems

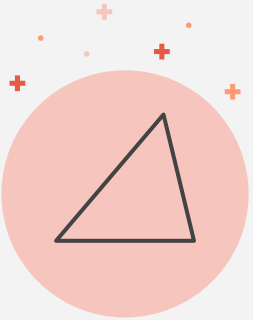
Venus has a beautiful name and is the second planet from the Sun. It's terribly hot

Maths

01. _____ is the fourth-largest planet in the Solar System



02. _____ is the biggest planet in the Solar System

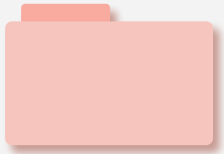


03. _____ is the farthest planet from the Sun

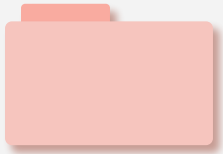
$$a^2=b^2+c^2$$

Mercury is the closest planet to the Sun

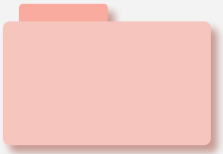
Friday



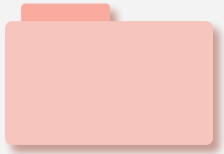
Despite being red, Mars is actually a cold place



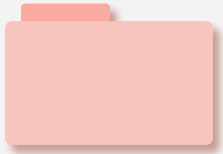
Mercury is the smallest planet in the Solar System



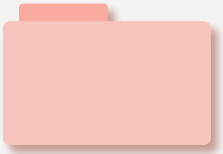
Saturn is composed of hydrogen and helium



Venus has a beautiful name, but it's very hot



Neptune is the farthest planet from the Sun



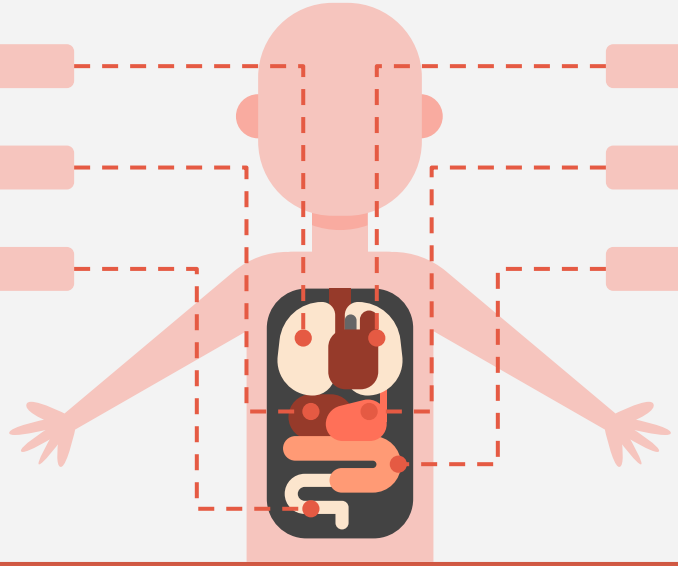
Jupiter is the biggest planet in the Solar System

Biology

a.

b.

c.



d.

e.

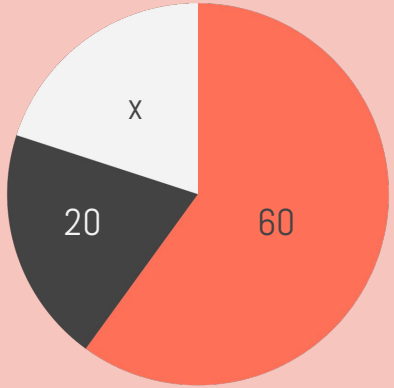
f.

a. Is Jupiter a gas giant and the biggest planet in the Solar System?

b. Despite being red, is Mars actually a cold place full of iron oxide dust?

c. To modify this graph, click on it, follow the link, change the data and paste the resulting graph here

Maths



Maths

25,120,356 cm^2

Insert the solution here

62,152 cm

Insert the solution here

872,532 cm

Insert the solution here

580,346 cm^2

Insert the solution here

Maths

$\frac{1}{2} + \frac{1}{2} = ?$

Insert the solution here

$\frac{1}{4} + \frac{1}{4} = ?$

Insert the solution here

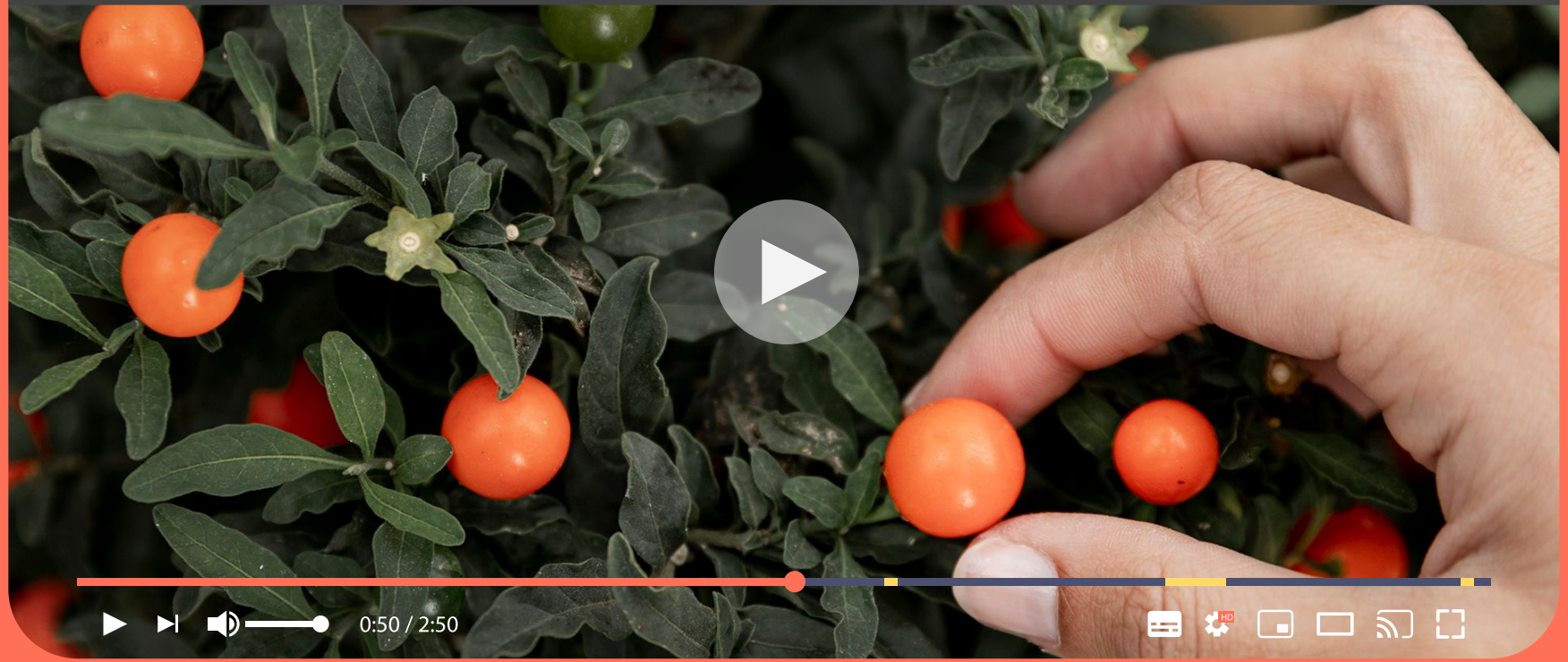
$\frac{1}{2} - \frac{1}{4} = ?$

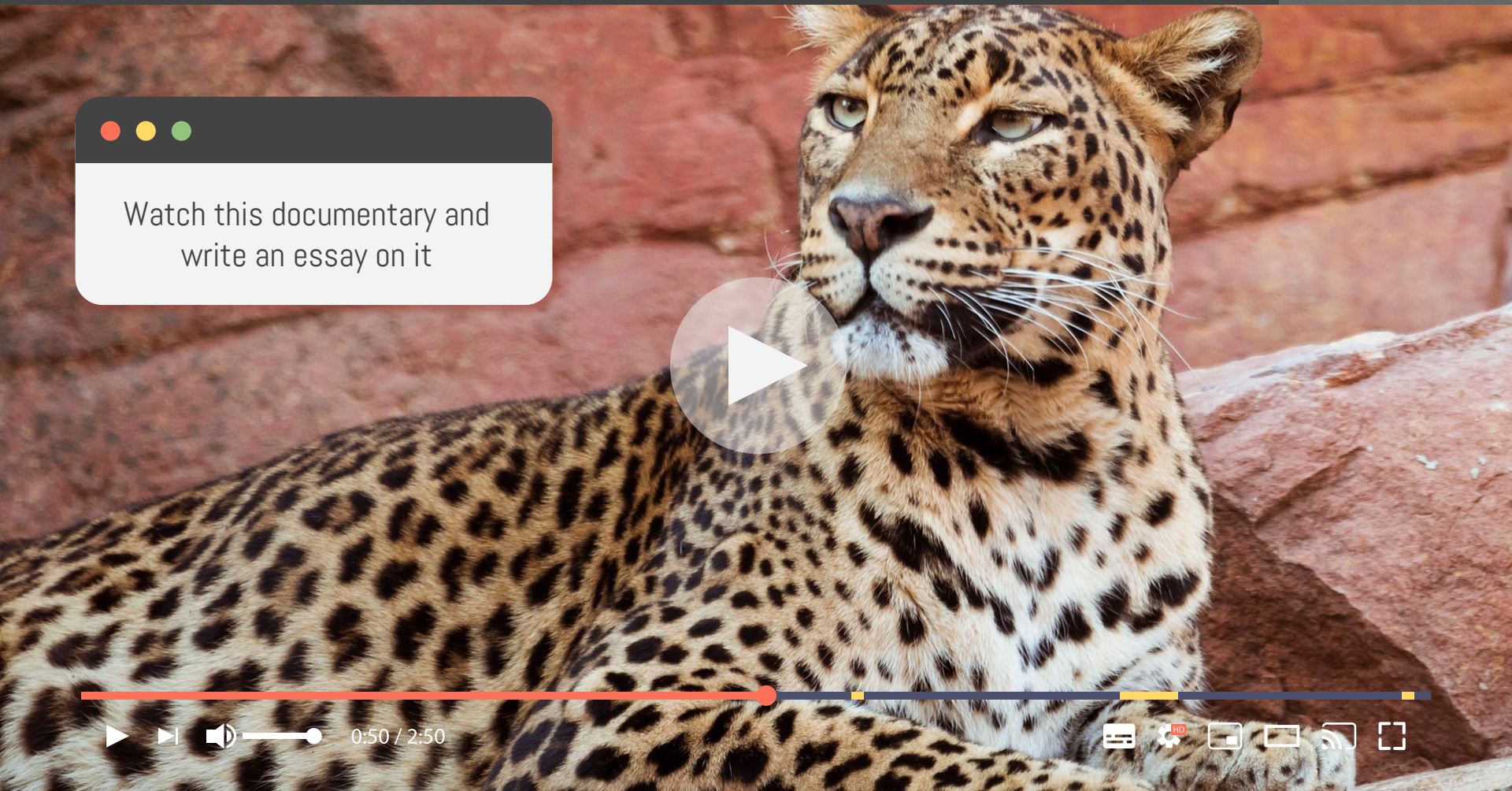
Insert the solution here

Natural Science

- [x] Venus has a beautiful name
- [ ] Despite being yellow, Mars is actually a cold place
- [ ] Mercury was named after a Roman god
- [ ] Saturn is composed mostly of hydrogen and helium
- [ ] Neptune is the farthest planet from the Sun
- [ ] Jupiter is a gas giant and the smallest planet

Natural Science




Watch this documentary and
write an essay on it
0:50 / 2:50

Monday

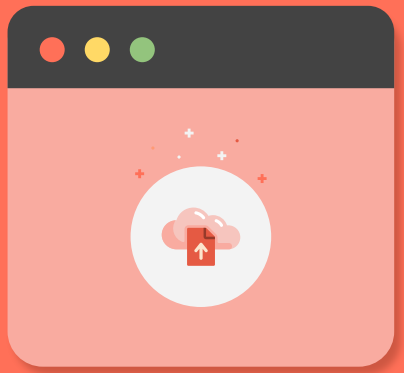
Tuesday

Wednesday

Thursday

Friday

Homework

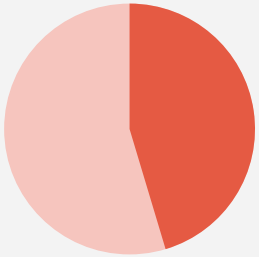


Homework Upload

1. Jupiter is the biggest planet in the Solar System
2. Despite being red, Mars is actually a cold place
3. Mercury is the smallest planet in the Solar System

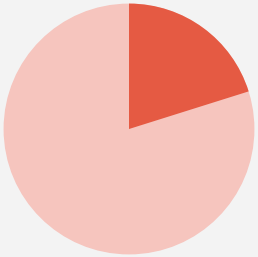
Subject - A

45%



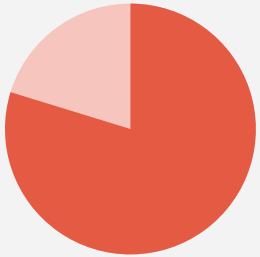
Completed chapters

15%



Homework

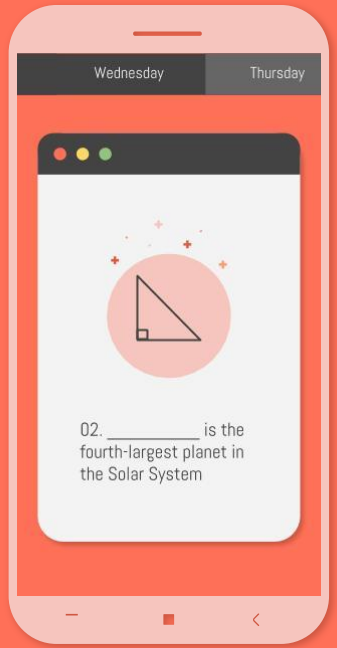
80%



Average grade

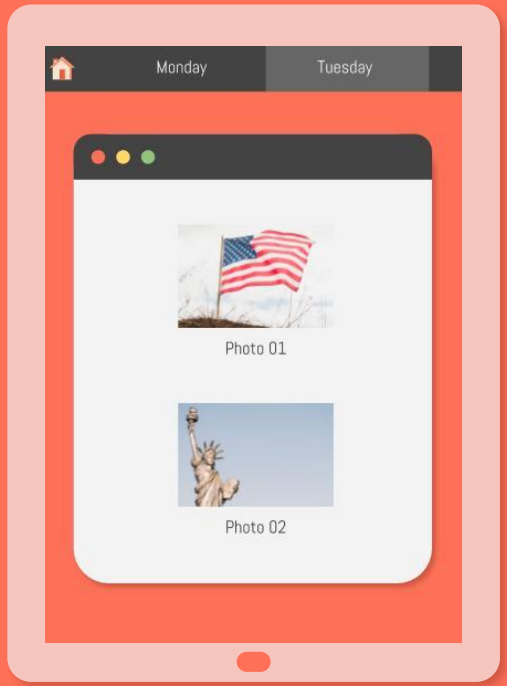
Desktop Software

You can change the image on the screen with your own work. Just delete this one, add yours and send it to the back



Mobile Web

You can change the image on the screen with your own work. Just delete this one, add yours and send it to the back



Tablet App

You can change the image on the screen with your own work. Just delete this one, add yours and send it to the back

Your Teacher

Jenna Doe

Mercury is the closest planet to the Sun and the smallest one in the Solar System

Thanks!

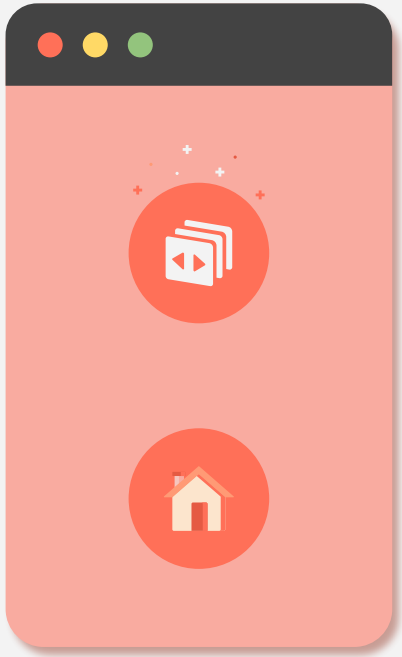
Do you have any questions?

youremail@freepik.com
+91 620 421 838
yourcompany.com

Alternative Resources

Photos

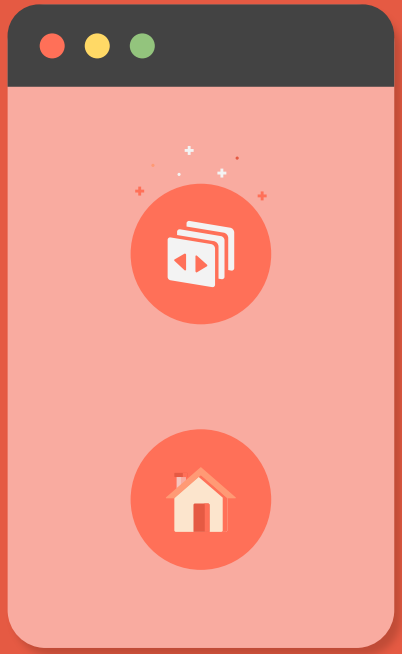
- Close-up of a vintage old globe
- Usa flag on dark marble
- Hanging bridge in rainforest at costa rica
- Daisy flowers near the river
- Teen schoolgirl in glasses reading book
- Statue of liberty
- Gorilla
- Meerkat
- Animal habitat
- Flamingos
- Green plant with orange berries
- Top view arrangement with red and yellow flowers



Alternative Resources

Photos

- Close-up of a vintage old globe
- Usa flag on dark marble
- Hanging bridge in rainforest at costa rica
- Daisy flowers near the river
- Teen schoolgirl in glasses reading book
- Statue of liberty
- Gorilla
- Meerkat
- Animal habitat
- Flamingos
- Green plant with orange berries
- Top view arrangement with red and yellow flowers



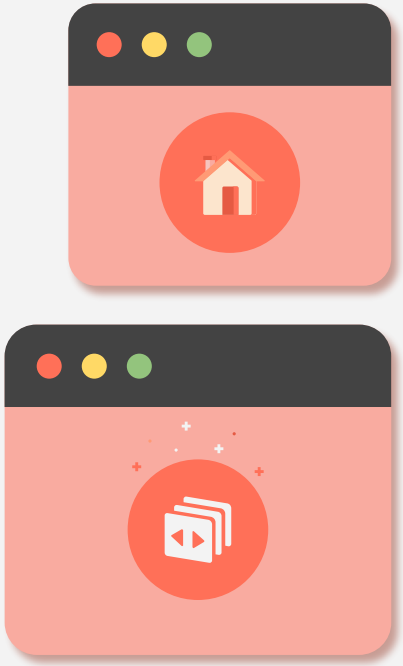
Photos

- Idyllic river near costa rican rainforest
- Daisy flowers near the river
- Glowing usa flag
- Lizard
- Fishes in a pond
- American flag on hill
- Statue of liberty
- Detail of a vintage globe
- Honeycomb with bees
- Green plant with orange berries
- Leopard
- Lady looking at camera and holding book near window

Vector

- Nice infographic of the human body in flat design

Resources



Instructions for use

In order to use this template, you must credit **Slidesgo** by keeping the **Thanks** slide.

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 Slidesgo School:

https://slidesgo.com/faqs and https://slidesgo.com/slidesgo-school

Fonts & colors used

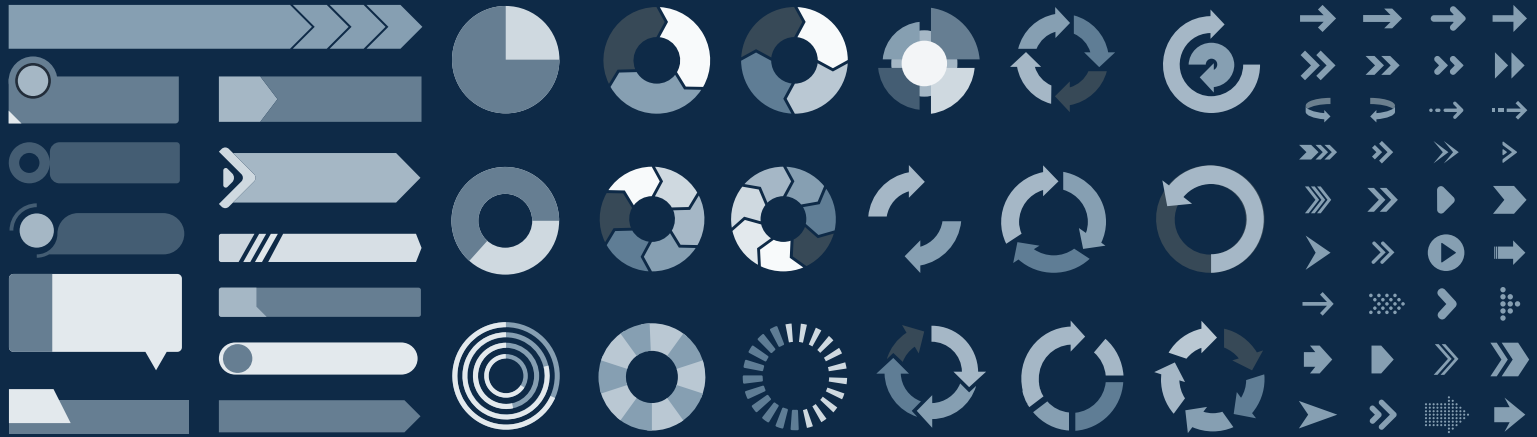
This presentation has been made using the following fonts:

Hepta Slab
(https://fonts.google.com/specimen/Hepta+Slab)

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

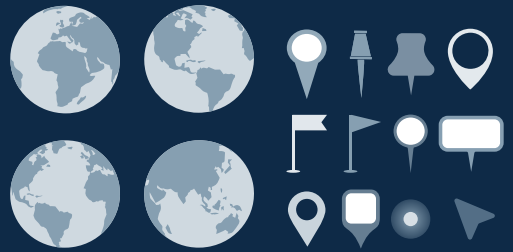
Use our editable graphic resources...

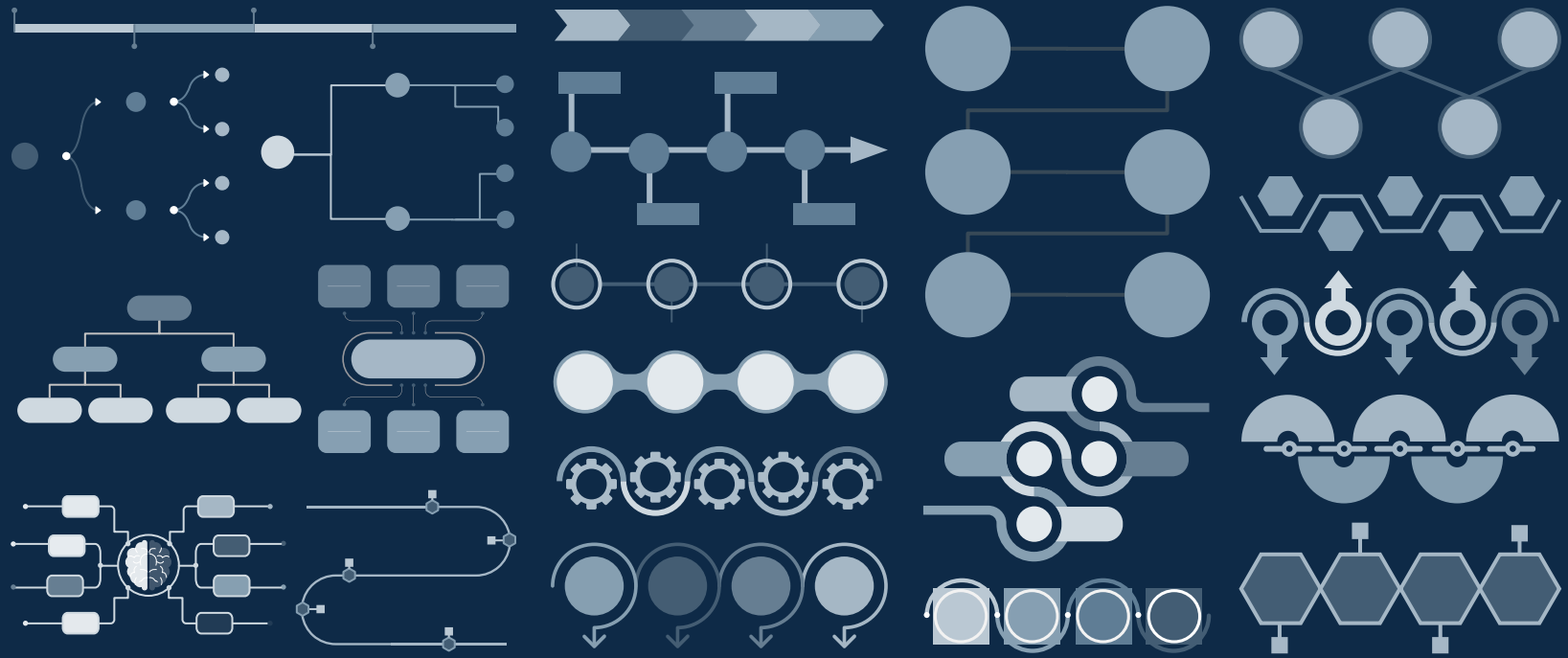
You can easily resize these resources without losing quality. To change the color, just ungroup the resource and click on the object you want to change. Then, click on the paint bucket and select the color you want. Group the resource again when you're done.

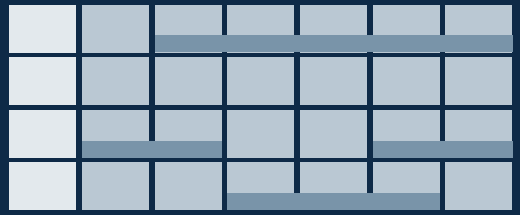


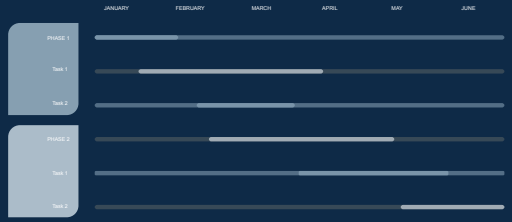


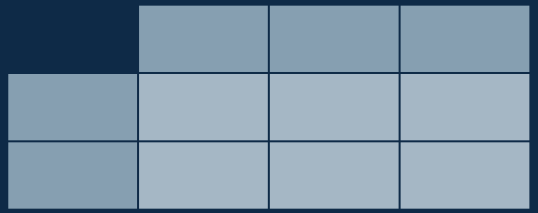


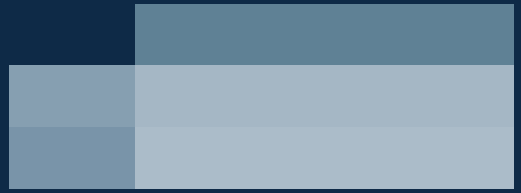



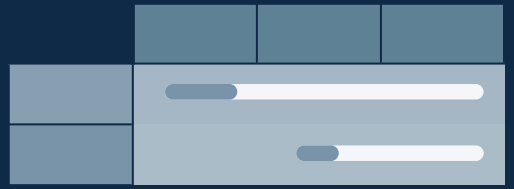


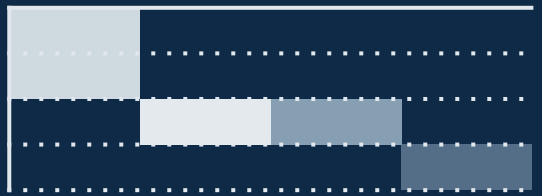


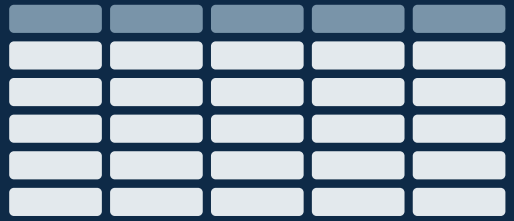

JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
PHASE 1
Task 1
Task 2
PHASE 2
Task 1
Task 2

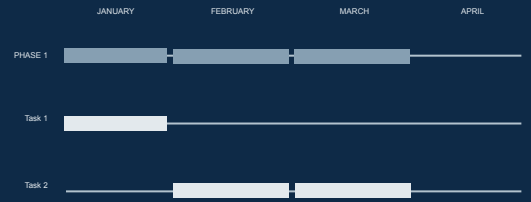


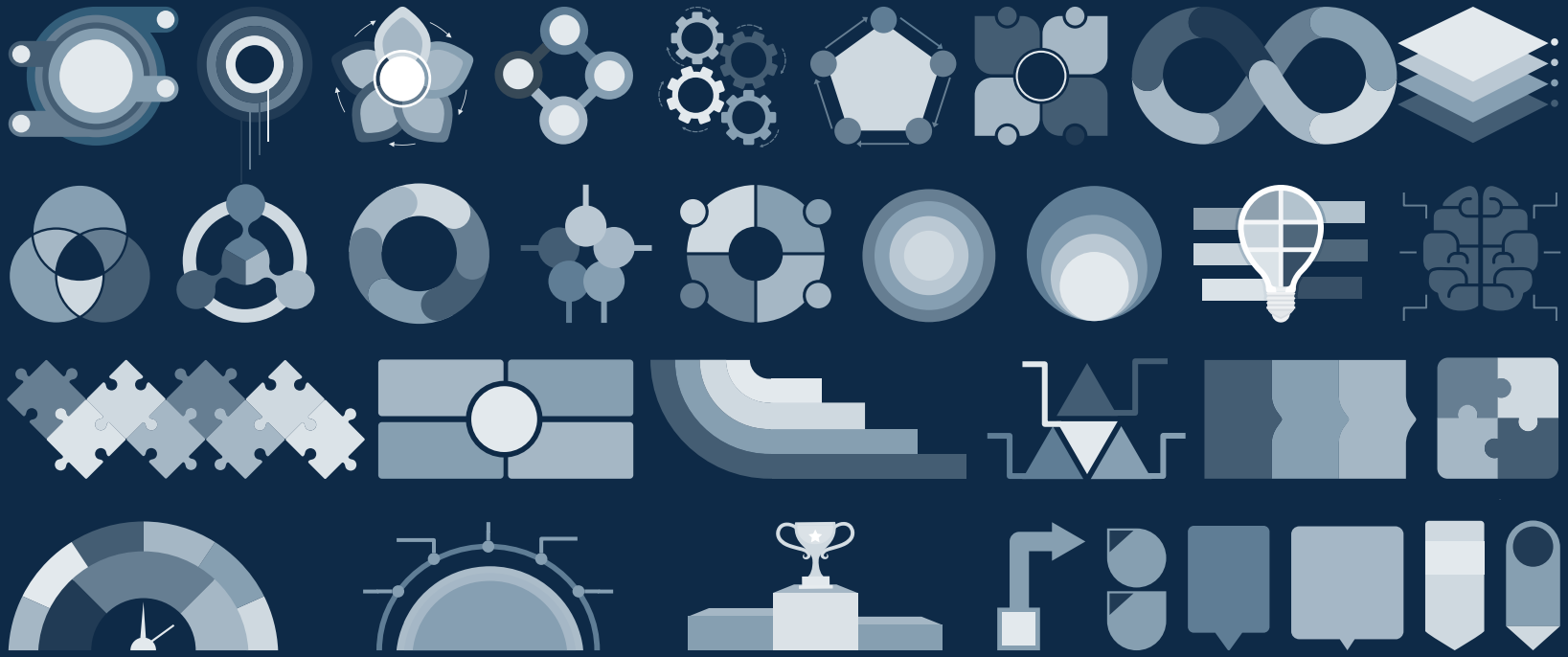


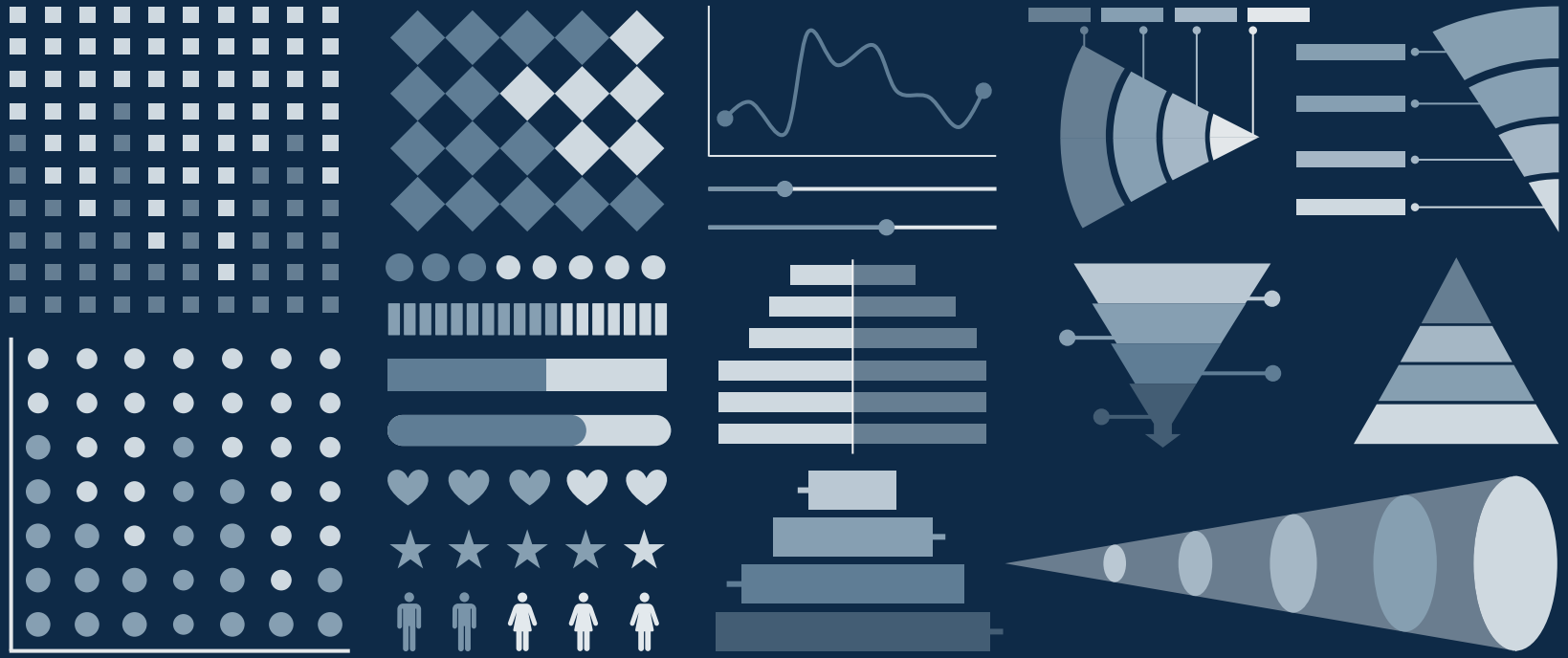







JANUARY
FEBRUARY
MARCH
APRIL
PHASE 1
Task 1
Task 2





...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