Welcome Everyone!

We will wait for others to join in!

We Will start in 10

KNOW ABOUT ME:

Python Basics

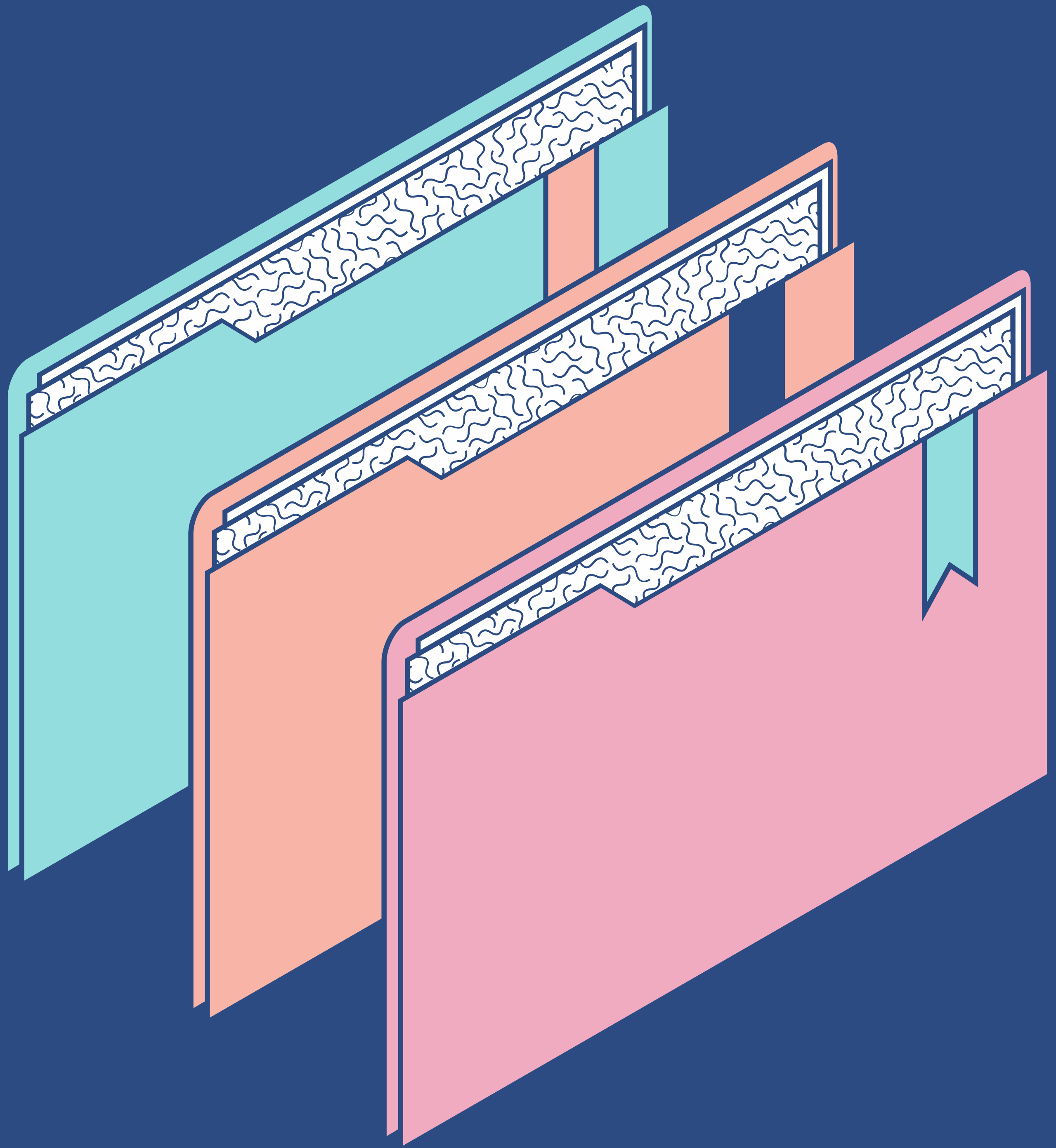
with, Aaryan Kapur

WEEK 5

Nikhil Kumar

TOP PERFORMER

Week 4



Functions in Python

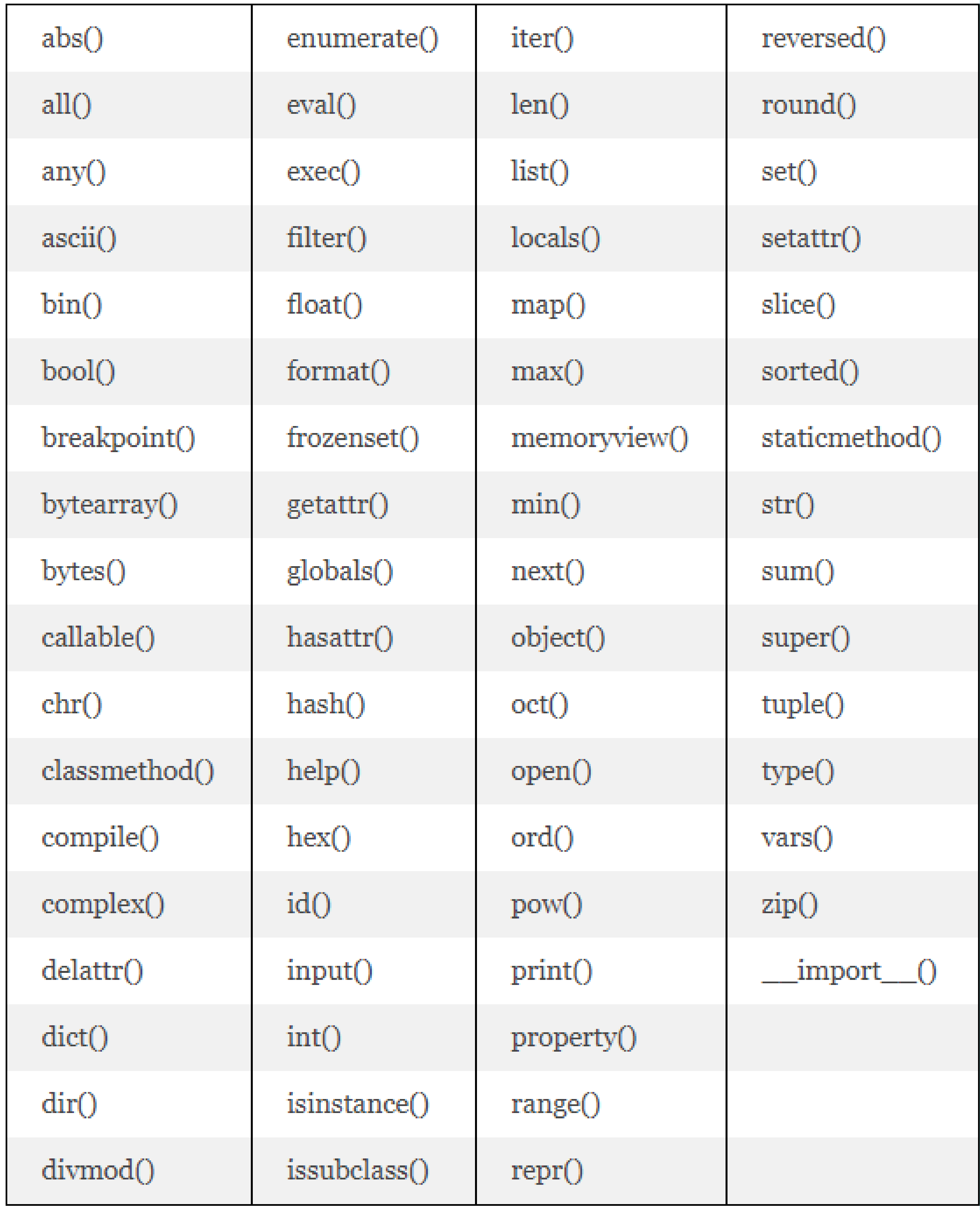
There are several functions in python and allows us to create more as well!

Functions allow us to create code blocks that can be used repeatedly and recursively!

In-Built functions in Python

Python gives us several In-Built functions

Learn more at w3schools

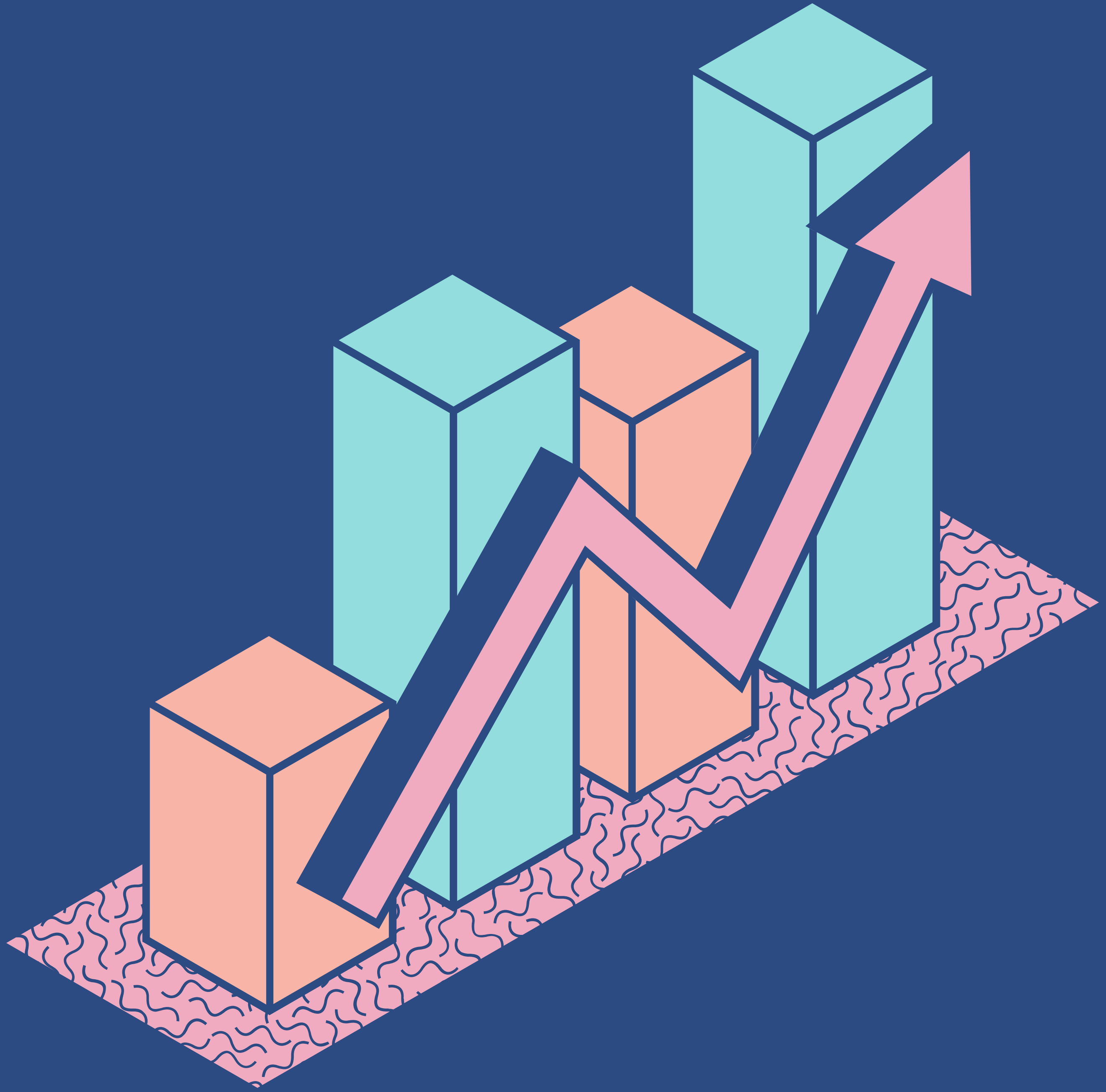


abs()	enumerate()	iter()	reversed()
all()	eval()	len()	round()
any()	exec()	list()	set()
ascii()	filter()	locals()	setattr()
bin()	float()	map()	slice()
bool()	format()	max()	sorted()
breakpoint()	frozenset()	memoryview()	staticmethod()
bytearray()	getattr()	min()	str()
bytes()	globals()	next()	sum()
callable()	hasattr()	object()	super()
chr()	hash()	oct()	tuple()
classmethod()	help()	open()	type()
compile()	hex()	ord()	vars()
complex()	id()	pow()	zip()
delattr()	input()	print()	__import__()
dict()	int()	property()	
dir()	isinstance()	range()	
divmod()	issubclass()	repr()	

Important Built-in Functions in Python

PRINT	**ABSOLUTE**	**ROUND**	**MINIMUM**	**MAXIMUM**
print()	**abs()**	**round()**	**min()**	**max()**
Used to print	Find absolute value	Round off number	Find Minimum	Find Maximum
SORTED	**SUM**	**LENGTH**	**TYPE**	**ITER**
sorted()	**sum()**	**len()**	**type()**	**iter()**
Sort List	Sum of all items	Find Length	Type of variable	Make iterable

Defining Functions

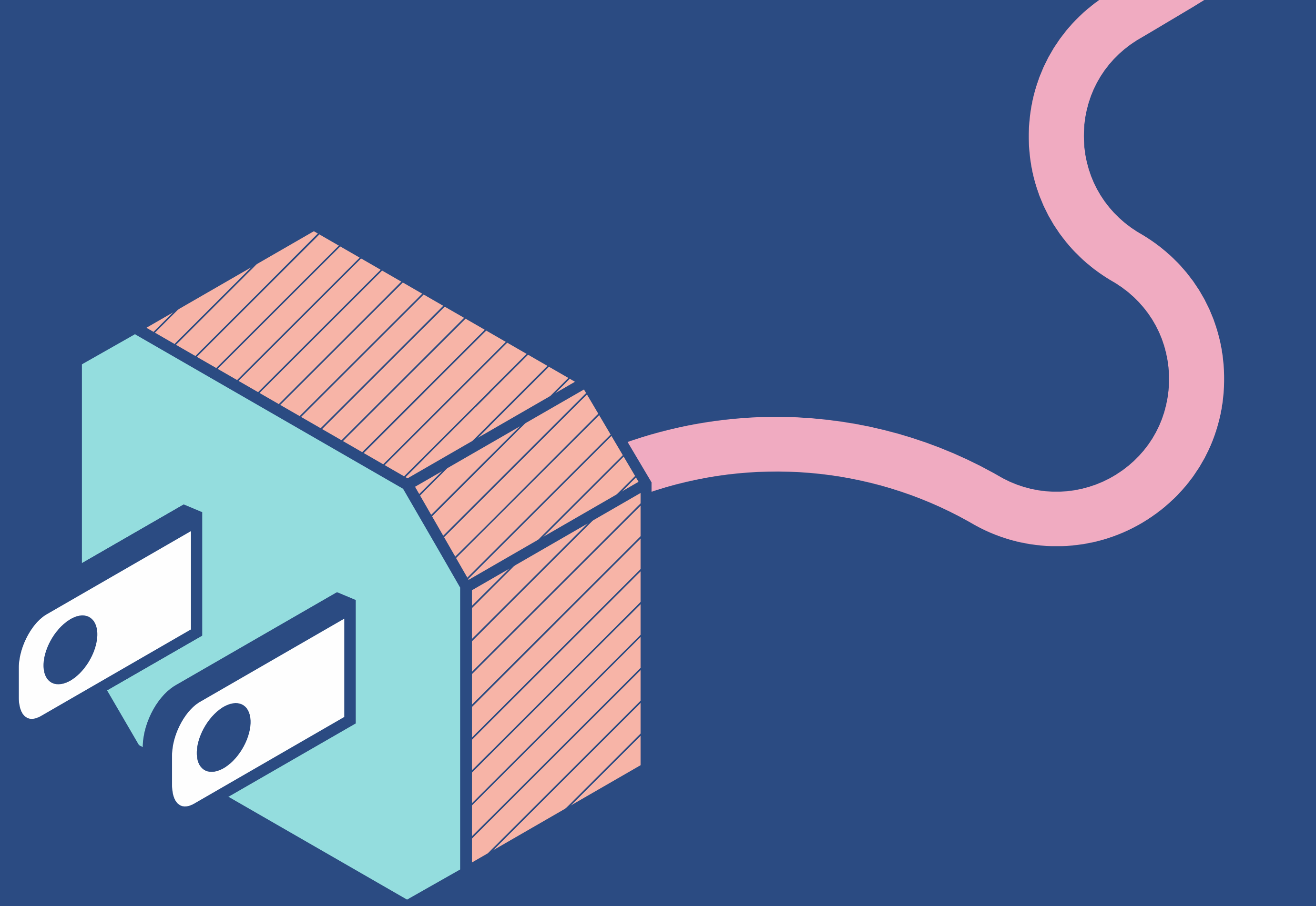


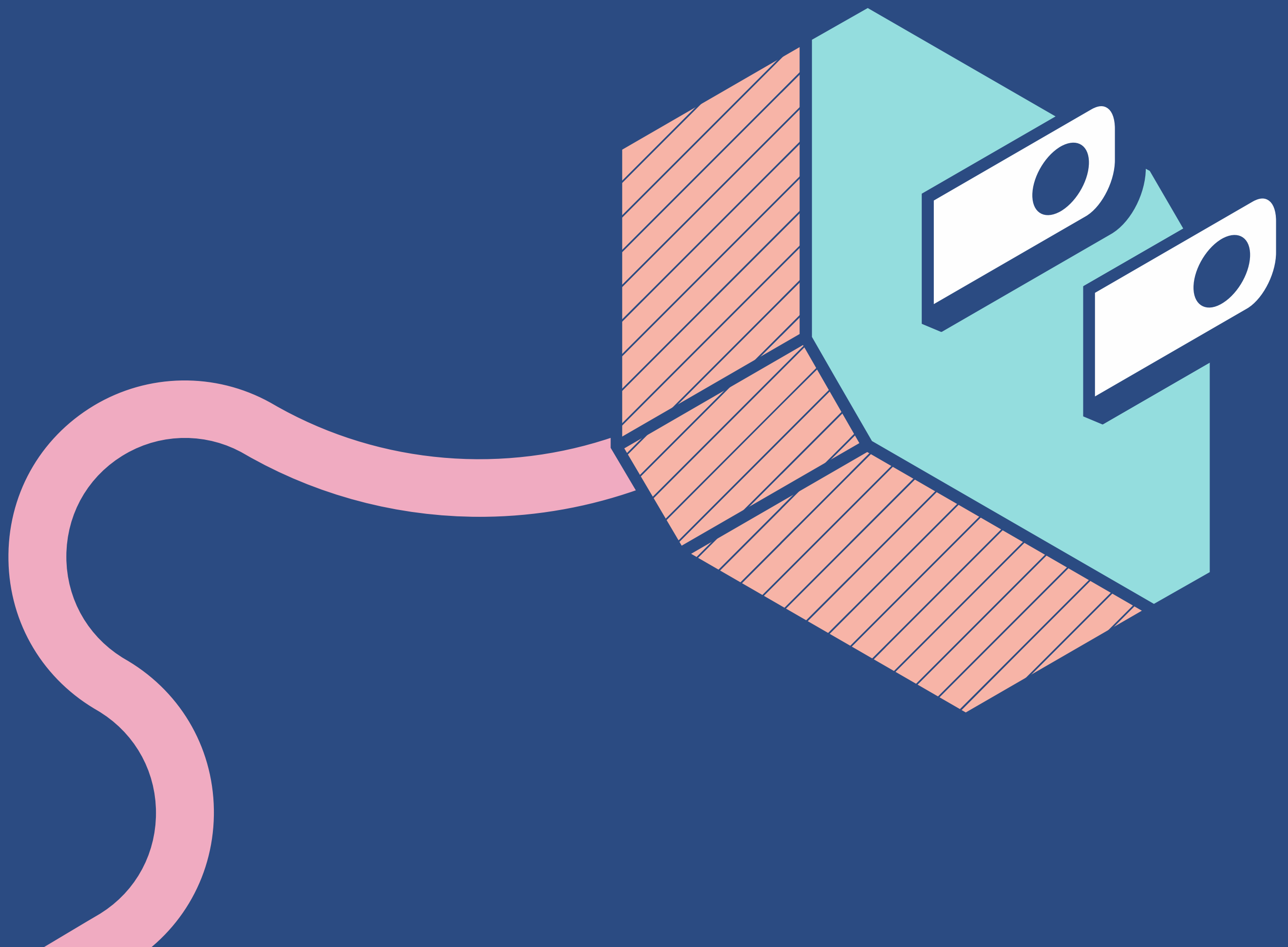
```
def FunctionName():
      CODE........


def FunctionName(A,R,G,U,M,E,N,T,S):
      CODE........
```

Return in Functions

Return allows us to return an object that can be used!

```
def FunctionName(a):
    return(a)
```





Print in Functions

Print allows us to simply return as a value!

```
def FunctionName(a):
    print(a)
```

Types of Arguments in Functions

Positional Arguments

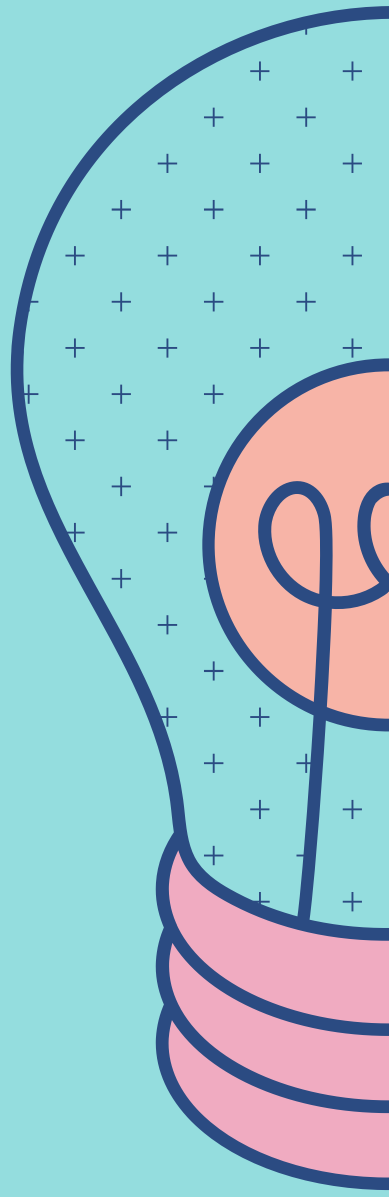
```
def fungt(a,b,c):
    return(a,b,c)
fungt(3,2,33)
```

Keyword Arguments

```
def fungt(a,b,c):
    return(a,b,c)
fungt(3,c= 33,b= 2)
```

Default Arguments

```
def fungt(a,b,c=33):
    return(a,b,c)
fungt(3,2)
```



Taking variable number of arguments

xargs

```
def vFunction(*numbers):
        print(sum(numbers))
vFunction(1,2,3)
```

xxargs

```
def vFunction(**numbers):
        print(numbers)
vFunction(one = "One", two = "Two", three = "Three")
```

Scope of variable

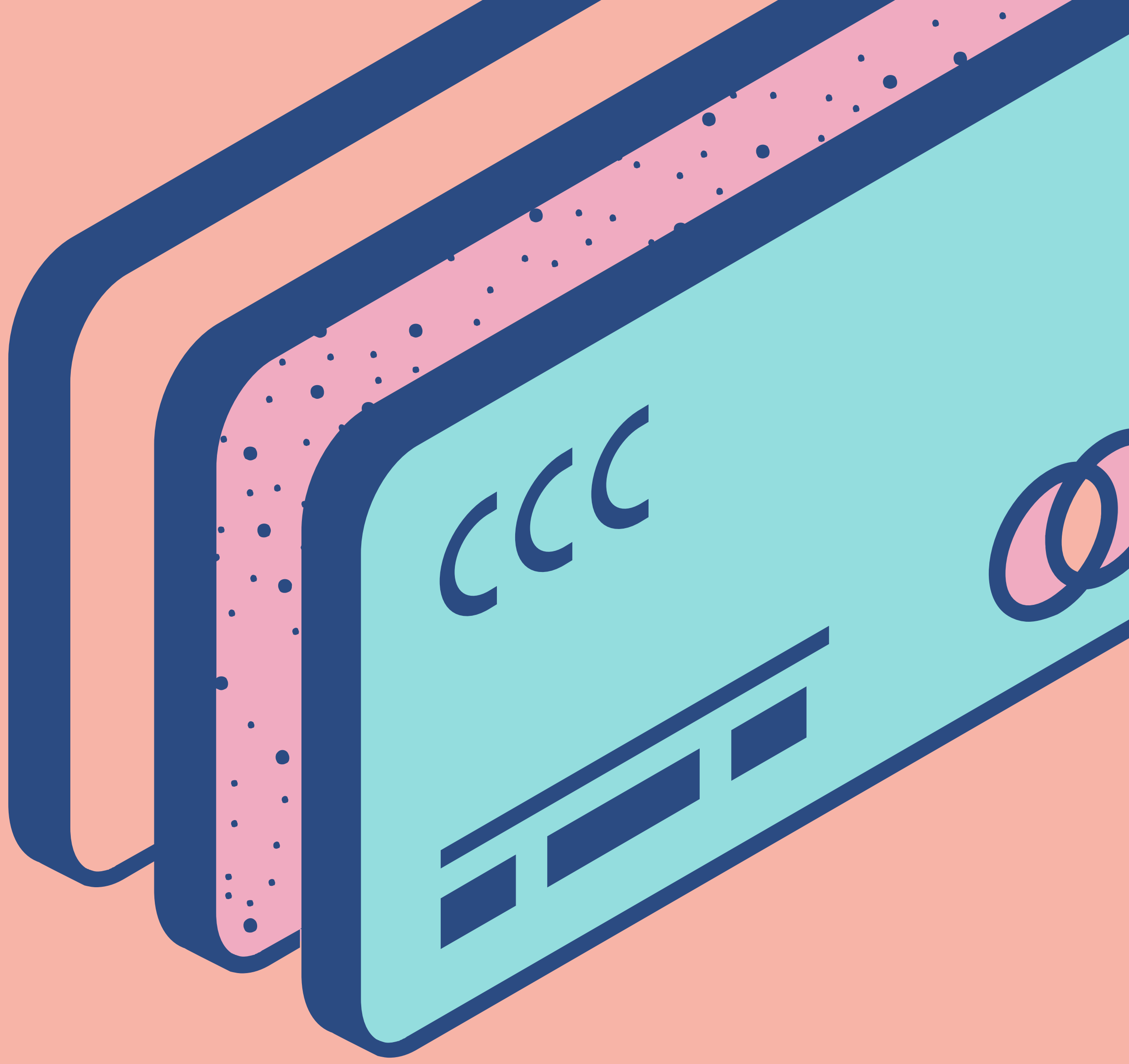
Variables have a scope that governs access to the variable!

```
a = 10
def functionHello():
  a = 4
  return(a)
```

```
print(a)    print(functionHello())
10          4
```

How can the same Variable have 2 Values?

Scopes of Variables



Global Variables

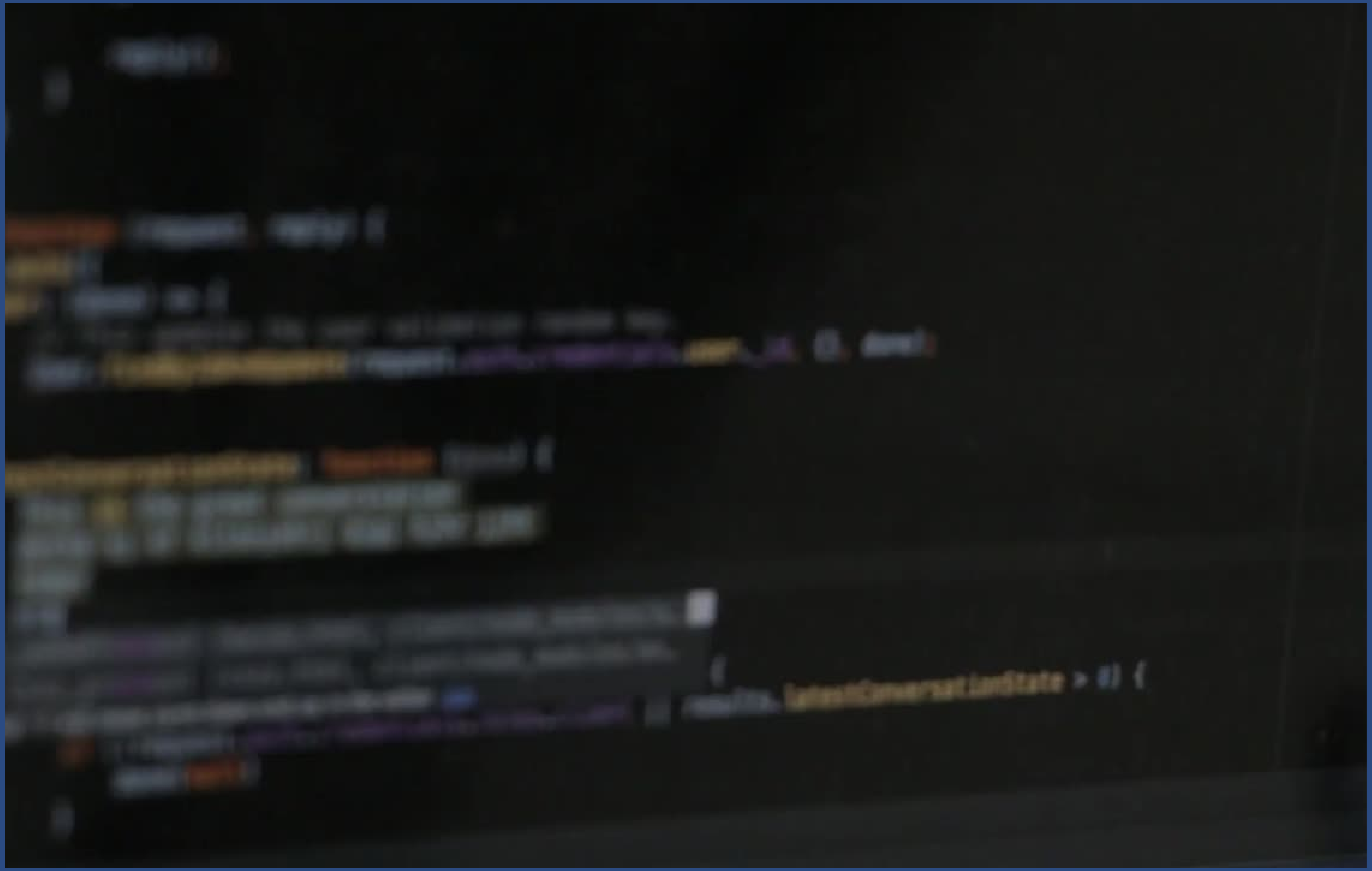
Access to these Variables are globally granted to all functions etc.

```
a = 10
def functionHello():
   global a
   a = 4
  return(a)
```

Local Variables

Access to these Variables are globally granted to only specific functions etc.

```
a = 10
def functionHello():
  a = 4
  return(a)
```



```
[illegible] latestConversationState > 0) {
```

</> CODE SCREEN

Time in python

```
import time
```

```
time.time()          1621763583.2455165
```

Tells us time in seconds since January 1, 1970, 00:00:00 UTC

```
time.ctime()    Sun May 23 09:53:21 2021
```

Tells us the current time, with day and date

```
time.sleep(x)  -waiting for x seconds-
```

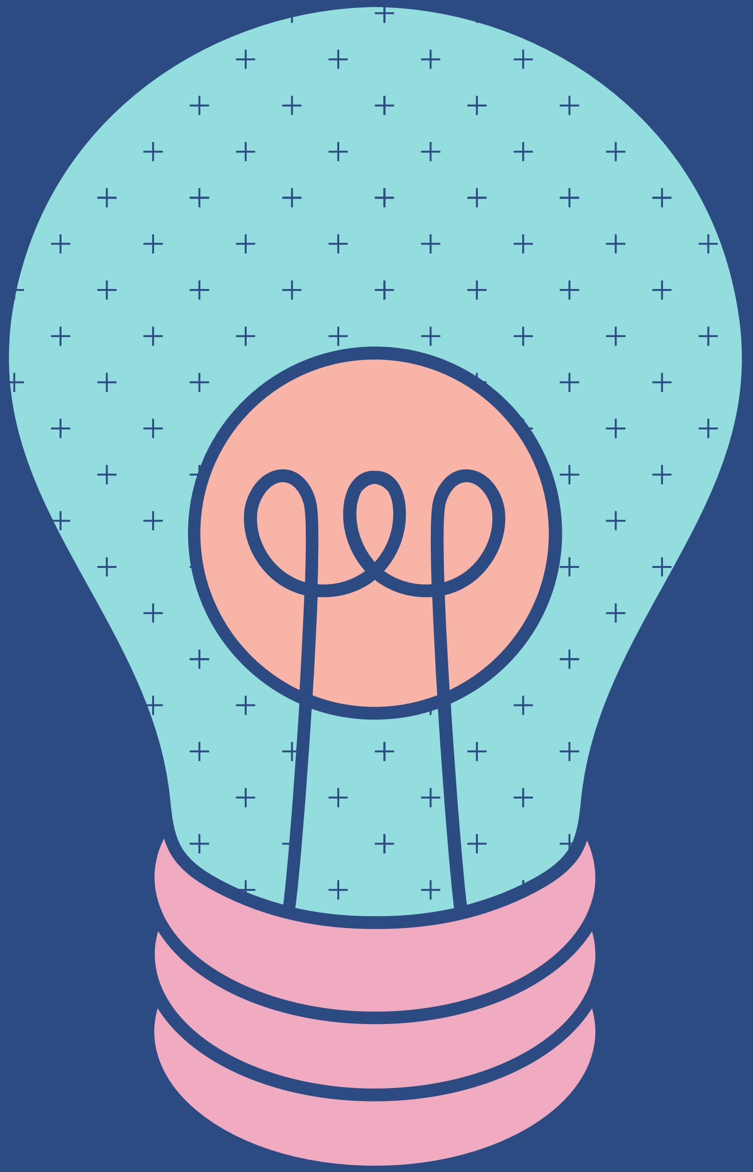
Adds a delay of x seconds during code execution

Datetime in python

```
import datetime
```

The date contains year, month, day, hour, minute, second, and microsecond.

```
datetime.datetime.now()
2021-05-23 10:01:00.978506
datetime.datetime.now().year
2021
datetime.datetime.now().month
5
datetime.datetime.now().hour
10
```



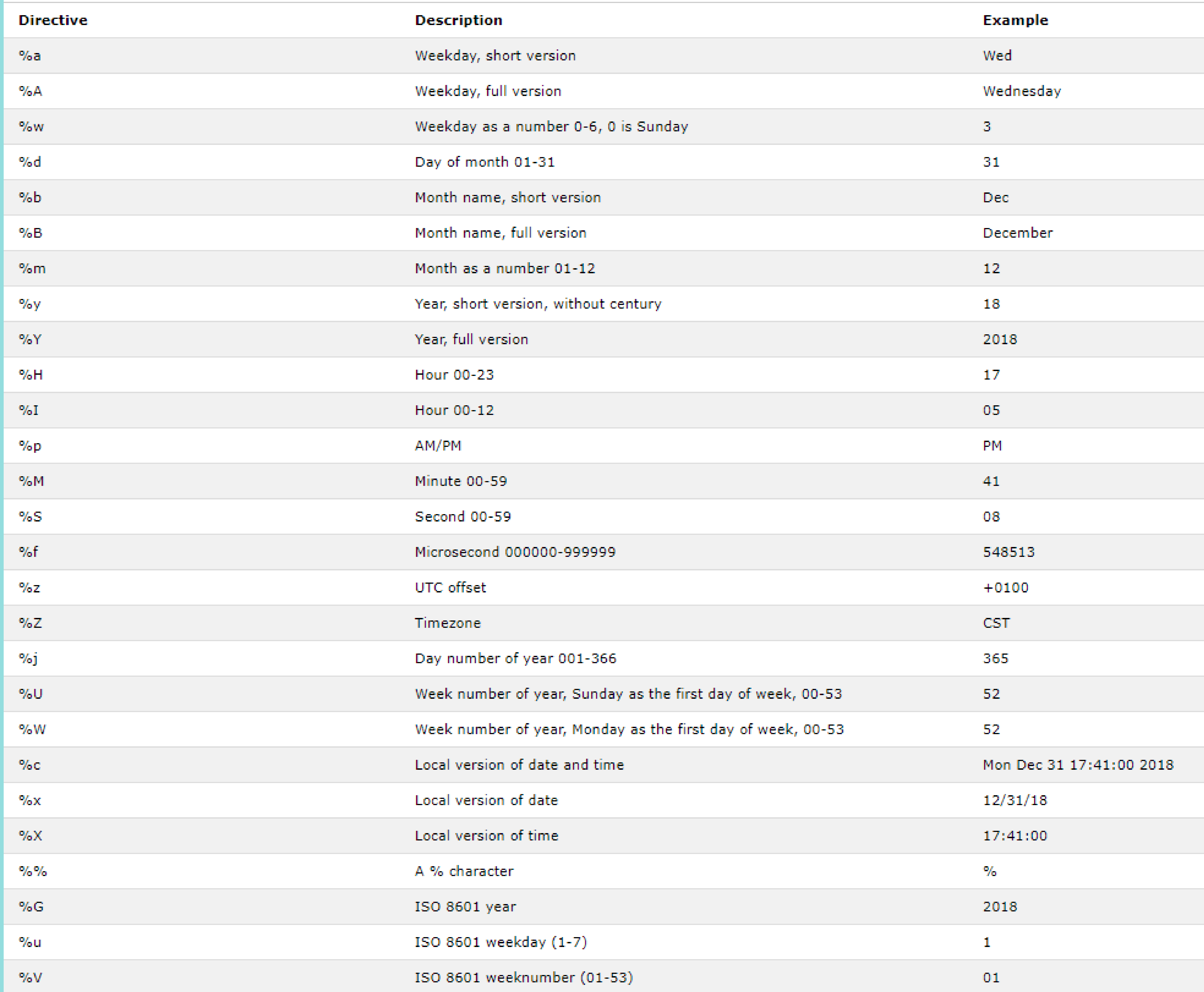
strftime()

String Format Time

We can present the date in a string formatted readable manner!

```
datetime.datetime.now().strftime("%B")
```

Key for strftime()



Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%
%G	ISO 8601 year	2018
%u	ISO 8601 weekday (1-7)	1
%V	ISO 8601 weeknumber (01-53)	01

timedelta

```
from datetime import timedelta
```

To add time delay to the initial/start time!

```
from datetime import timedelta
initial = datetime.now()
final = initial + timedelta(days = 2)
```

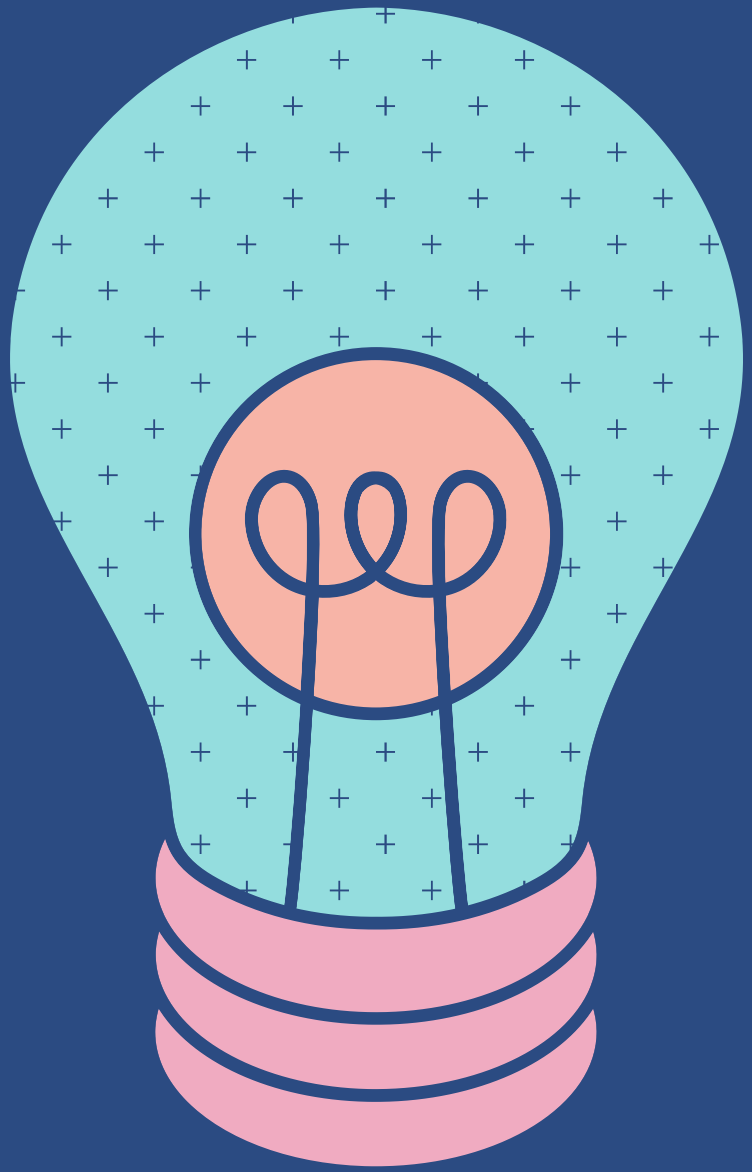
```
str(initial)
2021-05-23 10:16:49.834909
str(final)
2021-05-25 10:16:49.834909
```

Random in python

```
import random
```

The date contains year, month, day, hour, minute, second, and microsecond.

```
random.randint(0,9)
5
random.random()
 0.82793309921671
```



```
random.randrange(1, 10, 2)
5
a = [12,23,45,67,65,43]
random.choice(a)
23
```

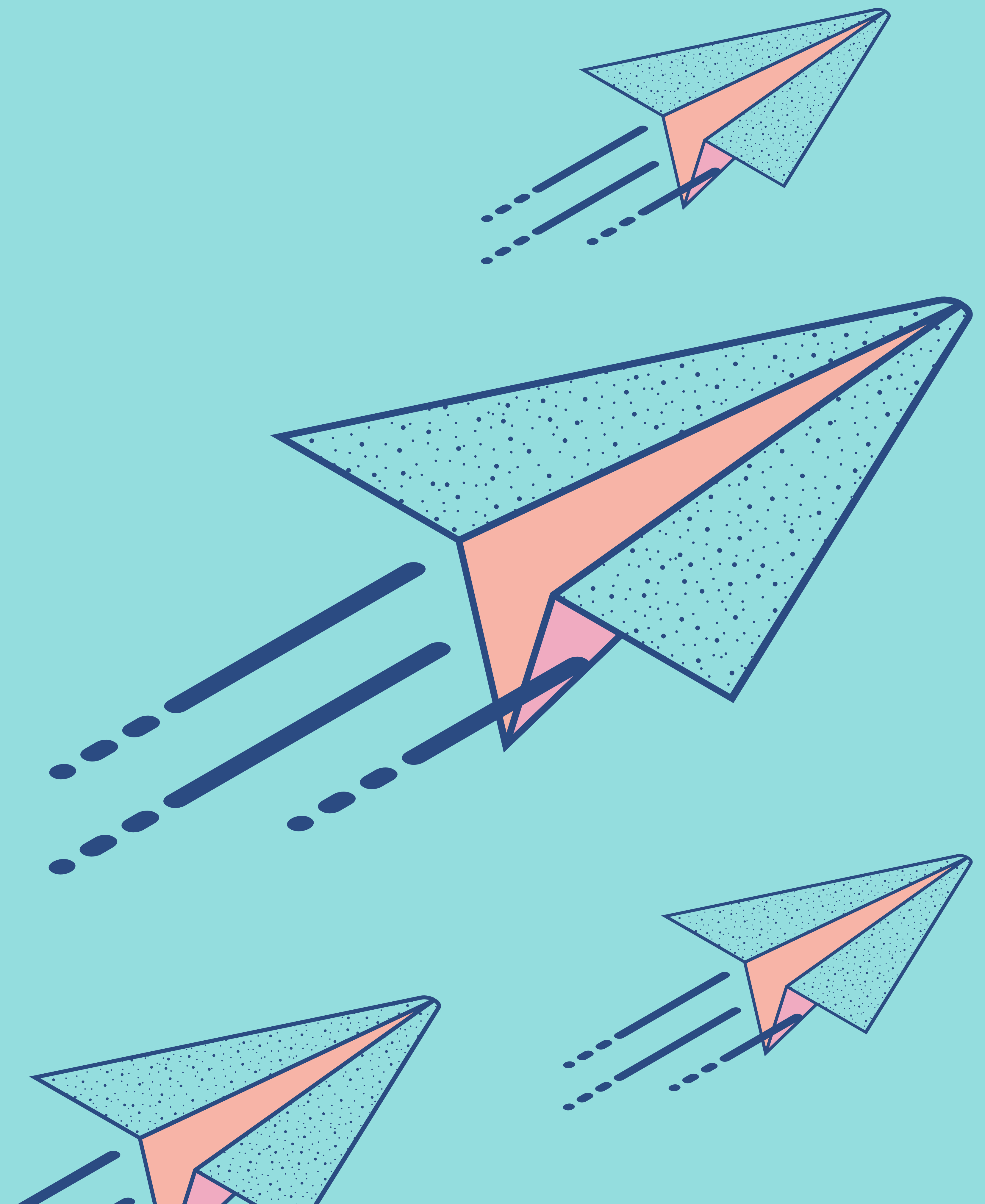
```
a = [12,23,45,67,65,43]
random.shuffle(a)
a
[67, 43, 45, 65, 23, 12]5
```

Send email

```
import smtplib
```

```
import smtplib

content = "Hello "+ name+"!!!! \nHow are you!"
mail = smtplib.SMTP('smtp.gmail.com', 587)
mail.ehlo()
mail.starttls()
mail.login("testidpy61@gmail.com", 'testidpy61@123')
mail.sendmail('testidpy61@gmail.com', 'aaryankapur1309@gmail.com', content)
mail.close()
```

Do you have any questions?



Join Here!

Thank you!