<https://docs.python.org/3/reference/>

<https://docs.python.org/3/library/index.html#library-index> Python standard library

<https://www.w3schools.com/python/python_reference.asp>

Build a simple clock

import time

from datetime import datetime

now = datetime.now()

current\_time = now.strftime("%H:%M:%S")

print("Current Time =", current\_time)

time\_split = current\_time.split(":")

print(time\_split)

hour = int(time\_split[0])

minute = int(time\_split[1])

second = int(time\_split[2])

def display():

print(hour, ':', minute, ':', second)

def increment():

global hour

global minute

global second

second=second+1

if second==60:

minute=minute+1

second=0

if minute==60:

hour=hour+1

minute=0

if hour==24:

hour=0

print('\n')

while True:

time.sleep(1)

increment()

display()

# Armstrong number.

## The problem

Check whether a number is an Armstrong number.

## Armstrong number

Armstrong is a special number.

A number is an Armstrong Number or narcissistic number if it is equal to the sum of its own digits raised to the power of the number of digits.

Think about the number 371. The total number of digits is 3. Now, for each digit, put the power of 3 and then add them. It will be:

33 + 73 + 13

= 27 + 343 + 1

= 371

Hence, 371 is an Armstrong number.

Similarly, 1634 is another Armstrong number, because the total number of digits is 4. Now, power each digit and sum them. You will get the Armstrong number.

= 14 + 64 + 34 + 44

= 1 + 1296 + 81 + 256

= 1634

## The solution

def check\_armstrong(num):

order = len(str(num))

sum = 0

# use temp to find each digit. Then power it

temp = num

while temp > 0:

digit = temp % 10

sum += digit \*\* order

temp //= 10

return num == sum

num = int(input('Enter your number: '))

if check\_armstrong(num):

print(num,"is an Armstrong number")

else:

print(num,"is not an Armstrong number")

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Explanation

Three things could be new for you.

First of all, to know the total number of digits in a number, we converted the number to a string. We used the str(num) to convert it to the string. And then we pass it to the len function.

This mean, len function will tell you how many characters are on that string.

Another thing we did is to store the input number in a temp variable. We had to do that because, while getting the digits, we are dividing the number by 10.

However, at the end, we need to compare the result of the while loop in the sum with the original number.

That's why, we set the num to the temp variable. And then, we divided the temp variable. In this way, the num variable remained the same.

Finally, return num == sum means if num is equal to the sum, it will return True. Otherwise, it will return False.

Hence, return num == sum is a shortcut method of:

result = False

if num == sum:

result = True

else:

result = False

return result

# 2.3: Password with Requirements [Premium]

## The Problem

Generate a password that has a minimum of one uppercase, one lowercase, one digit, and one special character

## Hint

This one is easier. The string module has two more things to get the uppercase and lower case characters...

import string

print('All letters')

print(string.ascii\_letters)

print('Al lowercase characters')

print(string.ascii\_lowercase)

print('All uppercase characters')

print(string.ascii\_uppercase)

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Solution

import random

import string

def randomPassword(size):

all\_chars = string.ascii\_letters + string.digits + string.punctuation

password = ""

password += random.choice(string.ascii\_lowercase)

password += random.choice(string.ascii\_uppercase)

password += random.choice(string.digits)

password += random.choice(string.punctuation)

for i in range(size-4):

password += random.choice(all\_chars)

return password

pass\_len = int(input("What would be the password length? "))

print ("First Random Password is:", randomPassword(pass\_len))

print ("Second Random Password is:", randomPassword(pass\_len))

print ("Third Random Password is:", randomPassword(pass\_len))

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Explanation

The all\_chars variable is the same as before. After that, we created a password variable with an empty string.

Then, we added random choice from lowercase. This way, we will make sure that at least one lowercase character is added in the password. Similarly, we added one uppercase, one digit, and one special character.

After that, we ran a for loop to select random characters from the all\_chars. This is similar to the previous problem. We just ran the for loop for (size-4) times, because we already added 4 characters in the password before the loop.

Make sense?

Keep going. Only a few more left. I am sure you can do it

# 12.1 Simple Calculator

## The task

Create a simple calculator. That will be able to take user input of two numbers and the operation the user wants to perform.

## Solution strategy

Create a bunch of functions to perform add, subtract, multiply, division or modulo.

Then take two numbers from the user and the operation he/she wants to perform. Either +,-,\*,/ or %.

Then call the appropriate function based on the operation.

Think for a few minutes and try it yourself first.

## The solution

def add(num1, num2):

return num1 + num2

def subtract(num1, num2):

return num1 - num2

def multiply(num1, num2):

return num1 \* num2

def divide(num1, num2):

return num1 / num2

def modulo(num1, num2):

return num1 % num2

# Take input from the user

num1 = int(input("Enter first number: "))

operation = input("What you want to do(+, -, \*, /, %):")

num2 = int(input("Enter second number: "))

result = 0

if operation == '+':

result = add(num1,num2)

elif operation == '-':

result = subtract(num1,num2)

elif operation == '\*':

result = multiply(num1,num2)

elif operation == '/':

result = divide(num1,num2)

elif operation == '%':

result = modulo(num1,num2)

else:

print("Please enter: +, -, \*, / or %")

print(num1, operation, num2, '=', result)

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## How it works

You saw five functions to add, subtracts, etc. Those are easy.

Then we are taking user inputs. Three inputs. They are easy too.

Then we have if-elif-else. And based on the operation, we call the right method to perform the task.

That’s it.

# 11.2: Rock Paper Scissor

## The Problem

Build s simple Rock paper Scissor game.

## Hint

You will write a lot of if-else. Think about it in a way that if the first player types rock, which one the second player should have picked to win the game. Otherwise, the second player will lose the game.

If you are not familiar with Rock Paper Scissors, you might be living under a rock for a while.

Don’t worry. You always have google. If you don’t know about the game, Google it.

## Solution

def get\_winner(p1, p2):

if p1 == p2:

return "It's a tie!"

elif p1 == 'rock':

if p2 == 'scissors':

return "First player wins!"

else:

return "Second Player wins!"

elif p1 == 'scissors':

if p2 == 'paper':

return "First player win!"

else:

return"Second player wins!"

elif p1 == 'paper':

if p2 == 'rock':

return "First player wins!"

else:

return "Second player win!"

else:

return "Invalid input!"

player1 = input("First player: rock, paper or scissors: ")

player2 = input("Second Player: rock, paper or scissors: ")

print(get\_winner(player1, player2))

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Explanation

We took 2 players' input. Send those to the get\_winner function. Inside the function, first, we check whether both the player entered the same thing. Then it will become a tie.

The next step is to check whether the first player entered the rock. Then we see the second player input. If the second player entered scissor, the second player is the winner. Otherwise, the first player is the winner

We repeated this two more times. And checked whether the first player entered paper or scissor. And based on the first player input, we compared the second player’s input.

Now think for 5 minutes, is there any different way you could have written the if-else logic here.

If you find a way, add a question here so that everyone can see your code.

# 12.2: Password generator

## The Problem

Generate a password. Your password may contain letters in uppercase or lowercase. It also may contain digits or special characters.

## Hints

To select a random character from a string, you can import random. Then use the random. choice method. This will select a character from the provided string.

Also, you can import the string module.

This is not just the string type variable. Instead, it has a lot of extra functionalities.

For example, you can use string.ascii\_letters to get all the characters a to z both in lowercase and upper case. Similarly, you can use string.digits to get all the single digits. Also, you can use string.punctuation to get all the special characters.

import string

print('All letters')

print(string.ascii\_letters)

print('all digits')

print(string.digits)

print('all special characters')

print(string.punctuation)

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Solution

import string

import random

def generate\_password(size):

all\_chars = string.ascii\_letters + string.digits + string.punctuation

password = ''

for char in range(size):

rand\_char = random.choice(all\_chars)

password = password + rand\_char

return password

pass\_len = int(input('How many characters in your password?'))

new\_password = generate\_password(pass\_len)

print('Your new password: ', new\_password)

[**Try it on Programming Hero**](https://play.google.com/store/apps/details?id=com.learnprogramming.codecamp)

## Explanation

The solution is rather simple. We imported the random module and the string module. Then we created a long string by joining all ascii\_letters, digits and special characters.

For that, we ran a for loop. In the loop, we select a random letter from the all\_chars. To select a random character, we used random.choice. Then we add the random character to the password.

# 13.3: Calculate age

## The Problem

Take a birthday of a person and then calculate the age.

This code and solution is contributed by Programmer Karim.

## Hints

Take the user birthday. Use the same method that you used in the previous problem.

Then, to calculate age, get the days and then divide the days by 365.

(This is not the perfect way to calculate age...Feel free to find better solutions)

## Solution

This code is contributed by Programmer Karim...

from datetime import datetime

import time

def calculate\_age(born):

today = datetime.today()

days = (today-born).days

age = days // 365

return age

def get\_user\_birthday():

date\_str = input("Enter your birth date in DD/MM/YYYY: ")

try:

birthday = datetime.strptime(date\_str, "%d/%m/%Y")

except TypeError:

birthday = datetime.datetime(\*(time.strptime(date\_str, "%d/%m/%Y")[0:6]))

return birthday

birthday = get\_user\_birthday()

age = calculate\_age(birthday)

print("Your age: ", age)

### Exercise 9: Check Palindrome Number

Write a program to check if the given number is a palindrome number.

A palindrome number is a number that is same after reverse. For example 545, is the palindrome numbers

**def** palindrome(number):

**print**("original number", number)

original\_num = number

# reverse the given number

reverse\_num = 0

**while** number > 0:

reminder = number % 10

reverse\_num = (reverse\_num \* 10) + reminder

number = number // 10

# check numbers

**if** original\_num == reverse\_num:

**print**("Given number palindrome")

**else**:

**print**("Given number is not palindrome")

palindrome(121)

palindrome(125)

### Exercise 10: Create a new list from a two list using the following condition

Given a two list of numbers, write a program to create a new list such that the new list should contain odd numbers from the first list and even numbers from the second list.

* Create an empty list named result\_list
* Iterate first list using a [for loop](https://pynative.com/python-for-loop/)
* In each iteration, check if the current number is odd number using num % 2 != 0 formula. If the current number is an odd number, add it to the result list
* Now, Iterate the first list using a loop.
* In each iteration, check if the current number is odd number using num % 2 == 0 formula. If the current number is an even number, add it to the result list
* print the result list

**def** merge\_list(list1, list2):

result\_list = []

# iterate first list

**for** num **in** list1:

# check if current number is odd

**if** num % 2 != 0:

# add odd number to result list

result\_list.append(num)

# iterate second list

**for** num **in** list2:

# check if current number is even

**if** num % 2 == 0:

# add even number to result list

result\_list.append(num)

**return** result\_list

list1 = [10, 20, 25, 30, 35]

list2 = [40, 45, 60, 75, 90]

**print**("result list:", merge\_list(list1, list2))

### Exercise 12: Calculate income tax for the given income by adhering to the below rules

| **Taxable Income** | **Rate (in %)** |
| --- | --- |
| First $10,000 | 0 |
| Next $10,000 | 10 |
| The remaining | 20 |

**Expected Output**:

For example, suppose the taxable income is 45000 the income tax payable is

10000\*0% + 10000\*10%  + 25000\*20% = $6000.

income = 45000

tax\_payable = 0

**print**("Given income", income)

**if** income <= 10000:

tax\_payable = 0

**elif** income <= 20000:

# no tax on first 10,000

x = income - 10000

# 10% tax

tax\_payable = x \* 10 / 100

**else**:

# first 10,000

tax\_payable = 0

# next 10,000 10% tax

tax\_payable = 10000 \* 10 / 100

# remaining 20%tax

tax\_payable += (income - 20000) \* 20 / 100

**print**("Total tax to pay is", tax\_payable)

### Exercise 14: Print downward Half-Pyramid Pattern with Star (asterisk)

**for** i **in** **range**(6, 0, -1):

**for** j **in** **range**(0, i - 1):

**print**("\*", end=' ')

**print**(" ")

### Exercise 5: Check if the first and last number of a list is the same

Write a function to return True if the first and last number of a given list is same. If numbers are different then return False

**def** first\_last\_same(numberList):

**print**("Given list:", numberList)

first\_num = numberList[0]

last\_num = numberList[-1]

**if** first\_num == last\_num:

**return** True

**else**:

**return** False

numbers\_x = [10, 20, 30, 40, 10]

**print**("result is", first\_last\_same(numbers\_x))

numbers\_y = [75, 65, 35, 75, 30]

**print**("result is", first\_last\_same(numbers\_y))

**Convert Hours into Seconds**

Write a function that converts hours into seconds.

# Python program to interchange first and last elements in a list

Given a list, write a Python program to swap first and last element of the list.

**Examples:**

**Input :** [12, 35, 9, 56, 24]

**Output :** [24, 35, 9, 56, 12]

**Input :** [1, 2, 3]

**Output :** [3, 2, 1]

**def** swapList(newList):

    size **=** len(newList)

    # Swapping

    temp **=** newList[0]

    newList[0] **=** newList[size **-** 1]

    newList[size **-** 1] **=** temp

**return** newList

# Driver code

newList **=** [12, 35, 9, 56, 24]

print(swapList(newList))

# Maximum of two numbers in Python

Given two numbers, write a Python code to find the Maximum of these two numbers.

**Examples:**

**Input:** a = 2, b = 4

**Output:** 4

**Input:** a = -1, b = -4

**Output:** -1

**Method #1:** This is the naive approach where we will compare two numbers using [if-else](https://www.geeksforgeeks.org/python-if-else/) statement and will print the output accordingly.

**def** maximum(a, b):

**if** a >**=** b:

**return** a

**else**:

**return** b

# Driver code

a **=** 2

b **=** 4

print(maximum(a, b))

# Minimum of two numbers in Python

Given two numbers, write a Python code to find the Minimum of these two numbers.

**Examples:**

**Input:** a = 2, b = 4

**Output:** 2

**Input:** a = -1, b = -4

**Output:** -4

**Method #1:** This is the naive approach where we will compare the numbers using [if-else](https://www.geeksforgeeks.org/python-if-else/) statement and will print the output accordingly.

# Python program to find the

# minimum of two numbers

**def** minimum(a, b):

**if** a <**=** b:

**return** a

**else**:

**return** b

# Driver code

a **=** 2

b **=** 4

print(minimum(a, b))

## Remove the ith character from the string using the native method

In this method, one just has to run a [Python loop](https://www.geeksforgeeks.org/python-for-loops/) and append the characters as they come and build a new string from the existing one except when the index is i.

* Python3

|  |
| --- |
| test\_str **=** "GeeksForGeeks"    # Removing char at pos 3  new\_str **=** ""    **for** i **in** range(len(test\_str)):  **if** i !**=** 2:          new\_str **=** new\_str **+** test\_str[i]    # Printing string after removal  **print** ("The string after removal of i'th character : " **+** new\_str) |

# Get Yesterday’s date using Python

**from** datetime **import** date

**from** datetime **import** timedelta

# Get today's date

today **=** date.today()

print("Today is: ", today)

# Yesterday date

yesterday **=** today **-** timedelta(days **=** 1)

print("Yesterday was: ", yesterday)

# Python program to find the type of IP Address using Regex

* **IPv4** was the primary version brought into action for production within the ARPANET in 1983. IP version four addresses are 32-bit integers which will be expressed in hexadecimal notation.
* **IPv6** was developed by the Internet Engineering Task Force (IETF) to deal with the problem of IP v4 exhaustion. IP v6 is 128-bits address having an address space of 2128, which is way bigger than IPv4. In IPv6 we use Colon-Hexa representation. There are 8 groups and each group represents 2 Bytes.

**Input:** 192.0.2.126

**Output:** IPv4

**Input:** 3001:0da8:75a3:0000:0000:8a2e:0370:7334

**Output:** IPv6

**Input:** 36.12.08.20.52

**Output:** Neither

* Take the IP address as input.
* Now, check if this IP address resembles IPv4 type addresses using regex.
* If yes, then print “IPv4” else check if this IP address resembles IPv6 type addresses using regex.
* If yes, then print “IPv6”.
* If the address doesn’t resemble any of the above types then we will print “Neither”.

# Python program to find the type of Ip address

# re module provides support

# for regular expressions

**import** re

# Make a regular expression

# for validating an Ipv4

ipv4 **=** '''^(25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)\.(

            25[0-5]|2[0-4][0-9]|[0-1]?[0-9][0-9]?)$'''

# Make a regular expression

# for validating an Ipv6

ipv6 **=** '''(([0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|

        ([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:)

        {1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1

        ,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}

        :){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{

        1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA

        -F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a

        -fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6})|:((:[0

        -9a-fA-F]{1,4}){1,7}|:)|fe80:(:[0-9a-fA-F]{0,

        4}){0,4}%[0-9a-zA-Z]{1,}|::(ffff(:0{1,4}){0,1}

        :){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9

        ])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0

        -9])|([0-9a-fA-F]{1,4}:){1,4}:((25[0-5]|(2[0-4]

        |1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]

        |1{0,1}[0-9]){0,1}[0-9]))'''

# Define a function for finding

# the type of Ip address

**def** find(Ip):

    # pass the regular expression

    # and the string in search() method

**if** re.search(ipv4, Ip):

        print("IPv4")

**elif** re.search(ipv6, Ip):

        print("IPv6")

**else**:

        print("Neither")

# Driver Code

**if** \_\_name\_\_ **==** '\_\_main\_\_' :

    # Enter the Ip address

    Ip **=** "192.0.2.126"

    # calling run function

    find(Ip)

    Ip **=** "3001:0da8:75a3:0000:0000:8a2e:0370:7334"

    find(Ip)

    Ip **=** "36.12.08.20.52"

    find(Ip)