a = "Hello, World!"

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

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

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* [**Tuple**](https://www.w3schools.com/python/python_tuples.asp) is a collection which is ordered and unchangeable. Allows duplicate members.
* [**Set**](https://www.w3schools.com/python/python_sets.asp) is a collection which is unordered, unchangeable\*, and unindexed. No duplicate members.
* [**Dictionary**](https://www.w3schools.com/python/python_dictionaries.asp) is a collection which is ordered\*\* and changeable. No duplicate members.

|  |  |
| --- | --- |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |

thistuple = ("apple", "banana", "cherry")  
print(len(thistuple))#3

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

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

* Use the style attribute for styling HTML elements
* Use background-color for background color
* Use color for text colors
* Use font-family for text fonts
* Use font-size for text sizes
* Use text-align for text alignment
* <b> - Bold text
* <strong> - Important text
* <i> - Italic text
* <em> - Emphasized text
* <mark> - Marked text
* <small> - Smaller text
* <del> - Deleted text
* <ins> - Inserted text
* <sub> - Subscript text
* <sup> - Superscript text

|  |  |
| --- | --- |
| 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 |
| None Type: | NoneType |

|  |  |  |
| --- | --- | --- |
| x = "Hello World" | str | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_str) |
| x = 20 | int | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_int) |
| x = 20.5 | float | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_float) |
| x = 1j | complex | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_complex) |
| x = ["apple", "banana", "cherry"] | list | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_list) |
| x = ("apple", "banana", "cherry") | tuple | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_tuple) |
| x = range(6) | range | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_range) |
| x = {"name" : "John", "age" : 36} | dict | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_dict) |
| x = {"apple", "banana", "cherry"} | set | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_set) |
| x = frozenset({"apple", "banana", "cherry"}) | frozenset | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_frozenset) |
| x = True | bool | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bool) |
| x = b"Hello" | bytes | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytes) |
| x = bytearray(5) | bytearray | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_bytearray) |
| x = memoryview(bytes(5)) | memoryview | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_type_memoryview) |
| x = None | NoneType |  |

x = tuple(("apple", "banana", "cherry"))

#display x:

print(x)

#display the data type of x:

print(type(x))

KIIRÁS:

('apple', 'banana', 'cherry')  
<class 'tuple'>

x = list(("apple", "banana", "cherry"))

#display x:

print(x)

#display the data type of x:

print(type(x))

KIIRÁS:

['apple', 'banana', 'cherry']  
<class 'list'>

def EgeszBeolvasas(promptszoveg): # Függvény, ami egy érvényes számot kér be

while True:

try:

szam = int(input(promptszoveg)) # Szám beolvasása

except:

pass # Ha nem számot adunk meg, akkor újra próbálkozik

if 100 < szam < 1000: # Ellenőrzi, hogy a szám 100 és 1000 között van-e

break

else:

continue

return szam

szam1 = EgeszBeolvasas("Egyik szám: ")

szam2 = EgeszBeolvasas("Másik szám: ")

if szam1 < szam2:

szamok = range(szam1, szam2 + 1) # Tartomány meghatározása

else:

szamok = range(szam2, szam1 + 1)

db = 0 # Számláló a tökéletes négyzetekhez

from math import sqrt

for szam in szamok:

if str(sqrt(szam)).endswith(".0"): # Ellenőrzi, hogy tökéletes négyzet-e

print(szam, sqrt(szam)) # Kiírja a számot és annak négyzetgyökét

db += 1 # Növeli a számlálót

print(f"{db} darab szám van {szam1} és {szam2} között, amik tökéletes négyzetek.") # Kiírja a tökéletes négyzetek számát

while True:

try:

szam=int(input("Vizsgálandó szám: "))

break

except:

pass

oszto=2

vanoszto=False

while oszto\*oszto<=szam:

if szam%oszto==0:

#print("NEM PRÍMSZÁM")

vanoszto=True

break

oszto+=1 #ami ekvivalens az oszto=oszto+1 formulával

if vanoszto:

print("NEM PRÍMSZÁM")

else:

print("PRÍMSZÁM")

while True:

try:

szam = int(input("Vizsgálandó szám: ")) # Beolvassuk a számot

break # Kilépünk a ciklusból, ha sikerült

except ValueError: # Ha nem egész számot adtak meg

print("Kérlek, adj meg egy érvényes számot!")

# Prímszám ellenőrzés

if szam < 2:

print("NEM PRÍMSZÁM") # Mivel a 0 és 1 nem prímszámok

else:

oszto = 2

vanoszto = False

while oszto \* oszto <= szam: # Csak a négyzetgyökig kell keresni

if szam % oszto == 0: # Ha van osztó

vanoszto = True

break

oszto += 1

if vanoszto:

print("NEM PRÍMSZÁM")

else:

print("PRÍMSZÁM")

def DolgozatPontszam(elertpontszam):

while True:

try:

a = int(input(elertpontszam))

except:

pass

if 0 <= a <= 100:

break

else:

print("A pontszámnak 0 és 100 között kell lennie!")

continue

return a

a = DolgozatPontszam("Írd be az elért pontszámot: ")

if a < 40:

print("elégtelen")

elif a < 55:

print("elégséges")

elif a < 70:

print("közepes")

elif a < 85:

print("jó")

else:

print("jeles")szo="Árvíztűrőtükörfúrógép"

for betu in szo:

print(betu, ord(betu))#KIIRAJA:betűk kódjait

szo="Árvíztűrőtükörfúrógép"

for betu in szo.upper():#upper beírásával, nagybeűket ír, lower-rel: kisbetű

print(betu, ord(betu))

szo="Árvíztűrőtükörfúrógép"

for betu in enumerate(szo):#indexet ad minden karakternek

print(betu)

primek1=[1,2,3,5,7]

primek2=(1,2,3,5,7)

print(f"előtte: {primek1}")

primek1[2]=113#második helyre 113-at ír

print(f"utána: {primek1}")

szo="Árvíztűrőtükörfúrógép"

print(szo)#Árvíztűrőtükörfúrógép

print(szo[3:7])#íztű

print(szo[:5])#Árvíz

print(szo[2:])#víztűrőtükörfúrógép

print(szo[:-1])#Árvíztűrőtükörfúrógé

print(szo[-1:])#p

print(szo[0:len(szo):2])#Ávzűőüöfrgp

print(szo[0:21:2])#Ávzűőüöfrgp

print(szo[::-1])#pégórúfrökütőrűtzívrÁ

employee.py-ban:

#Név:Anyja neve:Település:Lakcím:Fizetés:Születés:Jutalom

class Employee:

def \_\_init\_\_(self,name, mother, city, address, salary, born, bonus):

self.name=name

self.mother=mother

self.city=city

self.address=address

self.salary=salary

self.born=born

self.bonus=bonus

filereader.py-ben:

from employee import Employee

class FileReader:

def \_\_init\_\_(self):

pass

def reading(self):

empList=[]

file=open("employee.txt","r",encoding="utf8")

file.readline()

row=file.readline()

while(row):

rowSp=row.split(":")

emp=Employee(rowSp[0],rowSp[1],rowSp[2],rowSp[3],rowSp[4],rowSp[5],rowSp[6])

empList.append(emp)

row=file.readline()

return empList

worker.py-ben:

from filereader import FileReader

class Worker:

def \_\_init\_\_(self):

self.empList=[]

def getFileContent(self):

self.empList=FileReader().reading()

def getVacCount(self):

counter=0

for emp in self.empList:

if(emp.city=="Vác"):

counter+=1

print(counter)

worker=Worker()

worker.getFileContent()

worker.getVacCount()

import math

def plus(num01,num02):

result=num01+num02

return result

def multiply(number): #hatvanyozas

# result = number\*number - bena

# result=number\*\*2 - ez is

result=pow(number,6)

return result

def square(number):

result=math.sqrt(number)

return result

def calcCirArea(rad):

result=pow(rad,2)\*math.pi

# result=rad\*\*2\*3,14

# print(math.pi)

return result

class Worker: #egy osztalyhoz mindig kell konstruktor

def \_\_init\_\_(self): #konstruktor

self.name=""

self.role=""

def getName(self):

return self.name

def setName(self,name):

self.name=name

def getRole(self):

return self.role

def setRole(self,role):

self.role=role

class Hr:

pek=Worker()

pek.setName("Pista")

pek.setRole("Pek")

print(pek.getName())

print(pek.getRole())

Hr()

pi kiírása:

import math

print(math.pi)

négyzetgyök számítás:

import math

def negyzetgyok(szam):

a = math.sqrt(szam)

return a

szam = int(input("szám"))

print("A négyzetgyöke:", negyzetgyok(szam))