

# Python

base

# Start(شروع)

```
message = 'hello'
```

```
print(message)
```

```
print('hello')
```

```
print(54)
```

```
print(7+9)
```

```
print(15/4)
```

```
print('5+4')
```

```
print('5'+'4')
```

```
print('ali'+'reza')
```

```
print(5>3)
```

```
print(6<5)
```

# انواع داده (Data Type)

- متغیرها و نام گذاری متغیرها

\_name

num1

خطای نامگذاری که با عدد شروع شود #number 1

خطای نامگذاری فاصله بین کلمات #first\_name

first\_name

خطای نام گذاری با کلمات زبان پایتون #Print

```
message = 'hello'
```

```
print(message)
```

# اعداد - (انواع داده) Data Type

- integer >> 5, 10, 17
- float >> 2.4, 3.25, 0.8

#2, 2.0

3/2 >> 1 #تقسیم صحیح

3.0/2 >> 1.5 #تقسیم اعشاری

age = 23

print("happy" + age + "birthday") #error : int in str

print("happy" + str(age) + "birthday") #تبدیل عدد به متن

int(...), float(...), str(...), ...

## اعداد - (انواع داده) Data Type

```
name="ali"
```

```
family="asadi"
```

```
print("name is : "+name+" "+family)
```

```
print(f"name is : {name} {family}")
```

# دریافت ورودی (Input)

دریافت ورودی از کاربر #

```
num = input()
```

```
name = input("enter your name : ")
```

# IF (دستورات شرطی)

مخالف (نا برابر)

<	<=	>	>=	==	!=	not
---	----	---	----	----	----	-----

- **if - else**

```
pass = "1234"
```

```
if pass == inputPass :
```

```
    print("yes")
```

```
else :
```

```
    print("error")
```

- **if - elif - else**

```
num = 10
```

```
if num > 0 :
```

```
    print("pos")
```

```
elif num < 0 :
```

```
    print("neg")
```

```
else :
```

```
    print("zero")
```

## IF (دستورات شرطی) - AND, OR

- if **Condition1** or **Condition2**:

if **a > 0** and **a < 10** → ?

- if **Condition1** and **Condition2**:

if **a < 0** or **a > 10** → ?

OR		
Condition 1	Condition 2	or
T	T	T
T	F	T
F	T	T
F	F	F

AND		
Condition 1	Condition 2	and
T	T	T
T	F	F
F	T	F
F	F	F

T → True  
F → False



# Match

```
n=int(input('enter number : '))
```

```
match n:
    case 1:
        print('saturday')
    case 2:
        print('sunday')
    case 3:
        print('monday')
    case 4:
        print('tuesday')
    case 5:
        print('wednesday')
    case 6:
        print('thursday')
    case 7:
        print('friday')
    case _:#-----default case
        print('invalid input')
```

# Beep

- `import winsound`
- `winsound.Beep(زمان, فرکانس)`

```
winsound.Beep(800,500)
```

```
winsound.Beep(1000,1000)
```

```
winsound.Beep(1500,500)
```

```
winsound.Beep(500,800)
```

# Try (کنترل خطا)

try:

#کد برنامه

except :

#جزییات خطا

#کد برنامه شما در قسمت try نوشته خواهد شد  
و در صورت بروز خطا اجرای کد لغو و برنامه به  
قسمت except منتقل شده و دستورات این بخش  
اجرا خواهد شد،

در صورت عدم بروز خطا catch اجرا نخواهد شد

try:

num1=int(input('enter number 1 : '))

num2=int(input('enter number 2 : '))

print(num1+num2)

except:

print('error')

# Try (کنترل خطا)

نمایش متن دقیق علت خطا#

برای مشاهده خطا یک ورودی غیر عددی وارد کنید#

```
try:
    num1=int(input('enter number 1 : '))
    num2=int(input('enter number 2 : '))
    print(num1+num2)
except Exception as e:
    print(e)
```

# Try (کنترل خطا)

کنترل خطا بدون نمایش یا اطلاع خطا#

try:

```
num1=int(input('enter number 1 : '))
```

```
num2=int(input('enter number 2 : '))
```

```
print(num1+num2)
```

except:

```
pass
```

# لیست ها (List)

- names = ['ali', 'reza', 'maryam']

- names[index]

```
print(names)
```

```
print(*names, sep='/')
```

```
print(names[0])
```

```
print(names[1])
```

```
print(names[2])
```

```
print(names[-1]) # آخرین آیتم (عنصر) لیست
```

```
print(names[-2]) # یکی مانده به آخرین آیتم لیست
```

ali	reza	maryam
0	1	2

- edit/change مقداردهی و تغییر مقدار #

```
names[0] = 'amir'
```

- len() طول (تعداد عناصر) لیست #

```
print(len(names))
```

```
print(len(input('enter text : ')))
```

# لیست ها (List)

- `min()`, `max()`, `sum()` # برای لیست های عددی

```
digits = [1,2,3,4,5,6,7,8,9]
```

```
print(min(digits))      1
```

```
print(max(digits))      9
```

```
print(sum(digits))      45
```

- `Slice` # ایجاد برشی از لیست اصلی

```
print(names[0:3]) # اندیس 0 تا 3
```

```
print(names[:4]) # اندیس 0 تا 4
```

```
print(names[2:]) # اندیس 2 تا انتهای لیست
```

```
print(names[:]) # به اندازه لیست
```

```
print(names[-3:]) # سه عضو آخر
```

# لیست ها (List)

- **sorted()** # صرف نمایش عناصر لیست به ترتیب بدون جابجایی عناصر

```
print(sorted(names))  
print(sorted(names, reverse=True))  
print(names)
```

- **sort()** # مرتب سازی عناصر لیست جابجایی عناصر

```
names.sort()  
print(names)  
names.sort(reverse=True)  
print(names)
```

- **reverse()** # ایجا لیست معکوس (انتها به ابتدا)

```
names.reverse()  
print(names)
```



# لیست ها (List)

- **append(value)** # اضافه کردن عنصر به انتهای لیست

```
names.append('Zahra')
```

- **insert(index, value)** # اضافه کردن عنصر به مکان (اندیس) خاصی از لیست

```
names.insert(0, 'Sadegh')
```

- **del** # حذف عنصر بر اساس اندیس

```
del names[0]
```

- **remove(value)** # حذف عنصر بر اساس مقدار (اولین مقدار معادل حذف خواهد شد)

```
names.remove('reza')
```

- **pop(index)**

گرفتن (خواندن) آخرین عنصر لیست و سپس حذف آن # `name = names.pop()`

گرفتن (خواندن) اندیس خاصی از لیست و سپس حذف عنصر آن # `name = names.pop(1)`

# لیست ها (List)

```
names = ['ali', 'reza', 'maryam']
```

```
if 'ali' in names :
```

```
    #1
```

```
else :
```

```
    #2
```

```
#تست خالی نبودن لیست
```

```
if names :
```

```
    #1
```

```
else :
```

```
    #2
```

# لیست ها (List)

جستجوی یک لیست در لیست دیگر#

```
allNames = ['ali' , 'reza' , 'maryam']
```

```
names = ['ali' , 'maryam']
```

```
if names in allNames :
```

```
    #1
```

```
else :
```

```
    #2
```

خارج از مبحث (مطالعه آزاد)#

نمونه متصل vs نمونه مجزا#

```
tempList = names[:] vs tempList = names
```

# Range, RandRange, Sample

- range (گام, اندیس پایان, اندیس شروع)

```
print(range(5)) #0 1 2 3 4
```

```
print(range(3,10)) #3 4 5 6 7 8 9
```

```
print(range(5,1,-1)) #5 4 3 2
```

```
print(*range(5,1,-1)) #5 4 3 2
```

```
print(list(range(5))) #[0, 1, 2, 3, 4]
```

# Range, RandRange, Sample

- randrange(انديس پايان, انديس شروع)

```
import random
```

```
print(random.randrange(1,10))
```

```
from random import randrange
```

```
print(randrange(1,10))
```

- sample(تعداد, محدوده بازه)

```
random.sample(range(100), 10)
```

# Range, List()

- **list()** #ایجاد لیست اعداد

```
numbers = list(range(1,10))
```

```
print(numbers)          #[1,2,3,4,5,6,7,8,9]
```

```
numbers = list(range(2,10,2)) #ایجاد دنباله اعداد با فاصله منظم
```

```
print(numbers)          #[2,4,6,8]
```

# تاپل، دیکشنری، مجموعه ( Tuple, Dictionary, Set )

- **Tuple**

```
person=('ali','ahmadi','0911')
```

```
person[0] → 'ali'
```

```
person[1] → 'ahmadi'
```

```
person[2] → '0911'
```

نمیتوان به تاپل یک عضو جدید اضافه کرد یا یک عضو را ویرایش کرد#

# Dictionary

- **Dictionary**

```
person={'fname':'ali','lname':'ahmadi'}
```

```
person['fname'] → 'ali'
```

#افزودن عضو جدید به فرهنگ لغت

```
person['phone']='0911'
```

#نتیجه

```
{'fname':'ali','lname':'Ahmadi','phone':'0911'}
```

به تاپل یک عضو جدید اضافه کرد یا یک عضو را ویرایش کرد



# Dictionary

```
listDict=[{'name':'ali','family':'amiri'},  
           {'name':'reza','family':'ahmadi'}]  
print(listDict)  
print(listDict[0])  
print(listDict[0]['family'])
```

# Set

```
setA={5,8,3,12}  
setB={10,8,4,9}  
print(setA)  
print(*setA)  
print(list(setA)[2])  
print(setA.union(setB))  
print(setA.intersection(setB))  
print(setA.difference(setB))  
print(setB.difference(setA))  
setA.remove(5)  
setA.add(11)  
print(setA)  
print(max(setA))
```

# lambda

```
#filter(lambda
```

- `filter(lambda [متغیر]:[شرط],[نام لیست])`

```
nums = [1,5,-3,12,9,6]
```

```
_f = filter(lambda i: i>5, nums)
```

```
print(_f)
```

```
print(list(_f))
```

```
score=[12.5,15,14.25,8,17.5,18,15,19,6.75,20]
```

```
result=list(filter(lambda n: n>10 and n<15, score))
```

```
result=list(filter(lambda s: s>15 or s<10, score))
```

## For (حلقه ها)

```
names = ['ali' , 'reza' , 'Maryam','reza']
```

- **for** item **in** names :

```
    print(item)
```

- **for** item **in** names :

```
    if item == 'reza' :
```

```
        print(item)
```

- end

```
print(...,end=' ')
```

```
print(...,end='- ')
```

```
print(...,end='\n')
```

## For (حلقه ها)

```
myStr = 'hello world'
```

- **for** n **in** myStr :  
    print(n)
- **for** n **in** myStr :  
    **if** n=='l' :  
        print(n.upper(),end=' ')  
    **else**:  
        print(n,end=' ')

## For (حلقه ها)

#تورفتگی های مهم

```
for item in names :
```

```
    print(item)
```

```
    print('finish')
```

#یا

```
for item in names :
```

```
    print(item)
```

```
print('finish')
```

## For (حلقه ها)

- `range()` ایجاد دنباله اعداد

```
for item in range(1,5) :  
    print(item)  
#1, 2, 3, 4
```

```
price=[15000,37500,5000,23000,20000]  
for i in range(0,len(price)):  
    if price[i]<=15000:  
        p=price[i]  
        p=p-(p*10/100)  
        price[i]=p  
print(price)
```

## For (حلقه ها)

```
squares=[]  
for item in range(1,11):  
    square = item**2      به توان دو  
    squares.append(square)  
print(squares)
```

خارج از مبحث (مطالعه آزاد)

- **List Comprehension**

```
squares = [item**2 for item in range(1,11)]  
print(squares)
```



# Jump (پرش)

- **break**

خروج کامل از حلقه ها و بلوک های کد#

- **continue**

دور فعلی حلقه را لغو و به ابتدای دور بعدی جهش میکند#

- **return**

انتقال (بازگشت) مقادیر در توابع و متدها#

- **pass**

جهش از هر دستور و بلوک کد در حلقه ها مشابه `continue`#

# While (حلقه ها)

**while** شرط :

کد برنامه #

```
n=input("enter number : ")
```

```
while n>0 :
```

```
    n=n/10
```

```
    m+=1
```

```
print(m)
```

## While (حلقه ها)

**while** item **in** List :

کد برنامه #

```
names=['cat','dog','cat']  
print(names)
```

```
while 'cat' in names:  
    names.remove('cat')
```

```
print(names)
```

## string function (متن یا رشته)

- string >> "my first code" یا 'my first code'

```
mystr = "My name" >> mystr.upper() >> MY NAME
```

```
>> mystr.lower() >> my name
```

```
>> mystr.title() >> My Name
```

```
mystr = " my "
```

```
>> mystr >> my
```

```
>> mystr.strip() >> my #lstrip(), rstrip()
```

"\n" رفتن خط بعدی

"\t" یک تب فاصله

# string function string (متن یا رشته)

- توابع رشته ای

```
str="hello, world!"
```

```
print(str.replace('l','*')) # جایگزین کردن متن اول با الگوی دوم
```

```
print(str.find('w')) # جستجوی عبارت ورودی
```

```
print(str.index('w')) # جستجوی عبارت ورودی
```

```
print(str.capitalize()) # تبدیل حرف اول عبارت به حرف بزرگ
```

```
print(str.title()) # تبدیل حرف اول کلمات به حرف بزرگ
```

```
print(str.lower()) # تبدیل حروف به حروف کوچک
```

```
print(str.upper()) # تبدیل حروف به حروف بزرگ
```

```
print(str.count('l')) # تعداد تکرار عبارت ورودی در متن
```

```
print(str.isdigit()) # آیا متن ورودی عدد است؟
```

```
print(str.isspace()) # آیا عبارت ورودی فضای خالی (اسپیس) است؟
```

```
print(str.swapcase()) # آیا عبارت ورودی فضای خالی (اسپیس) است؟
```

```
print(str[6:10:1]) # گام : اندیس پایان : اندیس شروع
```

# Tkinter

```
from tkinter import *
```

```
app=Tk()
```

```
#gui code
```

```
app.mainloop()
```

# Tkinter

```
# form=Tk(className='my app')
#form
form.title('
form.geometry('512x512')
form.resizable(0,0)
# form.configure(bg='red')
form['bg']=' #0B666A'
#label
label1=Label(form)
label1['text']='سلام به همه'
label1.place(x=10,y=10)
label1['font']=(None,25)
label1['bg']=' #071952'
label1['fg']=' #97FEED'
label1['width']=25
label1['height']=2
```

```
#entry
entry1=Entry(form)
entry1.place(x=10,y=100)
entry1['font']=(None,18)
Entry1['show']='*'
#text
text1=Text(form)
text1.place(x=10,y=150)
text1['width']=50
text1['height']=10
form.mainloop()
#button
btn1=Button(app,text='submit')
btn1.pack()
```

```
#OptionMenu/dropdown
option=['red','green','blue']
default=StringVar(form,[text])
default.set(option[0])
drop1=OptionMenu(form,default,*option)
drop1.place(x=150,y=150)
#checkbox
Checkbutton
#radiobutton
Radiobutton
#listBox
listBox1=ListBox(form)
listBox1.insert(1,"value1")
listBox1.insert(2,"value")
listBox1.size()
#menu
menubar=Menu(form)
Form.config(menu=menubar)
fileMenu=Menu(menubar,tearoff=0)
menubar.add_cascade(label='file',menu=fileMenu)
fileMenu.add_command(label='exit')
```

# TKinter

**Label** -> ['text']

**Entry** -> Insert()

Insert('end', str)

get()

get('1.0', 'end')



# TKinter -> photoImage

1-

```
#imageLabel
imgpath = 'images/ball.png'
imgData = PhotoImage(file=imgpath)
#imgData = imgData.zoom(2) #with 250, I ended up running out of memory
imgData = imgData.subsample(10) #mechanically, here it is adjusted to 32
instead of 320
#imgData=imgData.subsample(-1)
imgBox = Label(form, image = imgData)
imgBox.pack()
```

# install library

Terminal>

```
py -m pip list
```

```
py -m pip install ...
```

*Pillow*

*persiantools*

*pytz*

# TKinter -> photoImage

2-

```
Terminal> py -m pip install Pillow
```

```
#imagePil
from PIL import Image
from PIL import ImageTk
width = 50
height = 50
img = Image.open("images/ball.png")
img = img.resize((width,height))
imgData = ImageTk.PhotoImage(img)
imgBox = Label(form,image=imgData)
imgBox.pack()
```

# Function (توابع)

- **def**

**def** (مقادیر پیش فرض=پارامترها) نام تابع :

کد برنامه#

#مقادیر پیش فرض اختیاریست

#####

#تعریف تابع

**def** Sum1():

n=int(input('enter number 1 : '))

m=int(input('enter number 2 : '))

print(n+m)

#####

#فراخوانی (اجرای) تابع

**Sum1()**

# Function (توابع)

#تابع با پارامتر ورودی

```
def Sum2(a,b):  
    print(a+b)
```

```
n=int(input('enter number 1 : '))
```

```
m=int(input('enter number 2 : '))
```

#فراخوانی (اجرا) تابع به همراه ارسال پارامترهای مورد نیاز

```
Sum2(n,m)
```

# Function (توابع)

#تابع با مقدار بازگشتی

```
def Sum3(a,b):
```

```
    return a+b
```

```
n=int(input('enter number 1 : '))
```

```
m=int(input('enter number 2 : '))
```

#فراخوانی (اجرا) تابع به همراه دریافت نتیجه اجرای تابع (مقدار بازگشتی تابع)

```
result=Sum3(n,m)
```

```
print(result)
```

```
#یا
```

```
print(Sum3(n,m))
```

# Function (توابع)

تابع بامقادیر پیش فرض برای پارامتر ورودی

```
def Sum4(a, b=5):  
    print(a+b)
```

فراخوانی (اجرا) تابع بدون ارسال تمام پارامترها و در نتیجه استفاده تابع از مقادیر پیش فرض تعریف شده

```
n=int(input('enter number 1 : '))  
Sum4(n)
```

```
#####
```

```
def Sum5(a, b):  
    print(a+b)
```

```
Sum4(b=5, a=4)
```

# Event

```
defaultColor = btnBck.cget('bg')  
btnBck.bind("<Enter>", MouseEnter)  
btnBck.bind("<Leave>", MouseLeave)
```

```
def MouseEnter(enter):  
    btnBck['bg']='red'  
    btnBck['fg']='white'
```

```
def MouseLeave(leave):  
    btnBck['bg']=defaultColor  
    btnBck['fg']='black'
```



# Text File (فایل های متنی)

بازکردن و نمایش محتوای یک فایل موجود#

فایل ها در مسیر ذخیره کد پایتون#

```
with open('نام و پسوند فایل') as مخزن ذخیره:  
    txt = مخزن ذخیره.read()  
    print(txt)
```

#####

```
with open('myfile1.txt') as file_object:  
    txt = file_object.read()  
    print(txt)
```

# Text File (فایل های متنی)

بازکردن و نمایش محتوای یک فایل موجود#

فایل ها درون پوشه ای در مسیر ذخیره کد پایتون#

```
with open('نام و پسوند فایل/نام پوشه') as مخزن ذخیره:  
    txt = مخزن ذخیره.read()  
    print(txt)
```

#####

```
with open('myfiles/myfile1.txt') as file_object:  
    txt = file_object.read()  
    print(txt)
```

# Text File (فایل های متنی)

#در صورتی که فایل وجود نداشته باشد فایل را خواهد ساخت#

#نوشتن در فایل خالی#

#محتوای فعلی(قبلی) پاک خواهد شد#

```
with open('فایل', 'w') as file_object:  
    file_object.write("متن")
```

#####

```
with open('myfile4.txt', 'w') as file_object:  
    file_object.write("create by python\n")
```

# Text File (فایل های متنی)

افزودن به محتوای فعلی فایل#

```
with open('فایل', 'a') as file_object:  
    file_object.write("متن")
```

#####

```
with open('myfile3.txt', 'a') as file_object:  
    file_object.write("appen text by python\n")
```

# Text File (فایل های متنی)

خواندن سطر به سطر#

```
with open('myfiles/myfile2.txt') as file_object:  
    for line in file_object:  
        print(line)
```

حذف فاصله بین خطوط#

```
print(line.rstrip())
```

# Text File (فایل های متنی)

```
import os  
os.remove(fileName)
```

# CSV File (فایل های دارای قالب)

.CSV

#Comma Separated Values

#مقادیر جدا شده با ویرگول

#write/create csv file

```
data = [['sara', 22], ['omid', 24], ['nima', 25]]
```

```
with open('data.csv', 'w') as csv_obj:
```

```
    writer=csv.writer(csv_obj)
```

```
    writer.writerow(['name', 'age'] )
```

```
    writer.writerows( data )
```

# CSV File (فایل های دارای قالب)

.CSV

#Comma Separated Values

#مقادیر جدا شده با ویرگول

#read csv file

```
with open('data.csv') as csv_obj:
```

```
    data = csv.reader(csv_obj)
```

```
    header = next(data)
```

```
for row in data:
```

```
    print(row)
```

```
    print(header[0] + " : " + row[0])
```



# JSON File (فایل های دارای قالب)

**.json**

**#[key] : [value]**

**import json**

**#json object**

**person = '{ "name": "John", "age": 30, "city": "New York" }'**

**#parse json**

**p=json.loads(person)**

**print(p["age"])**

# JSON File (فایل های دارای قالب)

```
import json
```

```
#json object
```

```
person = '{ "name":"John", "age":30, "city":"New York" }'
```

```
#create/write json file
```

```
with open('person.json','w') as json_obj:
```

```
    json.dump(person,json_obj)
```

## JSON File (فایل های دارای قالب)

```
#read json file
with open('person.json') as json_obj:
    person = json.load(json_obj)

p = json.loads(person)

print(p["age"])
```

# JSON File (فایل های دارای قالب)

#کار با مجموعه ای از جیسون ها

```
import json
```

```
persons =[{"name":"ali", "age":35, "city":"tehran" }, {"name":"sara", "age":30, "city":"rasht" }]
```

```
p=json.dumps(persons)#convert python object to json string
```

```
with open('person.json','w') as json_obj:
```

```
    json.dump(p,json_obj)#write into a file with json format
```

# JSON File (فایل های دارای قالب)

#کار با مجموعه ای از جیسون ها

```
with open('person.json') as json_obj:
```

```
    data=json.load(json_obj)#read the JSON document from file(stream)
```

```
print(data)
```

```
newp=json.loads(data) #read the JSON string
```

```
print(newp)
```

```
print(newp[0])
```

```
print(newp[0]["name"])
```

```
for item in newp:
```

```
    print(item)
```

```
    print(item["name"])
```

# XML File (فایل های دارای قالب)

فایلی به نام myxml.xml بسازید

```
<data>  
  <model name="model1">  
    <name>sara</name>  
    <age>30</age>  
  </model>  
  <model name="model2">  
    <name>ali</name>  
    <age>35</age>  
  </model>  
</data>
```

# XML File (فایل های دارای قالب)

```
from xml.dom import minidom
```

```
data = minidom.parse('myxml.xml')
```

```
models = data.documentElement.getElementsByTagName('model')
```

# نمایش اطلاعات یک سطر

```
print(models[1].getElementsByTagName('name')[0].childNodes[0].data)
```

# نمایش تمام اطلاعات

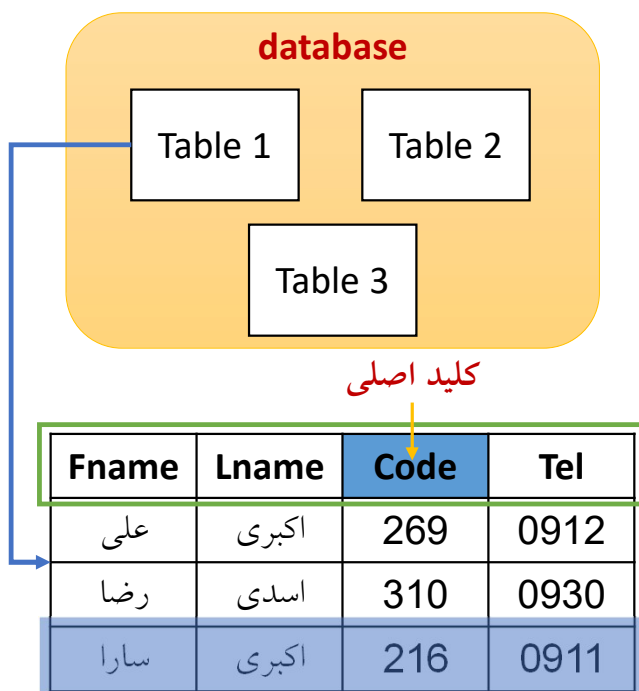
```
for model in models:
```

```
    print(model.getElementsByTagName('name')[0].childNodes[0].data)
```

```
    print(model.getElementsByTagName('age')[0].childNodes[0].data)
```

# Database (پایگاه داده)

## • SQL



پایگاه داده (Database)

- مجموعه یک یا چند جدول از اطلاعات

جدول (Table)

- مجموعه اطلاعات مرتبط با یک موضوع یا مسئله #مثال : اطلاعات فردی دانش آموزان

فیلد (Field)

- عناوین اطلاعات هر جدول شامل نام و نوع فیلد #متن، عدد، تاریخ و ...

رکورد (Record)

- یک سطر اطلاعات از هر جدول

کلید اصلی (Primary Key)

- فیلد منحصر به فرد (محتوای غیر تکراری) که نباید خالی باشد

کلید خارجی (Foreign Key)

- فیلد ارتباط از یک جدول به جدول دیگر



# Database (پایگاه داده) - CRUD

Table 1: Account

Fname	Lname	Code	Tel
علی	اکبری	269	0912
رضا	اسدی	310	0930
سارا	اکبری	216	0911

## SELECT نمایش اطلاعات

SELECT	نام فیلد	From	نام جدول
SELECT	Fname, Lname	From	Account
SELECT	Fname, Lname, Code, Tel یا *	From	Account

## SELECT جستجو

SELECT	نام فیلد	From	نام جدول	where	شرط
SELECT	*	From	Account	where	Lname = 'اکبری'

## DELETE حذف رکورد

DELETE	From	نام جدول	where	شرط
DELETE	From	Account	where	Code = 216

## INSERT ثبت رکورد

INSERT	Into	(فیلدها) نام جدول	values	مقادیر متناظر هر فیلد
INSERT	Into	Account(Fname, Lname, Code, Tel)	values	(@F='مریم', @L='میرزاخانی', @C=190, @T='0915')

## UPDATE ویرایش رکورد

UPDATE	نام جدول	Set	مقادیر متناظر هر فیلد	where	شرط
UPDATE	Account	Set	Tel='0915'	where	Code = 310

# Database (پایگاه داده)

- **SQLite**

```
import sqlite3
```

#ساخت دیتابیس در صورتی که وجود نداشته باشد و اتصال به آن

```
conn = sqlite3.connect('mydatabase.db')
```

- **Create Table**

#ساخت جدول در صورتی که وجود نداشته باشد

```
conn.execute('''CREATE TABLE IF NOT EXISTS Persons  
                (Id INTEGER PRIMARY KEY,  
                 Fname TEXT NOT NULL,  
                 Lname TEXT NOT NULL,  
                 Phone TEXT NOT NULL);''')
```

# Database (پایگاه داده)

ثبت (درج) رکورد در جدول #

- **Insert**

```
person=('ali','ahmadi','0911') # تاپل tuple
```

```
conn.execute('INSERT INTO Persons(Fname,Lname,Phone) VALUES (?, ?, ?)', person)
```

```
person=('sara','sadeghi','0930')
```

```
conn.execute('INSERT INTO Persons(Fname,Lname,Phone) VALUES (?, ?, ?)', person)
```

```
conn.commit()
```

# Database (پایگاه داده)

نمایش رکوردهای جدول#

- **Select**

```
cursor=conn.execute('SELECT * FROM Persons')  
for c in cursor:  
    print(c)  
    print(c[1])  
  
cursor.close();
```

# Database (پایگاه داده) – tree view

نمایش رکوردهای جدول

```
from tkinter import ttk
```

```
tree = ttk.Treeview(form, column=("c1", "c2"), show='headings')
```

```
tree.column("#1", anchor=CENTER)
```

```
tree.heading("#1", text="ID")
```

```
tree.column("#2", anchor=CENTER)
```

```
tree.heading("#2", text="FNAME")
```

```
tree.pack()
```

```
rows=con.execute('SELECT * FROM Persons')
```

```
for row in rows:
```

```
    #print(row)
```

```
    tree.insert('',END,values=row)
```

# Database (پایگاه داده)

ویرایش رکورد موجود در جدول#

- **Update**

```
data=('0911222','sadeghi')
```

```
conn.execute('UPDATE Persons SET Phone=? WHERE Lname=?',data)
```

```
conn.commit()
```

جستجو بین رکوردهای جدول#

- **Select-Where (Search)**

```
search=('sadeghi',)
```

```
cursor=conn.execute('SELECT * FROM Persons where Lname=?',search)
```

```
for c in cursor:
```

```
    print(c)
```

```
cursor.close();
```

# Database (پایگاه داده)

حذف رکورد موجود از جدول

- **Delete**

```
value=('sadeghi',)
```

```
conn.execute('DELETE FROM Persons where Lname=?',value)
```

```
conn.commit()
```

حذف جدول

- **Drop**

```
conn.execute('DROP TABLE persons')
```

```
conn.commit()
```

# کلاس ها و اشیاء (class)

```
import math
```

```
#شی دایره
```

```
class Circle():
```

```
    PI=math.pi #شامل ویژگی ثابت عدد پی
```

```
    radius = 1 #ویژگی متغیر شعاع
```

```
    #متد سازنده (برای همه کلاس ها) جهت تعریف و مقداردهی اولیه ویژگی ها
```

```
    def __init__(self, _radius=1):
```

```
        self.radius = _radius
```

```
    def Perimeter(self):#متد محاسبه محیط دایره
```

```
        r=float(self.radius)
```

```
        return 2*r*self.PI
```

```
    def Area(self):#متد محاسبه مساحت دایره
```

```
        r=float(self.radius)
```

```
        return r*r*self.PI
```

کلاس ها (اشیا) مجموعه ای از ویژگی ها یا خاصیت ها (property) و متدها یا عملکردها (method) هستند

هرپدیده ای در جهان واقعی میتوان توسط این دو مورد (ویژگی ها و عملکردها) شبیه سازی شود

PI متغیر با حروف بزرگ مفهوم const یا ثابت (غیر قابل تغییر)



# کلاس ها و اشیاء (class)

نمونه سازی از کلاس و اجرای متدهای داخلی

```
c = Circle(input('enter circle radius : '))  
  
print(c.radius)  
print(c.Perimeter())  
print(c.Area())
```

کلاس و کد پایتون دو فایل جدا از هم (در یک دایرکتوری یا پوشه)

```
MyClass.py -> class [className]:  
MyPython.py -> from MyClass import [className]  
[className].[method]
```

# پروژه پایان دوره

مرور مباحث

[PythonPractice/FinalProject at main · h-abdizadeh/PythonPractice \(github.com\)](https://github.com/h-abdizadeh/PythonPractice/tree/main/FinalProject)

<https://github.com/h-abdizadeh/PythonPractice/tree/main/FinalProject>