1. Simple hello world program

Rather then any other language python is easy here we just write print and our string and that’s it will print output on console.

print("Hello world")

2.User input

In this input you enter int float it will take as a string always

a=input("Enter your name:")

print(a)

So for different data type you need to convert string to a int or float.

2.1 Conversion of user input

Here we convert a to int

a=input("Enter your number:")

a=int(a)

3 Operator

Operators are used to perform operations on variables and values.

3.1 Arithmetic operator

This operator perform the mathematical operation in 2 variable.

a=30

b=10

print("The value of 30+10 ",a+b)

print("The value of 30-10 ",a-b)

print("The value of 30\*10 ",a\*b)

print("The value of 30/10 ",a/b)

3.2 Comparison operator

a==b is equal or not

a!=b is does not equals

a>b a is greater then b

a<b a is less then b

a>= b is greater then or equals to b

a<= b is less then or equals to b

a=30

b=10

print("== operator ",(a==b))

print("!= operator ",(a!=b))

print("> operator ",(a>b))

print("< operator ",(a<b))

print(">= operator ",(a>=b))

print("<= operator ",(a<=b))

3.3 Logical operator

and operator if both condition is true then it written true

or operator if one condition is true then it written true

not operator reverse the result, returns False if the result is true

a=30

b=10

print("and operator ",(a<5 and b>9))

print("or operator ",(a<5 or b>9))

print("not operator ",(not(a<5 and b>9)))

3.4 Identify operator

a is b means both variable is same then it returns true

a is not b means both variable is not same then it returns true

a=30

b=30

print("is operator ",(a is b))

print("is not operator ",(a is not b))

4. List

sts are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are [Tuple](https://www.w3schools.com/python/python_tuples.asp), [Set](https://www.w3schools.com/python/python_sets.asp), and [Dictionary](https://www.w3schools.com/python/python_dictionaries.asp), all with different qualities and usage.

Lists are created using square brackets

4.1 Create a list

Here we make a variable a as a list and simple print it.

a=["Banana","Grapes","Cheery","Apple"]

print(a)

4.2 Print the member of list by its index

a[0] means a first element a[1] means second.

a=["Banana","Grapes","Cheery","Apple"]

print(a[0])

4.3 List slicing

Means to print a index 0 to 2 means 2 element is printed

a=["Banana","Grapes","Cheery","Apple"]

print(a[0:2])

4.4 Append the list

.append() is function to append a list

a=["Banana","Grapes","Cheery","Apple"]

a.append("Mangoes")

print(a)

4.5 Remove the list

.remove is used for removing a element from the list

a=["Banana","Grapes","Cheery","Apple"]

print(a)

a.remove("Apple")

print(a)

4.6 Sort a list

.sort is used for sort a list

a=["Banana","Grapes","Cheery","Apple"]

a.sort()

print(a)

4.7 Combine a list

a=["Banana","Grapes"]

b=["Kiwi","Mangoes"]

c=a+b

print(c)

4.8 All List method

clear() Removes all the elements from the list

copy() Returns a copy of the list

count() Returns the number of elements with the specified value

index() Returns the index of the first element with the specified value

insert() Adds an element at the specified position

pop() Removes the element at the specified position

reverse() Reverses the order of the list

a=["Banana","Grapes","Kiwi"]

print("---clear---")

print(a)

a.clear()

print(a)

print("---clear---")

b=["Banana","Grapes","Kiwi"]

print("---copy---")

print(b)

c=b.copy()

print(c)

print("---copy---")

print("---count---")

print(b.count("Banana"))

print("---count---")

print("---Index---")

print(b.index("Kiwi"))

print("---Index---")

print("---Insert---")

b.insert(1,"Apple")

print(b)

print("---Insert---")

print("---Pop---")

b.pop(2)

print(b)

print("---Pop---")

print("---Reverse---")

b.reverse()

print(b)

print("---Reverse---")

5. Tuples

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and unchangeable.

Tuples are written with round brackets

5.1 Create a tuple and access it

Here we first define a tuple and then print it and also access it one element by print with it’s index.

a=("Banana","Apple","Kiwi")

print(a)

print(a[0])

5.2 Join tuples

You simple use + operator for combine of a tuples.

a=("Banana","Apple")

b=("Brinjal","Potato")

c=a+b

print(c)

5.3 Tuples methods

count() Returns the number of times a specified value occurs in a tuple

index() Searches the tuple for a specified value and returns the position of where it was found

a=("Banana","Apple","Apple","Kiwi")

print(a.count("Apple"))

print(a.index("Kiwi"))

6. Set

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is unordered, unchangeable\*, and unindexed

6.1 Create a set

You can create a set by having a curly brackets open and close.

a={1,3,4,54,1}

print(a)

6.2 Access the element

you cannot access items in a set by referring to an index or a key. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

You can print it directly by using a loop.

a={1,2,4,56,89}

for z in a:

    print(z)

You can check that the element is present or not in set.

a={1,2,4,56,89}

print(2 in a)

print("Student Bread" in a)

6.3 Combine the set

union() method print the all element but not element which are repeat

intersection() method print the element which are common on both

a={1,2,4,56,89}

b={2,89,45,56,23}

c=a.union(b)

print(c)

d=a.intersection(b)

print(d)

6.4 Set methods

add() Adds an element to the set

clear() Removes all the elements from the set

copy() Returns a copy of the set

pop() Removes an element from the set

update() Update the set with the union of this set and others

a={1,2,4,56,89}

b={2,89,45,56,23}

cd={"Ferrai","Mercedes"}

d={"B.M.W","Audi"}

a.add("Student bread")

print("add",a)

a.clear()

print("clear",a)

c=b.copy()

print("copy",c)

b.pop()

print("pop",b)

cd.update(d)

print("update",cd)

7. Dictionaries

Dictionaries are used to store data values in key:value pairs.A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values

my={

    "Fast": "quick as much as possible",

    "Student bread": "A coding tutorial website ",

    "marks": [1,2,3],

    "Student": {'Future':'citizen'},

    1:2

}

print(my['Student bread'])

print(my['Student']['Future'])

7.1 update ,add ,remove values

7.1.1 You can update values and key by following

thisdict = {

  "Student": "Bread",

  "College": "Search engine",

  "Devloped by": "3 person"

}

print(thisdict)

thisdict["Devloped by"] = "2 person"

print(thisdict)

7.1.2 You can add key and values by following

thisdict = {

  "Student": "Bread",

  "College": "Search engine",

  "Devloped by": "3 person"

}

print(thisdict)

thisdict["Hosted by"] = "1 person"

print(thisdict)

7.1.3 Remove key and values

For remove the pop method is used

thisdict = {

  "Student": "Bread",

  "College": "Search engine",

  "Devloped by": "3 person"

}

print(thisdict)

thisdict.pop("College")

print(thisdict)

8 Conditional statement

Conditional statement is used when compiler needs to choose a decision between 2 condition or more than 2

8.1 If else statement

In this program we entered one value and check the number is even or odd.

a=int(input("Enter your number: "))

if a%2==0:

    print("Number is even")

else:

    print("Number is odd")

8.2 If elif else statement

This statement is used when we have more then 2 condition then this statement is used.

a=int(input("Enter your number: "))

if a>0:

    print("Number is positive")

elif a<0:

    print("Number is negative")

else:

    print("Number is zero")

8.3 short hand if else

This type of statement is used when we must write if else in one line

a=int(input("Enter your number: "))

print("a is even") if a%2==0 else print("a is odd")

9 loops

Python supported 2 types of loop while and for loop. In loop we have to initialize write a condition and increment/decrement the variable.

9.1 While loop

With the while loop we can execute a set of statements as long as a condition is true.

i=1

while i<=5:

    print(i)

    i=i+1

9.1.1 break and continue

break used when we need to terminate a loop

i=1

while i<=5:

    if(i==3):

        break

    print(i)

    i=i+1

continue used when we just executed loop and don’t print the particular condition.

i = 0

while i < 4:

  i += 1

  if i == 3:

    continue

  print(i)

9.2 for loop

Here we use range which means it start with 1 end with 4 we write 5 but it execute 5-1 means 4.

for i in range(1,5):

    print(i)

10 Function

In function we have to define it and call it

10.1 Function with no argument

You can run a function in multiple times

*def* myFunction():

    print("function with no argument")

myFunction()

myFunction()

10.2 Function with argument

Here we gave argument and return the value.

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

    return *a*+*b*

print("Answer is",myFunction(12,12))

10.3 Default parameter value

Here we gave a default parameter if you do not give a parameter it will take it by default argument.

*def* myFunction(*a*="person"):

    print("Good morning "+*a*)

myFunction("Student")

myFunction()

11 OOPS concept

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Here we define a class and have a getdata function and gave 3 argument one is name and age.

11.1 Define a class and method.

*class* A:

*def* getdata(*self*, *name*, *age*):

*self*.name = *name*

*self*.age = *age*

p1 = A()

p1.getdata("Student bread", 36)

print(p1.name)

print(p1.age)

11.2 init method

This init method is running when object is made we have to give argument to the object and it assign the value to name and age.

*class* A:

*def* \_\_init\_\_(*self*, *name*, *age*):

*self*.name = *name*

*self*.age = *age*

p1 = A("Student bread", 36)

print(p1.name)

print(p1.age)

11.3 Self parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

You can use any word in self

*class* A:

*def* \_\_init\_\_(*ab*, *name*, *age*):

*ab*.name = *name*

*ab*.age = *age*

p1 = A("Loy", 56)

print(p1.name)

print(p1.age)

In this program we input name depart and post from the user and just print it.

*class* A:

*def* getdata(*self*, *name*,*depart*,*post*):

*self*.name=*name*

*self*.depart=*depart*

*self*.post=*post*

*def* display(*self*):

        print("Your name is "+*self*.name)

        print("Your department is "+*self*.depart)

        print("Your post is "+*self*.post)

a=input("Enter name: ")

b=input("Enter department: ")

c=input("Enter post: ")

obj=A()

obj.getdata(a,b,c)

obj.display()

11.4 Inheritances

*class* A:

*def* getdata(*self*, *name*,*depart*,*post*):

*self*.name=*name*

*self*.depart=*depart*

*self*.post=*post*

*def* display(*self*):

        print("Your name is "+*self*.name)

        print("Your department is "+*self*.depart)

        print("Your post is "+*self*.post)

*class*  B(A):

    pass

a=input("Enter name: ")

b=input("Enter department: ")

c=input("Enter post: ")

obj=B()

obj.getdata(a,b,c)

obj.display()

12. Exceptional handling

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The else block lets you execute code when there is no error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

try:

  print(x)

except:

  print("An exception occurred")

In this program we will not defined variable a

try:

  print(a)

except NameError:

  print("Variable a is not defined")

except:

  print("Something else went wrong")

In same program we will defined variable a

a=10

try:

  print(a)

except NameError:

  print("Variable a is not defined")

except:

  print("Something else went wrong")

12.1 Exception with else

Else is executed if no error is generated

a=10

try:

  print(a)

except NameError:

  print("Variable a is not defined")

else:

  print("Else part if no error")

Else is not executed when error is generated

try:

  print(a)

except NameError:

  print("Variable a is not defined")

else:

  print("Else part if no error")

12.2 Finally

Finally is executed if error is generate or not

try:

  print(a)

except NameError:

  print("Variable a is not defined")

else:

  print("Else part if no error")

finally:

    print("i am finally")