Python

Wednesday, 5 March 2025 3:23 PM

Random- is using for getting random values.
Eg:-
(0-5) will get result from randomly between 0-5

Variable :- dynamically type . using for assigning values

Data types

Int,float, string,list(mutable), tuple(inmutable), dict, booleans(T or F)

#int and floats

age = 25
Height = 5.9

#Strings

Name = "Sameer"
Greeting = "Hello, " + name

#List and tuple

Numbers = [1,2,3,4]
Names = ["sameer", "sam"]
Cordinates = (10, 20)

#Dictionary

Person = {"name": "Sameer", "age": 25}

#Booleans

Is_students = false

#Prog

```
integer_var = 10
float_var = 3.12
string_var = "Ai"
list_var = [1, 2, 3]
tuple_var = (4, 5, 6)
dict_var = {"name": "Sameer", "role": "IT"}
bool_var = True
#print
print("integer: ", integer_var)
print("float: ", float_var)
print("string: ", string_var)
print("list: ", list_var)
print("tuple: ", tuple_var)
print("dictonary: ", dict_var["role"])
print("boolean: ", bool_var)
```

Conditional Statement

If: execute the code if condition is true
elif: adds additional condition after the initial if
else: if condition false

```
#if and elif and else
num = 0
if num > 0:
    print("true")
elif num == 0:
    print("same")
else:
    print("false")
```

```
#Nested condition
age = 25
if age > 18:
    if age <30:
       print("young")
    else:
        print("old")
```

Project:

Finding prime number:

```
#task finding prime
num = int(input("Enter a number: "))
if num > 1:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            print(f"{num} is not a prime number")
            break
    else:
        print(f"{num} is a prime number")
else:
    print(f"{num} is not a prime number")
```

Loops

For loop

1. Iterates over a sequence
   i. List, tuple,

```
#loop through a lis
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
#loop with range
for i in range(5):
    print(i)
```

2. While loop
   i. Executes as long as a condition is true
   ii. This is can variable name or boolean will check until condition true

```
#while loop
count = 5
while count > 0:
    print(count)
    count -= 1
print("outside while loop")
```

3. Break
   i. Terminates the loop once condition is met

```
#break
for i in range(20):
    if i == 9:
        break
    print(i)
print("outside th break")
```

4. Continue
   i. Skip the current iteration and proceeds to the next

```
for i in range(20):
    if i == 3:
        continue
    print(i)
print("outside th break")
```

Func:-

Function is reusable block of code that perform specific task
Eg: if we're seeing any codes repeating over and over again, just create funce , and call the function. That means , change it one place, no need to everywhere

Def using for function

Basic syntax (

```
def function_name(parameter):
#code blocks
Return result  #by using the return and pass the result to calling variable
```

```
def add_numbers(a, b, c):
    return a + b + c
result =  add_numbers(5, 3, 4)
print("sum: ", result)
```

Scope and lifetime of variable.

Scope
Local Scope :- when the variable define the function within the access of function

```
def add_numbers(a, b):
  c= a + b
Return c  #So, C will know the details within inside the function
```

Global scope:- where variable define outside function , through out access whole program.

Lifetime
Local variable exist, only long as function exist

```
def greet():
    message = "hello world"
    print(message)
greet()
```

Global

Modules

Python Containing functions and variables.

Import entire module

1. Import math - So we don't need write all program

```
import math
print(math.sqrt(25))
```

2. Import specific function – import only specific function

```
from math import sqrt
print(sqrt(25))
```

3. Use aliases

Creating custom modules
We can create .py with custom function and variable in that file and we can import that by using the script #import name of the module.

Tuples

Ordered inmutable

Colours = ("red", "yellow", "blue")

For creating single item
We need put end with coma (,). Means after element

For calling tuple

Colors = ("red", "yellow", "blue")
Single_item = ("Man,")

Print(colours[0])

Tuple cannot modify after we create.

Dictionaries

Store key value pair for fastlookup
We can access the item, by using the key value

```
student = {"name": "Alice", "age":25, "grade": "A"}
print(student)
```

For accessing

Print(student["name"])

For adding

Student["subject"] = "Math"

For remove

Del student["grade"] #or
Student.pop("subject")

Iteration

For key, value in student.items():
Print(key,value)

Working with string

Concatention.

is using for combine multiple stringtogether by using + operator

First = "Hello"
Second="world
Result = first + second
Print(result)

Slicing

It'll skip

Text = "Python Programing"
Print(text[0:6])
Print(text[-11:])

F string (formatting)

Name = "same"
Age = 25
Print(f"Myname is{name} and I am {age} year old".)

Common strings

Split()
Sentence = "Python,is,fun"
Words = sentence.split(",")

Join()

New_sentence = "|".join(words)
Print(new_sentence)

Replace()

Text = "I love Java"
Updated_text = text.replace("Java", "Python")

Strip()

Removing specific one

Messy = " Hello, world "
Cleaned_text = messy.strip()
Print(cleaned_text)

Using with Statement for file management

Ensure files are properly closed after operations, even if an exception occurs

Lambda Functions

```
greeting = "hi Sameer"
def say_hello():
    print(greeting + "from inside the function") #function will call from the inside the function
say_hello()
print(greeting + " from outside the function")
    #function will call from outside the function
```

It'll call function throughout the program

List:

Ordered, mutable collection that can hold veritiy type of datatypes

```
numbers = [1, 2, 3, 4]
fruits = ["apple", "banana"]
mixed = [1, "apple", True]
```

We can access by index

```
print(numbers[3])
print(fruits[1])
print(mixed[1])
```

4
banana
apple

We can modifying list for adding by using .append

```
Fruits.append("orange")
Fruits.insert(1, "grape")
```

For remove .remove("banana") or del fruits[0]

.pop it'll remove last item

Slicing list

Get particular item from list

```
Sliced_fruits = fruits[2:4]
Print(sliced_fruits)
```

Sets

Unorders collection ofunique items

eg: -1,1,1,2,3,A,B

Numbers = {1, 2, 3, 4}
Empty_set = set()

Adding

Numbers.add(5)

Remove

Numbers.remove(2)

Set operation

Union

Set1 = {1, 2, 3}
Set2 = {3, 4, 5}

Print (set1 | set)

Will get output together and remove duplicate {1,2,3,4,5}

Intersection

Print (set1& set2)

Will intersect of both set, means will get output {3}

Difference

Print (set1 - set2)

Will check difrence

Regular Expression

Using for pattern matching
Using re module
Import.re
Common functions
Re.search(pattern, string)

```
import re
text = "contact me at 123-456-789-0"
digits = re.findall(r"\d+", text)
print(digits)
```

Re.findall(pattern, string)
Re.sub(pattern, replacement)

```
import re
text = "contact me at 123-456-789-0"
digits = re.findall(r"\d+", text)
print(digits)
updated_text = re.sub(r"\d", "x", text)
print(updated_text)
```

Reading and writing text files

Opening files

Use the build-in open() function to open a file
R | w|a|r+

Reading files

.read() | .readline()| .readlines()

```
with open("sample.txt", "r") as file:
    content = file.read()
    print(content)
```

Writing to files

.write()| .writelines()

```
with open("sample.txt", "w") as file:
    file.write("Hello, World")
    file.writelines({"Sameer", "Sam", "Moh"})
```

List comprehensions

A concise way to create lists using a single line of code

Eg:-
#create a list of squares

```
squares = [x**2 for x in range(50)]
print(squares)
```

#Filter Even numbers

Map()

Applies a function to each item in an iterable

Filter()

Filters items based on condition

Reduce()

Reduce an iterable to a single value

```
squares = [x**2 for x in range(50)]
#print(squares)
evens = [x for x in range(100) if x % 2 != 0]
#print(evens)
add = lambda x, y: x+ y
#print(add(3,5))
numbers = [1, 2, 3, 4]
squares = map(lambda x: x**2, numbers)
#print(list(squares))
numbers = [1, 2, 3, 4]
evenList = filter(lambda x: x % 2 == 0, numbers)
#print(list(evenList))
from functools import reduce
numbers = [1, 2, 3, 4]
product = reduce(lambda x,y: x * y, numbers)
print(product)
```

```
squares = [x**2 for x in range(50)]
#print(squares)
evens = [x for x in range(100) if x % 2 != 0]
print(evens)
```

Os mudule

Os.remove

Remove the file

Sys module

Sys.argv