Introduction to Python - Recap

Numbers: Integers and Floats

Integers: Whole numbers

Floats: Numbers with decimal point

Complex Numbers: Numbers with real and imaginary parts.

In [ ]:

Note:

print(text) displays the the given text on the monitor.

variable=input(prompt) read the text from keyboard

type(object) returns the type(class) of the given object

In [1]:
```
# Display the type of different number objects
a=10
b=20.4
c=2+8j
print(type(a))
print(type(b))
print(type(c))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

Arithmetic Operators

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer Division
%	Remainder
**	Exponent

In [2]:
```
# Take any two numbers and display the results ofter performing all basic arithmetic operaions on them.
a = 5
b = 2
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a//b)
print(a%b)
print(a**b)
```

```
7
3
10
2.5
2
1
25
```

Strings

Surrounded in either single quotes or double quotes.

In [3]:
```
# Create two string objects one with double quotes and other with single quotes. Also display the type of both string objects.
s1 = "First String"
s2 = 'Second String'
print(type(s1))
print(type(s2))
```

```
<class 'str'>
<class 'str'>
```

Sring Concatenation:

+ operator when used between stings acts as concatenation operator to join them.

In [4]:
```
# Take first name and last name from user then display full name.
fn = input("Enter first name: ")
ln = input("Enter last name: ")
fullname = fn + " " + ln
print(fullname)
```

```
Enter first name: Ratna
Enter last name: Kishor
Ratna Kishor
```

String Indexing :

Index of first character is 0

Negative Indexing: Index of the last character is -1

In [5]:
```
# Create a string object with string literal "SVEC" and display it's characters using both positive & negative indexes.
clg = "SVEC"
print(clg[0])
print(clg[1])
print(clg[2])
print(clg[3])
print(clg[-1])
print(clg[-2])
print(clg[-3])
print(clg[-4])
```

```
S
V
E
C
C
E
V
S
```

String Slicing

new_string = string[start, end, step]

--> Note that end index is exclusive in the range

--> step is optional and its default value is 1

In [6]:
```
# Extract the word love from a string "I love my country"
s = "I love my contry"
print(s[2:6])
```

```
love
```

In [8]:
```
# Extract alternate characters from the begining in a string "SAVBECC"
s = "SAVBECC"
print(s[: : 2])
```

```
SVEC
```

In [9]:
```
# Reverse the given string
print(s[: : -1])
```

```
CCEBVAS
```

String case-conversion methods

new_string = string.lower(): lower() returns a new string with all lowercase letters in the original string

new_string = string.upper(): upper() returns a new string with all uppercase letters in the original string

In [11]:
```
print("AaBaCc".lower())
print("AaBaCc".upper())
```

```
aabacc
AABACC
```

String splitting

list_of_strings = string.split(delimiter): split() method returs a list of strings by splitting the original string upon the delimiter.

--> deliiter is is optional argument and has white space(space, tab or newline) as defalt value.

In [12]:
```
# Split the strings "Hi how are you", "The#king#is#back" and "I--love--my--india" with appropriate delimiters
print("Hi how are you".split())
print("The#king#is#back".split("#"))
print("I--love--my--india".split("--"))
```

```
['Hi', 'how', 'are', 'you']
['The', 'king', 'is', 'back']
['I', 'love', 'my', 'india']
```

Strings joining

new_string = seperator.join(list_of_strings): Returns a lnew string by joining all the string in the given list with specified separator.

In [13]:
```
# Join the list of strings ["Chiru", "Balayya", "Nag", "Venky"] with the delimiters " " and "_".

print(" ".join(["Chiru", "Balayya", "Nag", "Venky"]))
print("_".join(["Chiru", "Balayya", "Nag", "Venky"]))
```

```
Chiru Balayya Nag Venky
Chiru_Balayya_Nag_Venky
```

str(), int() and float() functions

str() function converts the given argument as a string

int() function converts the given argument as an integer

float() function converts the given argument as a float

In [14]:
```
# Create an integer object and reverse it without using loops

a = 123
ra = int(str(a)[: : -1])
print(ra)
```

```
321
```

Boolean Data Type

Values: True & False

Comparison Operators

Also called Relational Operators

To compare two values

Evaluates to Boolean value

Operator	Operation
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than equal to

In [ ]:

Logical Operators

To compare Boolean values or expressions.

To evaluate multiple comparisons at a time

Results in Boolean value

Operator	Operation
and	Results in True if all values are True
or	Results in True even if one value is True
not	Unary operator which results is opposite Boolean value

In [ ]:

Flow Control Statements

if statements

```
if condition:
    statements_block
```

```
if condition:
    statement_block1
else:
    statements_block2
```

```
if condition1:
    statement_block1
elif condition2:
    statements_block2
elif condition3:
    statements_block3
else:
    statements_block4
```

In [18]:
```
# Read the marks and display pass if greater than or equal to 35
marks = int(input("Enter marks: "))
if marks >= 35:
    print("Pass")
```

```
Enter marks: 34
```

In [19]:
```
# Read the marks and display pass or fail
marks = int(input("Enter marks: "))
if marks >= 35:
    print("Pass")
else:
    print("Fail")
```

```
Enter marks: 34
Fail
```

In [23]:
```
# Read the marks and display Grade the student got
marks = int(input("Enter marks: "))

if marks >= 60:
    print("Frst")
elif marks >= 50:
    print("Second")
elif marks >= 35:
    print("Third")
else:
    print("Fail")
```

```
Enter marks: 34
Fail
```

while Loop

```
while condition:
    statements_block
```

In [25]:
```
# Find out sum of squares first N numbers

N = int(input("Enter N value: "))

i = 1

while i <= N:
    print(i)
    i+=1
```

```
Enter N value: 4
1
2
3
4
```

for Loop

```
for variable in iterable:
    statements_block
```

In [ ]:

range() Function

Reurns a range object with sequence of numbers

range(stop): Returns a range object with a sequence of numbers from 0 to stop-1.

range(start, stop): Returns a range object with a sequence of numbers from start to stop-1.

range(start, stop, step): Retuns a range object with numbers from start with the given step upto stop (exclusive).

In [ ]:

In [26]:
```
# Display a message N times.
N = int(input("Enter N value: "))

for _ in range(N):
    print("Hi")
```

```
Enter N value: 3
Hi
Hi
Hi
```

In [28]:
```
# Sum of squares of first N numbers
N = int(input("Enter N value: "))

sos = 0

for i in range(1, N+1):
    sos += i ** 2

print(sos)
```

```
Enter N value: 3
14
```

In [29]:
```
# Sum of squares of the numbers from x to y
x = int(input("Enter lower limit: "))
y = int(input("Enter upper limit: "))
sos = 0
for i in range(x, y+1):
    sos += i ** 2
print(sos)
```

```
Enter lower limit: 2
Enter upper limit: 4
29
```

In [30]:
```
# Display all the even numbers between x and y.
x = int(input("Enter lower limit: "))
y = int(input("Enter upper limit: "))
for i in range(x, y+1):
    if i%2 == 0:
        print(i, end = " ")
```

```
Enter lower limit: 2
Enter upper limit: 10
2 4 6 8 10
```

In [32]:
```
# Display all odd numbers between 0 and N without using any arithmetic/logical operation
N = int(input("Enter N value: "))
for  i in range(1, N+1, 2):
    print(i, end =  " ")
```

```
Enter N value: 10
1 3 5 7 9
```

Functions

Subprogram which performs a specific task

Types:

1. Built-in Functions

print(), input(), len(), int(), float(), str(), type(), range() . . . .

2. User-defined Functions

$$\textbf{def}\ functinname(formal\ parameters):$$

functional *statements*
&
return statements

In [ ]:

In [33]:
```
# Implement a function to display welcome message
def invite(name):
    print("Hi", name, "welcome to SVEC")

invite("Ratnakishor")
invite("Lasritha")
```

```
Hi Ratnakishor welcome to SVEC
Hi Lasritha welcome to SVEC
```

In [34]:
```
# Implement a function to check password
def check(pwd):
    if pwd == "1234":
        print("Correct password")
    else:
        print("Wrong password")

check("king")
check("1234")
```

```
Wrong password
Correct password
```

In [35]:
```
# Implement a function to return the factorial of the given number
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n-1)

print(fact(4))
print(fact(0))
```

```
24
1
```

Lists

Ordered Sequence of items.

List is a comma separated items between opening and closing sqare brackets.

Individual items of a list can be accessed using indexing.

List are mutable objects in python

In [36]:
```
# Create a list object with different types of objects and show that it is mutable

list1 = ["Ratna", "CSE", [45, 67, 34]]
print(list1)
list1[-1][1] = 57
print(list1)
```

```
['Ratna', 'CSE', [45, 67, 34]]
['Ratna', 'CSE', [45, 57, 34]]
```

A list can have other list as its item

In [ ]:
```
# Create a list object with other list as one of the elements and access its elements usig indexing
```

List Slicing

In [ ]:
```

```

Length and Concatenation of Lists

In [ ]:
```

```

Replication

In [37]:
```
# Create a list of 100 zeros
lr = [0]*100
print(lr)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

List Methods

index = list.index(item)

- Returns the index of the item in the list.
- If the item not present python produces a ValueError

In [ ]:
```

```

list.append(item)

- Appends the argument at the end of the list.

In [ ]:

list.insert(index, item)

- Inserts an item at the given index in the list

In [ ]:

list.remove(item)

- Removes the first occurance of an item from the list
- If the item does not exist in the list we will get ValueError error.

In [ ]:

list.sort()

- Sorts the list in place

list.sort(reverse = True)

- Sorts the list in the reverse order

  Note: Python can't sort the list that has both numbers and strings.

In [ ]:

list.reverse()

- Reverse the order of the list items

In [ ]:

list.copy()

- Creates a duplicate copy of the list

In [ ]:

count = list.count(item)

- Return the number of occurances of given item

In [ ]:

Tuples

Same as Lists with the following differences

1. Items are enclosed in parenthesis.
2. Tuples are immutable objects

In [39]:
```
# Create a tuple object and show that it is immutable
tuple1 = (10, 20, 30)
print(tuple1)
tuuple[1] = 100
```

```
(10, 20, 30)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-39-67d5d18e5539> in <module>
      2 tuple1 = (10, 20, 30)
      3 print(tuple1)
----> 4 tuuple[1] = 100

NameError: name 'tuuple' is not defined
```

Methods

count = tuple.count(item)

index = tuple.index(item)

In [ ]:

Dictionary

- Mutable collection of key - value pairs enclosed in braces, { }.
- {key1:value1, key2:value2, ...}
- Values can be accessed though their keys
- Dictionaries are unordered --> Order of the keys does not matter to decide whether two dictionaries are same

In [ ]:

Methods

- keys( ): Returns a dict_keys object with all keys in the dictionary
- values( ): Returns a dict_values object with all values in the dictionary
- items( ): Returns a dict_items object with all key-value pairs in the dictionay

In [ ]: