PYTHON

Why Python :

1. Interpreter — byte Code
2. ~~Platform Independent~~ Cross platform
3. Easy readability, easy to learn
4. Dynamic Typing
5. Less complex syntax
6. High Level – object oriented Language
7. Open Source
8. Powerful Devlopment Libraries
9. Applications in m/c, AI, Data Science, web development.

Dynamically typed — you don't need to specify the variable ~~time~~ type at the time of variable declaration. The types are specified at the runtime based on the assigned value due to its dynamically typed feature.

Python popular Modules & libraries:

Numpy
Pandas
Matplotlib
Math
Scikit-learn, Tensorflow = OpenCV

Download from:

python.org

Cross Platform: A python program is first compiled to an intermediate platform independent byte code. The virtual machine inside the interpreter then executes the byte code. This behaviour makes Python a cross-platform language, and thus a Python program can be easily ported from one OS platform to other. When a C++ prog. source code Hello.cpp is compiled on Linux, it can be only run on any other computer with Linux.

Easy to learn: few Keywords, simple structure, clear syntax.

It is Case Sensitive Language:

Input Output:

```
a = int(input("Enter value of a"))
b = int(input("Enter value of b"))

print("The no you entered for a is", a)
print("The no you entered for b is", b)
```

input() is used to take input from the user. return type of input is string only, so we have to convert it explicitly to the type of data that we require.

print() is used to display the output.

→
```
a = 10
print(a)
```

f strings

→
```
a = 10
print(f"The value of a is: {a}")
```

output: The value of a is: 10

→
```
a = 10
b = 5
print(f"{a} + {b} = {a+b}")
```

10 + 5 = 15

Identifiers :

A python identifier is a name used to identify a variable, function, class, module.

An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9)

Python class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private identifier.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

classmate
Date
Page

Python Reserved Words :

and	as	assert
break	class	continue
def	del	elif
else	except	false
finally	for	from
global	if	import
in	is	lambda
None	nonlocal	not
or	pass	raise
return	True	try
while	with	yield.

No Braces

It provides no braces to indicate blocks of code for class, function definitions or flow control.

The no. of spaces in the indentation is variable, but all statements within the block must be indented the same amount.

```
total = item_one +
        item_two +
        item_three
```

→ line Continuation characters.

statements within [], {}, or () brackets do not need to use the line continuation character.

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotations in python:

```
word = 'word'
print (word)
```

```
sentence = "This is a sentence."
print (sentence)
```

```
paragraph = """ This is a paragraph. It is
made up of multiple lines
and sentence. """
print (paragraph)
```

Comments :

```
# This is a comment
# I said that already.
''' This is a multi line
comment. '''
```

Multiple statement Groups as Suites :

A group of individual statements, which make a single code block are called suites in Python.

Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin with the statement (with the keyword) and terminate with a colon and are followed by one or more lines which make up the suite.

```
if expression :
    suite
elif expression:
    suite
else :
    suite
```

Python Variables :

Python variables are the reserved memory locations used to store values with in a Python program. This means that when you create a variable you reserve some space in the memory.

Based on the data type of a variable, python interpreter allocates memory and decides what can be stored in the reserved memory.

Therefore, by assigning different data types to Python variables, you can store integers, decimals or characters in these variables.

id() function give the address of the ~~variable~~ object where it is ~~stored~~

```
>>> "may"
may
id("may")
```

21622892

A Python variable is a symbolic name that is a reference or pointer to an object.

```
>>> month = "May"
>>> age = 18
>>> id(month)
2167823
>>> id(age)
1407296.
```

So id of May and Month are same.

Creating Python Variable:

Python variable do not need explicit declaration to reserve memory space or you can say to create a variable.
A Python variable is created automatically when you assign a value to it.

```
Counter = 100 # Creates int variable
miles = 1000.0 # Creates float
name = "Zara Ali" # "string"
```

print (counter)
print (miles)
print (name)

output → 100
100.0
Zara Ali

Deleting Python Variables

You can delete a single object or multiple objects by using the del statement!

```
del var
del var_a, var_b
Counter = 100
print (counter)
del counter
print (counter)
```

output → 100
NameError: name 'counter' is not defined.

Getting Type of a variable:

```
x = "Zara"
y = 10
z = 10.10
print (type(x))
print (type(y))
print (type(z))
```

output → ~~string~~ integer
<class 'str'>
<class 'int'>
<class 'float'>

Casting Python Variables

```
x = str(10)
y = int(10)
z = float(10)

print("x=", x)
print("y=", y)
print("z=", z)
```

Output: ~~x = '10'~~ x = 10
y = 10
z = 10.0

Case Sensitive:

```
age = 10
Age = 20
print(age)
print(Age)
```

Python Variables Multiple Assignment:

```
>>> a=b=c=10
>>> print(a, b, c)
10 10 10
```

```
a
>> a = 10
>> b = 20
>> c = 30
```

```
>>> a, b, c = 10, 20, 30
>>> print (a, b, c)
10  20  30

a = b = c = 100
print (a)
print (b)
print (c)

a, b, c = 1, 2, "Zara Ali"
print (a)
print (b)
print (c)

1
2
Zara Ali
```

Python Variables — Naming Convention

```
x = 10
y = 20.5
name = "abc"
print (5)
print (x)
print (y)
print (name)
print("The no' is", y)
```

classmate
Date
Page

Programs

1. Area of rectangle
2. Perimeter of rectangle
3. Sum & product of two no.s
4. print name and address of a person.
5. type of variable
6. id of a variable
7. Comments
8. typecast
9. del variables
10. program to print wether a person is minor, adult or senior citizen
11.

Datatypes

(1) Numeric — int
float
Complex

(2) String

(3) Sequence → Lists
→ ~~Dictionary~~ tuple
range

(4) Binary → bytes
bytearray

& (5) Dictionary

(6) Boolean

(1) Numeric :

```
x = 5      # int datatype
x = 10.5   # float
y = 10 + 5j  # complex.
```

(2) String :

```
name = 'xyz' # string
       "xyz"
       """xyz"""
```

```
type (name)
<class 'str'>
```

```
str = 'Hello World'
print ( str)
print ( str[0])
print (str [0:5])
print ( str * 2)  # print string 2 times.
```

③ Sequence
- → Lists
- → Tuple
- → Range
- → Dictionaries

List :

x = [123, 'str', 30.5, 5+6j]

tuple :

x = (123, 'str', 30.5, 5+6j)

difference :

① [ ] — Lists

( ) → tuple

② Mutable → Lists

Immutable → tuple

eg: list = ['str', 123, 10.5]

tuple = (1234, 11.6, 'name')

list[2] = 1000 # valid syntax

tuple[2] = 3 # Invalid.

tuple is a read only list.

```
print(list[0:3])
print(list[2:3])

print(tuple[0])
print(tuple[0:3])
```

Range → immutable sequence of numbers which is typically used to iterate through a specific number of items.

range(start, stop, step)

- start → starting position
- stop → ending position
- step → specify increment

operators :

① Arithmetic

+ , − , * , / , % , //

↳ floor division

==

>=

<=

>

<

Control statements :

if elif else

nested if

```
if x>15 :
    print("no. is greater than 15")
elif x>5 :
    print("x is greater than 5 but
           not greater than 15)
else :
    print ('x is 5 or less')
```

```
x = 10
y = 20
if x > 5:
    if y > 15:
        print("Both condition are true")
    else:
        print("x > 5 but y <= 15")
else:
    print('x <= 5')
```

(1) age (adult, minor, senior citizen)

(2) no. is +ve, -ve even or negative

Sequence datatype:

dictionary:

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

A dictionary in Python is a mutable, unordered collection with key-value pairs separated by colon.

Accessing Values:

print (my_dict ["name"])

age = my_dict ['age']

Control statements & loops :

1. if statement
2. if elif
3. nested if

Loops : while loop
for loop

1. program to print Hello World
2. Prog. to add two no:s
3. Prog. to find square root
3. Prog. to Calculate area of triangle
4. Prog. to Calculate remainder of two no:s using modulo operator
5. Prog. to demonstrate.
   - (I) input ()
   - (II) print ()
   - (III) 'sep' attribute
   - (IV) 'end' attribute
   - (V) replacement operator ({})
   - ~~(VI)~~
6. Program to get a substring from a given string starting from 3rd character to 5th character.

print("Welcome", "to", "Codedamn", sep=",")

print("Welcome","to","Codedamn",sep="|t")

print("Welcome","to","Codedamn",sep="-")

Output:

Welcome, to, Codedamn
Welcome to Codedamn
Welcome-to-Codedamn

---

print("Welcome","to","Codedamn",end="@")

print("Welcome","to","Codedamn",end="***")

print("Welcome","to","Codedamn",end=" ")

Output:

Welcome to Codedamn @ Welcome to Codedamn *** welcome to Codedamn

sep — Specifies the string to use as a separator between the objects being printed. The default value is ' '.

end — It specifies the string to use as a terminator after the objects are printed. The defaut value is '\n'.

Control Loops

Type Casting :

Implicit

①
```
a = 10
b = 10.5
c = a + b
```

a is of int type

b is of float.

While adding, a gets upgraded to b.

②
```
a = True
b = 10.5
c = a + b
print (c)
```

Boolean gets upgraded to integer, then float.

Explicit type casting:

```
x = int (10.0)        output
print(x)               10
type (x)
```

= <class 'int'>

```
x = float (100)
print (x)
```

output
↳ 100.0

Data type Conversion function:

1. int ( )
2. long ( )
3. float ( )
4. complex ( )
5. str ( )
6. repr ( )
7. eval ( )
8. tuple ( )
9. list ( )
10. set ( )
11. dict ( )
12. chr ( )
13. hex ( )
14. oct ( )
15.

classmate
Date
Page

Eg: of dictionary data type:

```
capitals = {"Maharashtra": "Mumbai",
            "Gujrat": "gandhinagar"}

numbers = {10: "Ten", 20: "Twenty",
           30: "Thirty"}

marks = {"Sarika": 67 "Sumtiaz": 88,
         "Laxman": 91}

sports_player =
        {"name" : "Sachin Tendulkar",
         "Age" : 48,
         "sport" : "Cricket"
        }
```

Control statements :

1. if statement
2. for loop
3. while loop

1. if statement → if statement
   → if ... else
   → nested if statements

if statement :

```
var a = 100
if {var == 100}:
    print("value of exp. is 100")
    print("good bye")
```

if - ... else :

```
var = 100
if {var == 100}:
    print("value of var is equal to
           100")
else
    print("value of var is not
           equal to 100")
```

nested if statements :

```
x = 15
y = 10
if x > 10:
    print("x is greater than 10")
    if y > 5:
        print("y is also greater than 5")
    else:
        print("y is not greater than 5")
else:
    print("x is not greater than 10")
```

```
var = 100
if var == 100:
    print("The number is = to 100")
    if var % 2 == 0:
        print("The no. is even")
    else:
        print("The no. is odd")
elif var == 0:
    print("The no. is zero")
else:
    print("The given no. is negative")
```

match Case Statements :

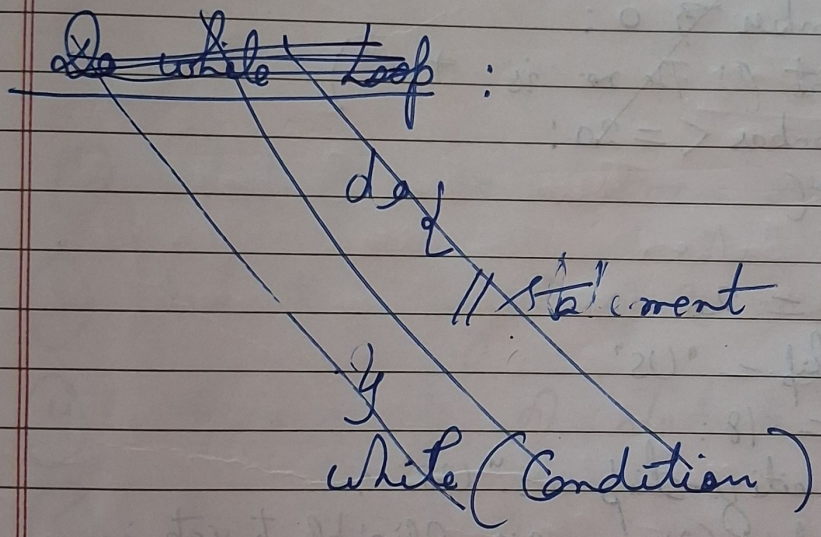
```
match variable_name:
    case 'pattern 1' : statement 1
    case 'pattern 2' : statement 2
    .....
    case 'pattern 3' : statement 3
```

While loop:

①
```
count = 0
while count < 5:
    count += 1
    print (count)
print ("End of while loop")
```

②
```
age = 28
while age > 19:
    print ('Infinite Loop')
```

~~Do while loop~~ :

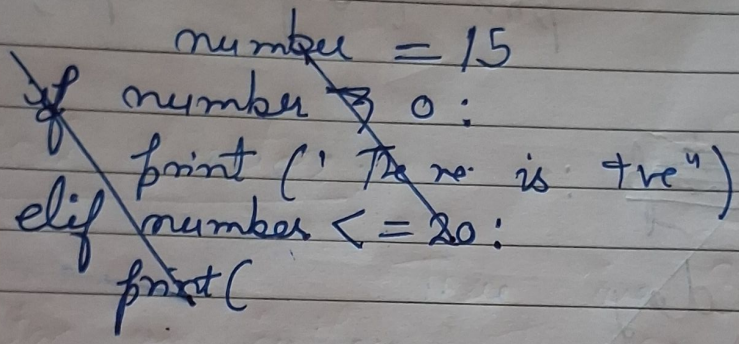


While else Loop:

```
while (condition):
    # Code to execute while
      the condition is true
else:
    # Code to execute
```

①
```
number = 5
if number > 0:
    print('The no. is +ve')
elif number == 0:
    print('The no. is zero')
else:
    print('The no. is -ve')
```

② nested if :



```
number = 15
if number > 0:
    print("The no. is +ve")
elif number <= 20:
    print(
```

①
```
age = 20
citizenship = "US"
if age >= 18:
    if citizenship == "US":
        print("you are eligible to vote in US")
    else:
        print("You are not eligible to vote in US")
else:
    print("You are too young to vote")
```

(2)
```
username = "admin"
password = "admin123"
if username == "admin":
    if password == "admin123":
        print ("Access granted")
    else:
        print ("Incorrect password")
else:
    print ("User not found").
```

Programs:

(1) Check if a no. is positive or Negative

(2) Check if a no. is Even or Odd.

(3) Find the largest of 2 no's.

(4) Find grade of students

(5) Find largest of 3 no's

(6) Write a program to print no's from 1 to N using while loop.

(7) Reverse a no. using while loop.