Date: _____

PYTHON:

Intro / history:

- It's a general purpose high level programming language.
- It can be used for: console apps, Desktop application, web app, mobile app, Machine learning IOT applications.
- Very simple and straight forward syntax. ~~It can be your first programming language.~~
- Python is case sensitive language.
- Use variables without declaration.
- Intepreted language.
- Emphasis on code readability.
- Platform independent.

Library for:

- GUI.
- Multimedia
- Networking
- Automation
- Web frameworks
- Databases
- Documentation
- Administration System / etc.

Beginning: The idea of python started in 1980s
- Real implementation started in 1989
- Finally published in 27 feb 1991.
- Open source project

Python 2	python 3
• Division operator work like C.	• Division operator.
• Print "hello"	• print ("hello")
• By default str type is ASSCII,	• Str type is unicode.
• range() & xrange()	• only range() function.
• Exception handling	• as keyword is required

Date:

Print (Hello) error
Print ("Hello")

A = 4+5
A = 4*5
A = 4/5 = 0.8 / A = 5/5 = 1.0 float answers.
A = 4//4 = 1 / A = 4//5 = 0 // used for integer type value.
A = 4**2 = 16 double ** is used as power.
A = 4*2 = 8
Print (A).

Comments:

Hash for single line comment.
""" *** double quotes asterics is used for Multiple lines of comments *** """

Print ("Khadija") output: Khadija
Print ("Tahira") Tahira

Print ("Khadija", end=" ") Khadija Tahira
Print ("Tahira".)

Print ("Khadija", "Tahira", end=" ") Khadija Tahira

Print ("C:\n array") \n next line: C
array.

Print ("C:\narray)

C:\narray.

Date:

print(" Khadija is \n a good programmer \t 1 ")

Output: Khadija is
a good programmer ⟷ 1.
(\t is tab.)

Variables: ⇒ container:

x [5]

Var1 = Hello! Khadija. , Var2 = 4 , Var3 = 36.7

print (var1)
print (type (var1))
print (type (var2))
print (type (var3))
print (var2 + var3)
print (var1 + var2) => error bcz of diff data types.).

outputs:
<class 'str'>
<class 'int'>
<class 'float'>
40.7

Var1 = "Hello" , Var2 = "32"

Same data type

+ is used to add string but, for other data types

Print (var1 + var2)

Hello 32

Typecasting: is the process to change datatypes.

Var1 = "54"
Var2 = "32"
print (int(var1) + int (var2))

Output.
86

functions:

""" str() , int() , float() """

print (100 * "Hello\n") => 100 print.

Print (100 * str (int(var1) + int(var2)))
86 100 times Print.

Date:

How to take input from user:

```
Print ("Enter any number")
inpnum = input()

print ("You entered", inpnum)
print ("  "  "  ", inpnum +10)  X error
```

bcz inpnum by default string data type.
First we convert input type:

```
print ("You entered", int (inpnum) +10).
```

Built in data types:

Text type	Str
Numeric type	int, float, complex.
Boolean type	Bool
Binary type	byte, bytearray.

```
x = "Hello word"      str
x = 20                int
x = 20.5              float
x = 1j                complex
x = ["apple", "banana"]   list
x = ("apple", "banana")   tuple
x = range (6)         range
x = {"name": "Khadija", "age": 36}   dict
x = True.             bool.
x = frozenset ({"Khadija", "maira"})   frozenset.
Rang(int(input())
```

Take input from the user

Date:

Casting:

```
x = int(1)          # x will be 1
x = int(2.8)        # "  "  "  2
x = int("3")        # "  "  "  3

x = float(1)        # x will be 1.0
x = float(2.8)      # "  "  "  2.8
x = float("4.2")    # "  "  "  4.2

x = str("S1")       → 'S1'      single quotes.
x = str(2)          → '2'
x = str(3.0)        → '3.0'
```

Booleans:

```
print(bool("Hello"))
print(bool(15))
```
] Trues.

```
x = "Hello"    y = 15
Print(bool(x))
print(bool(y))
```
] Trues.

Operators:

is	Return true if var. are same obj	x is y
is not	" " if var. are not same obj	x is not y

```
x = ["apples", "banana"].
z = x.
print(x is z)
```
or `print(x == z)`

```
Print(x is not y)
```
or. `Print(x != y)`

Date:

Casting:

```
x = int(1)        # x will be 1
x = int(2.8)      # "  "  "  2
x = int("3")      # "  "  "  3

x = float(1)      # x will be 1.0
x = float(2.8)    # "  "  "  2.8
x = float("4.2")  # "  "  "  4.2

x = str("S1")  → 'S1'     Single quotes.
x = str(2)     → '2'
x = str(3.0)   → '3.0'
```

Booleans:

```
print(bool("Hello"))
print(bool(15))
```
] Trues.

```
x = "Hello"   y = 15
Print(bool(x))
print(bool(y))
```
] Trues.

Operators:

is	Return true if var. are same obj	x is y
is not	" " if var. are not same obj	x is not y

```
x = ["apples", "banana"].
z = x.
print(x is z)
```
or
```
print(x == z)

Print(x is not y)
```
or.
```
Print(x != y)
```

Date:

Membership operators:

in	return True if a sequence with the specified value is present in object	x in y
not in	" " " is not present "	x not in y.

if x in y:

Python lists:

⇒ Mixed list ⇒ numeric / string / float

List: is a collection of ordered and ~~unchangeable~~. Allows duplicate members. Lists are written in brackets.

Mutable → changeable.

e.g:
```
thisList = ["Khadija", "Maira"]
print(thislist)
```

Access Items: print(thislist[1]) → (Maira)

Loop through a list:

```
for x in thislist:
    print(x)
```

Print all items one by one.

```
Print(numbers[::])
```

```
if "Khadija" in thislist:
    print("Yes")
```

```
thislist.clear()
thislist.append(7)
or thislist.insert(1, 2)
= {1, 2, 3, 4}
```

```
Thislist.sort()
thislist.reverse()
Print(thislist)
```

~~print(thislist.max())~~

Class name in upper case.
variable, object, lists → lower case.

Date:

List Methods:

~~add()~~	
append()	Adds an element at end of list
Clear()	Removes all elements from list
copy()	Returns the copy of list
count()	Returns the no. of element with specified value.
extend()	Add element of a list, to the end of current list.
index()	Returns the index of first element with specified value.
~~ins~~	
insert()	Adds an element at specified position.
↳ insert(index, item)	
Pop()	Removes the element
remove()	Removes the item
reverse()	Reverses the order of list
sort()	Sorts the list

Tuple: is a collection which is ordered and unchangeable, written in round brackets. (Immutable ← can't change)

```
thistuple = ("Khadija", "Bakhtawar")
print (thistuple)
```

Change tuple value:

```
x = ("ab", "bc", "ca")
y = list(x)
y[1] = "item"
x = tuple(y)
Print(x)
```

loop:

```
for x in this_tuple.
    Print x.
```

Length:

```
print( len(thistuple))
```

Date:

Tuple methods:

count( ) — Returns the no. 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.

Sets: is a collection which is unordered and unindexed. written with curly brackets.

```
thisset = {"vivo", "infinix", "redmi"}
print (thisset)
```

```
S1 = thisset.union({1,2,3})
```

S1 intersection

```
thisset.add("iphone")
thisset.update(["vivo", "iphone", "redmi"])
thisset.remove("vivo") → error → not remove.
```

Dictionary: collection of unordered, changeable & indexed, written curly brackets & they have keys and values.

```
thisdict = {"brand": "Ford", "year": 1969}.
print(thisdict)
```

Access → `x = thisdict["year"]`

else use get `x = thisdict.get("year").`

`del thisdict[brand]`

Date:

Change values:

thisdict["year"] = 2018

loop:

for x in thisdict:
for x in thisdict.values():
for x, y in thisdict.items():
↳ print(x, y)

Adding Item

thisdict["color"] = "red"

copy func:

d3 = dthisdict.copy()
del d3["Brand"]
print(thisdict)

↳ Print all data.

Nested dicts:

Child 1 = {"name": "Fatima", "year": 2004}
Child 2 = {"...": "...", "...": "..."}
Child 3 = {- - - -}
myfamily = {"child 1": child1, "child 2": child2,
"child 3": child3}

To add variables:

y = "abc" , x = 10
x = ~~x int~~ int(x)

z = x + " " + y → Print = 10 abc

Date:

If ... Else:

```
if a > b :
    ____
else :
    ____
elif b > a :
    ____
else :
    ____
```

required 1 tab means 4 spaces at the beginning of so cout.

```
if a > b and c > a :
if a > b or a > c
```

```
list = [5, 7, 3]
if 5 in list :
    print ("Yes its in list").
```

Nested if:

```
if x > 40
    ____
    if x > 20 :
        ____
    else :
        ____
```

While loop:

```
i = 1
While i < 6 :
    print(i)
    i += 1
```

break Statement:

```
i = 1
While i < 6 :
    print (i)
    if i == 3 :
        break
    i += 1
```

continue Statement:

```
i = 0
While i < 6 :
    i += 1
    if i == 3 :
        continue
    print (i)
```

continue keyword jb bi aye ga to usi wqt condition dobara keyn gy.

Date:

Else Statement:

```
i = 1
While i < 6
  print(i)
  i += 1
else:
print ("i is no longer")
```

Table with While:

```
i = 1
print("Enter any number for table")
inpa = input()
while i <= 10:
print (int(inpa), "*", i, "=", int(inpa) * i)
    i += 1
```

(Switches is Not use in python)

For loop:

```
list 1 = [["Harry", 1], ["Harry", 2]]
for N1 N2 in list1
  print (N1, N2)
```

output:

Harry 1

Harry 2

for name in list, dec (=> variable name)

Print ( ).

Date:

String Slicing

```
myste = "Khadija is a good programmer"
print (len(myste))
print (myste[0:29])  (print full line)
print (myste[78]) → error wrong way
print (myste[0:78])  print full string.

Print (myste[0:8:2])  Kaia → output
print (myste[0:])  → print full string.
print (myste[:8])  → Khadija
print (myste[:])
print (myste[::])  → Print full string.

Print (myste[0:8:1]  → Khadija
print (myste[0:29:559] → K

Print (myste[-1:0])  no error.
print (myste[-4:-2])  mm
```

-7	-6	-5	-4	-3	-2	-1
K	H	A	D	I	J	A
0	1	2	3	4	5	6

```
print (myste[::-1]) → it reverses all string.
```

like (reammargorp doog a si ajidaKK).

```
Print (myste[::-2] → firstly reverse all characters
                     by leaving 1 character.
```

Date:

Function :-

print (mystr.isalnum()) → False ⇒ boolean

↳ if we remove spaces b/w string.

↳ print(mystr.isalnum()) → True.

print(mystr.isalpha()) → False bcz is not it alpha or numeric.

print (mystr.endswith(boy)) → True.

print (mystr.count("a")) → 4 bcz 4 a's are in string.

print (mystr.capitalize())

print (mystr.find("is")) → Tells us index number where "is" is found 8

print (mystr.lower()) → all string in lower case.

print (mystr.upper())

print (mystr.replace("good", "intelligent")).

For swapping two number.

a = 5

b = 6

a, b = b, a

print (a, b) → 6 5

Date:

output :-

1- 12345

2- 992345

Functions & Docstings :

```
a = 9
b = 8
c = sum((a,b))
print(c)
```

→ Built in function :
⇒ output = 17.

```
def function1():
    print("Hello you are in function 1")
print(function1())
```

→ output :

Hello. you are in function 1.
None.

function1() → output :
Hello you are in function 1

Date:

```
def function1(a,b):
    print("Hello you are in function 1", a+b)

def function(a,b):
    """ This is a function which will calculate average of two numbers """
        average := (a+b)/2
    : # print average.
        return average

# v=function2(5,7)
# print(v)
print(function2.__doc__)
```

__doc__ is used to print docstring that is in """ """ triple inverted commas.

Output :-

This is a function which will calculate average of two numbers.

Date:

To print calendar

```
Import calendar

cal= calendar.month (2016,1)
print (cal)
```

OR: print (calendar.isleap (2016))
↳ output may be T/F.

To sort list

```
list 1 =[ 1, 5, 6, 7, 8]
print ( list1.sort.~~list~~( ) )
```
→ [1,5,6,7,8]

print descending order.

```
print ( list1.sort(reverse))
```
→ [8,7,6,5,1].

↳ list1 __ dict __

To make file and store data :-

```
# save data in files
    with open ("c://data //cal.txt","w") as f:
        f.write (cal)
```

Reading and writing files:- Date:

Writing:-

```
f = open("C:\\data\\funny.txt","w")
f = write("I Love Python")
f.close()
```

w → overwrite the file.
a → append.

Read file line by line:

```
f = open("C:\\data\\funny.txt","r")
print(f.read())
f.close()
```

Another way:-

```
f = open("C:\\data\\funny.txt","r")
for line in f:
    print(line)

f.close()
```

To get list of words in file:-

```
f = open("C:\\data\\funny.txt","r")
for line in f:
    tokens = line.split(' ')
    print(str(tokens))

f.close()
```

split() will split the string using input character & return a list of words.

Date:

f_out

```
f = open("C:\\data\\funny.txt","r")
f_out = open("C:\\data\\funny_wc.txt","w")
for line in f:
    tokens = line.split(' ')
    f_out.write(line + "word count: " + str(len(tokens)))
f.close()
f_out.close()
```

+ sign is used to append.

str(len(token))

r → Opens the file for reading only. Throws error if file doesn't exist.

w → Opens the file for writing (if file doesn't exist it will create new one. if it exist then it'll overwrite).

r+ → Opens file for both reading & writing.

w+ → Opens file for reading & writing. if file " " "

a → Opens a file in append mode. Whatever you write to file will get appended and original content will not be overwritten.

Date:

You don't need to do fclose() if you use with. Using with will automatically close the file for you.

```
with open("C:\\date\\funny.txt", "r") as f:
    print(f.read())

print(f.closed).
```