① Download Python 3.6 & Install.

download text editor ⇒ IDLE

Integrated development

⇓

Pycharm (Editor)

www.jetbrains.com/pycharm

② Basics

Hello world & printing on console

↓

Triangle shape

③ Variables & Data types

Variable: putting data inside something

* Application of variables by a story

eg: Char_name = "Har"

Age = "25"

print ("there is a mano named " + Char_name + ", ")

print ("His age is " + Age)

↳ working with strings :

① new line ⇒ print ("Hari \n phi")

② print (") symbol ⇒ print ("Hari \" Acaday ")

③ store string & then print

phrase = "Hari-phi"
print(phrase)

④ Concatination

then print (phrase + "is cool")

⑤ make everything lower or upper case

⑥ phrase = "Hari-phi".
print = (phrase.lower())
↳ upper()

o/p : hari-phi.

⑥ check upper or lower

⑥ phrase = "Hari-phi".
print (phrase.isupper())

o/p : false

⑦ length of char

print (len(phrase)).

⑧ printing individual chars.

phrase = "Hari - Phi"

print (phrase[0])

o/p : H

⑨ Getting the index (position).

phrase = "Hari - Phi".

print (phrase.index("H"))

o/p : '0'.

print (phrase.index("i"))

o/p : 4, 8.

print (phrase.index("ph"))

o/p : '5'. (starting value).

⑩ Replace :

phrase : "Hari - Phi"

print (phrase.replace("Phi", "Chari"))

inserting one name in another.

↳ Working with numbers.

① Math:

```
print (3 * 4)
print (3 / 4)
print (3 * 4 + 5)
```

o/p : 3×4 + 5 = 12 + 5 = 17

```
print (10 % 3)
```

o/p = 1 (remainder)

② number with variable

```
my_num = 5
print (str(my_num))
```

↳ then number '5' is converted to a string.

Note: If you want to put number beside a string, you always do like above.

③ Absolute value

```
my_num = -5
print (abs(my_num))
```

o/p : '5'

④ power

```
print (pow (3, 2))
```

o/p : $3^2 = 9$.

⑤ maximum number (or) min

```
print (max(4, 6)) ⟹ 6
print (min(4, 6)) ⟹ 4
```

⑧ rounding number

print(round(3.7)) ⇒ 4

Note: to do more we need math library

↳ from math import *

⑦ Square root

print (sqrt (36))

o/p : 6

⑨ ceil

print (ceil (3.7)) ⇒ 3

④ Input :

eg: name = input ("enter your name")
print = ("Hello" + name + "!")

o/p: Hello Hari !

Project 1: Calculator:

```
num1 = input ("enter a number:")
num2 = input ("enter another number:")
result = num1 + num2
print (result)
```

error: num1 & num2 are taken as string

So 2+3

= 23

to solve this

result = int(num1) + int(num2)

this convert num1, num2 string to integer.

problem: Here we can't add 2.3+3.5 because it's 'int'

So, we can use 'float'

result = float(num1) + float(num2)

op: 2.3+3.5 = 5.8

H/W: Finding Roots of equation

Project 2 : Mad libs.

Prompt

```
color = input ("enter a color : ")
noun = input ("enter a noun.")
name = input ("enter a name")
```

Give them work

```
print (" roses are " + color)
print ( noun + " are blue")
print ( " i love" + name ).
```

(5) Lists :-

List can be any type int, float, char etc.

```
Data = [ "Hari", 23, "yes" ]
```

Printing (1)

```
print (Data(-1))
```

o/p = yes

```
Print (Data (1))
```

o/p = ['23']

```
Print (Data [1:])
```

```
Print ['23', 'yes']
```

```
print (Data[1:][0], Data[1:][1])
```

Scanned with CamScanner

② modify:

```
Data ("Hari", 25, "Yes")
Data[0] = "chari"
print(Data[0])
```

o/p: chari.

③ ↳ Adding lists (beside) i.e extending

```
numbers = [2, 3, 4]
name = ['H', 'I', 'J']
name.extend(numbers)
print(name)
```

o/p = ['H', 'I', 'J', 2, 3, 4]

(extend in 'name' → then printed 'name')

④ ↳ appending

```
name.append('K')
print(name)
```

o/p = ['H', 'I', 'J', 'K'].

⑤ ↳ insert

```
name.insert(1, "O")
print(name)
```

o/p = ['H', 'O', 'I', 'J', 'K'].

(6) Remove :

name. remove ("I")

print (name)

o/p : ['H', 'J', 'K']

(7) clear :

name. clear ()

print (name)

o/p : []

(8) pop :

removes last element in the list

name. pop ().

(9) Search :-

name = ["H", "I", "J", "K"]

print (name. index ("H"))

o/p = '0' (index value)

(10) Counting How many values exist :

```
name=['H', 'A', 'E', 'I']
print(name.count("H"))
```

o/p = 2

(11) Sorting : Alphabetical

```
name.sort()
print(name)
```

o/p : ['A', 'E', 'H', 'I']

(12) Reverse the list

```
name.reverse()
```

(13) Copy :

```
name2 = name.copy().
```

(6) Tuple :—

Tuple is immutable i.e we can't change it.

↳ tuple is same as list but immutable

eg: coordinates = (4, 5)

(or)

coordinates = [(4, 6), (6, 7), (20, 34)].

print (coordinates[1])

o/p = (6, 7).

(7) Functions :—

Function is a collection of code.

↳ 'def' is a keyword used to create function

eg:1 :

```
def sayhi():          ← indentation is must after this
    print("Hello user!")
print("You")
sayhi()
print("me")
```

o/p : You
Hello user!
me

Condition : All should be lowercase, use underscore

eg 2: def say_hi(name):
    print("Hello" + name)

say_hi("mike")
say_hi("steve")

o/p : Hello mike
Hello steve.

eg 3: def say_hi(name, age):
    Print("Hello" + name + "you're" + age)

say_hi("mike", "30")

o/p : Hello mike you're 30

∴ (b) Return function :-

eg:

```
def cube(num):
    return num * num * num      # sd here
print(cube(3))                  # then brought here
```

("// this value goes to the function where ever we use")

eg 2:

```
def cube (a,b)
    return a*a*a
value = cube (4)
print (value)
```

O/p: 64.

Note: You can't reach after return statement

```
return a*a*a
print ("Hi")
```

↳ then "Hi" is not printed

8 If :—

eg: 1:

```
is_male = False
if is_male:
    print ("you are a male")
```

O/p: Blank (because it is false)

give real time examples like going to movie etc

eg 2:

```
is_male = False
if is_male:
    print("you are male.")
else:
    print("you are not a male")
```

eg 3:

```
is_male = True
is_tall = True
if is_male or is_tall
    print("your are male or tall or both")
if else: is_male and is_tall
else:
    print("you neither male nor tall")
```

O/P:

eg 4:

```
is_male = True
is_tall = False
if is_male and is_tall:
    print(" ")
elif is male and not (is_tall):
    print("you're male but not tall")
else
    print("you are either not male or not tall")
```

O/P:

8(b)

Comparisions in If statements :-

eg,

```
def max_num (a, b, c)
    if a>=b and a>=c
        return a
    elsif b>=a and b>=c
        return b
    else
        return c
print (max_num (2, 3, 4))
```

o/p: 4

!= not equal

>= greate or eq

== Compare

= equal

'and' → and operator

'or' → or operator

'not' → not operator

Project 3: Building Advanced Calculator.

```
num1 = float(input("enter first number:"))
num2 = float(input("enter 2nd number"))
op = input("enter operator:")

if op == "+":
    print(num1 + num2)
elif op == "-":
    print(num1 - num2)
elif op == "/":
    print(num1 / num2)
elif op == "*":
    print(num1 * num2)
else: print("error operator:")
```

making progress card

(9) Dictionaries :-

↳ to store key value pair we use dictionary.

word = key
definition = value } just like dictionary.

eg: monthConversion = {

"Jan" : "January",
"Feb" : "February",
"mar" : "march",
}

print (monthConversion["Jan"])

o/p = January.

(or)

print (monthConversions.get("Dec"))

(or)

print (monthConversions.get("key", "not a valid"))

o/p : not a valid.

Note: Instead of Jan, Feb we can use 0, 1, 2 also.

(10) while :—

```
i = 1
while i <= 10:
    print(i)
    i = i+1        ⟶ or  i += 1
print("done with loop")
```

o/p:

```
1
2
.
.
.
10
done with loop.
```

Project 4: Basic Guessing game : [password game]

```
secret_word = "giraffe"
guess = ""
while secret_word != guess:
    guess = input("enter guess: ")
print("you win!").
```

Set limit for this :

```
guess = ""
guess_count = 0
& put
guess = input("enter guess:
guess_count += 1
```

this increase

Set limit for above game:

```
secret_word = "giraffe"
guess = ""
guess_count = 0
guess_limit = 3
out_of_guesses = False

while guess != secret_word and not (out_of_
                                    guesses):
    if guess_count < guess_limit:
        guess = input("enter guess: ")

    else:
        out_of_guesses = True

if out_of_guesses:
    print("out of guesses. You lose!")
else:
    print("you win").
```

(11) For loop :-

eg: `for letter in "Hari phi":` (anything → letter, defined)
`    print(letter)` } this prints each letter in an iteration

o/p : H
a
r
i
p
h
i

eg2: `friends = ["Hari", "Chary", "phi"]`
`for friend in friends:`
`    print(friend)`

o/p: Hari
Chary
phi

eg3:
`for indx in range[3, 10]`
`    print(indx)`

o/p: 3
4
5
.
.
9

`for a in [0, 3]` → step1: 0 is kept in 'a'
Step2: then we use 'a' for next
`print(a)` etc.

Project 5: Create exponent value using for loop

```
def raise_to_power(base_num, pow_num):
    result = 1
    for indx in range(pow_num):
        result = result * base_num
    return result

print(raise_to_power(4, 2))
```

o/p : 16

(12) List of lists (2D) & nested loops

```
numb_grid = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
    [0]
]

print(numb_grid[2][1])
```

o/p = 8

eg: numb_grid = [
[1,2,3],
[4,5,6],
[3,5,9],
[0]
]

```
for row in numb_grid:
    for col in row:
        print(col)
```

o/p: 1
2
3
:
9

Project 6:- Translator :- (nesting for with if)

All vowels should become 'g'.

```
def translate(phrase):
    translation = ""
    for letter in phrase:
        if letter in "AEIOUaeiou"
            translation = translation + "g"
        else
            translation = translation + letter

print(translate(input("enter a phrase:")))
```

Give more to do. Something more

o/p: enter a phrase: Hello
o/p: Hgllg.

Excess 3 letters.

(13) Errors in python.

eg: → Value error

```
number = int(input("Enter a number"))
print(number)
```

o/p: enter a number : abcd

↳ "Value Error"

(13b) try & except:

eg: → try:

```
    number = int(input("Enter a number:"))
    print(number)
except:
    print("Invalid input")
```

o/p: enter a invalid input

Note: this is used when we need to ignore few inputs which disturb our program.

eg:

```
try:
    number = int(input("enter a number: "))
    print(number)
except ZeroDivisionError:
    print("Divided by zero")
except ValueError:
    print("invalid i/p")
```

→ Python takes the error name and compare it & give o/p.

(14) Reading Files (external):-

Sometimes we need to get data from text, CVC, word etc files. for that we'll have Commands

read (r): only reading
write (w): only writing
append (a): adding info after the file text etc
read & write (r+): read & write.

filename.readable ⇒ this gives boolean value ~~Yes or~~ true of false

filename.read() ⇒ reads file completely
we can print by putting
print(filenam.read())

filename.readline() ⇒ reads first line
if we repeat it it'll print
~~t~~ fully

print(filename.readlines()) ⇒ reads line & put in array & prints.

eg: employee_file = open("employee.txt", "r") → mode
filename

print(filename.read())

o/p = total file
Content

eg:2 appending

append

```
employee_file = open("employee.txt", "a")
employee_file.write("Toby - Human resources")
employee_file.close
```

eg:3 : writing:

when we put "w"
then whatever we write that will be stored in file everything vanishes.

(15) modules & pip :-

step1: create a python file '.py' eg: def heyhey etc

step2: use this file in current program
to do this we need to import previously created file

eg: print (heyhey)

step3: eg: import heyhey

eg: print (heyhey.roll_dice(10)) etc

3rd party modules needs to be downloaded

& to install we use Command "pip"

eg: pip install python-docx.

↳ open cmd

↳ Install pip → if we don't have then only by do this

↳ If we have [usually python 3 & above we will have]

↳ check version by command

↳ pip --version.

↳ pip install filename → including extension.

eg:

pip install python-docx

✓ uninstall pip using install python-docx

(16) Classes & Objects :-

Sometimes cannot be represented with available data types, so we can create a data type.

eg: phone information can be stored

Class is used to define datatypes.

eg: Create a student class :-

```
Class student:
    def __init__(self):
```

→ mapping attributes of this datatype

~~class.py~~ student.py ⟶ file name

↓

inside this write

↓

```
class student:
    def __init__(self, name, major, gpa, probation):
        self.name = name
        self.major = major
        self.gpa = gpa
        self.probation = probation
```

→ auto cue need this to proceed

→ [name of the student is equal to the name we passing in]

actual object name is equal to the name given by user/ programmer

↓

Save this & this can be used as follows

Create a new python program eg: (Pyth.py)

↓

file name → from student import student ← class name

```
student1 = Student("Jim", "Business", 3.1, False)
print(student1.gpa)
```

o/p : 3.1 ; we can use stud2 stud3 etc.

we can model any objects with any attributes.

Proj 7: multiple choice Quiz:

major project

Make a class for Questions: name it Question.py

```
class question:
    def __init__(self, prompt, answer):
        self.prompt = prompt
        self.answer = answer
```

make a py file

↳ check next page

(actual code : array of questions ;

```
from Question import Question
question_prompts = [
    "what colour are apple \n (a) Red (b) purple \n\n",
    "what colour are bana \n. (A) Red (B) Yellow (c) white",
]

questions = [
    Question(question_prompts[0], "a"),
    Question(question_prompts[1], "b"),
    Question(
]
```

a 'a' is Correct

```
def run_test(questions):
    score = 0
    for question in questions:
        answer = input(question.promt)
        if answer == question.answer:
            score += 1

    print("you got " + str(score) + "/" + str(len(questions)) + "Correct")

runtest(questions)
```

⟹ Run.

(12) Object functions :

Class function/object function is a type of function that we can use inside a class that can modify objects of that class or give a specific info about these objects.

Create a class file : student.py

eg:

```
class student :
    def __init__ (self, name, major, gpa):
        self.name = name
        self.major = major
        self.gpa = gpa

    def on_honor_roll (self):
        if self.gpa >= 3.5 :
            return True
        else:
            return False
```

Create actual file : accounting.py

```
from student import student
student1 = student ("oscar", "Accounting", 3.1)
student2 = student ("phyllis", "business", 3.5)
print (student1.on_honor_roll())
```

o/p : False

(18) Inheritance:

This is basically where we can define a bunch of attributes and functions and things inside of a class, and then we can create another class and we can inherit all of those attributes

eg: Create a class: chef.py

```
class Chef:
    def make_chicken(self):
        print("the chef make chicken")
    def make_special_dish(self):
        print("the chef makes bbq ribs")
```

this class can be moved to any class

eg. we need a chinesechef.py which should have above

put here

```
class chinesechef:
    def make_special_dish(self)
        print("the chef makes orange chicken")
    def make_rice(self)
        print("does fried rice")
```

except special dish rice

↳ both of them can be used for actual program

```
from chef import chef
from chinese_chef import chinese_chef
my_chef = chef()          ⟶ calling first class chef.py
my_chef.make_special_dish()  ⟶ this print special of chef.py )

my_chinese_chef = chinese_chef()
my_chinese_chef.make_special_dish()
```

op: the chef makes bbq ribs
the chef makes orange chicken.

So instead of copy pasting we can directly use inheritance:

↳ to do that : in class created for chinese chef we can write like this

```
from chef import chef
class chinese_chef(chef):
    def make_rice(self):
        print("makes rice")
```

this is like inherit one class to other

Note; python Interpreter :

Open Command prompt : cmd

type : python 3. if not working add python 3 to windows path variable check geeks

just to check not to write

django