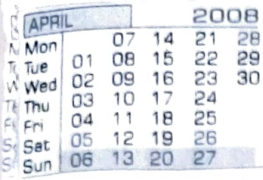
\n → new line
\t → tab
\\ → \



March
Tuesday
11

Python

i++ → not exists in python

=> extension → .py
=> No semicolon required in the end of line
=> print("Hello")
=> we can't give space in between
=> single line comments start with #
=> multi-line comments enclosed b/w ... ≡ ...
=> variables → no need to specify data-types.
=> name = "Yash" # strings
age = 30 # integer
cgpa = 9.6 # float
is_tall = True # boolean
=> print("My name is " + name)
=> print("My name is", name)

=> print(int(3.14)) # converting/casting into another data type
=> print(float(3))
=> print(str(True)) → string
=> print(int("50") + int("70"))

=> name = "yash"
print(f"Hello, {name}")

=> python hello.py
↳ to run in CLI

No Notes

Appointment

IDE

APR
MAY
JUN

"{:.2f}".format value

12 March Wednesday

MARCH					2008
Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

strings

greeting = "Hello"
# indices : 01234

=> print (len (greeting)) # printing length of a variable

=> print (greeting[0]) # printing letters at specific indices

=> print (greeting[-1]) # prints 'o'; counts from back side

=> print (greeting.find("llo")) # points 2, index position
↓
it gives -1 if string is not present

(greeting[2:]) → returns all values from given index (2) to end

(greeting[2:3]) → starting index, ending index
→ returns all values from 2 to 3

[:] ≡ [0:end]

[::2]
→ to select every second variable

[0:5:2] → step size
2nd
→ every index upto 4

[::-1]
↓
reversing a string

Notes Appointment

operators
operands

=> print(2*3) # 2×3
=> print(2**3) # 2^3
=> (10 %.3) # render.

=> print(10/3.0) # okay to use int with float
// ⟶ answer is in integers (output)
=> num +=10 , -=, *=, /=

=> we have to import maths library to use math f^n's.

=> import math
print(pow(2,3)) # 2^3
print(math.sqrt(144))
print(round(2.7))
=> BODMAS is followed

# USER INPUT
A = list(map(int, input().split()))
name = input("Enter your name:")
print("Hello" + name + "!")

=> it takes input as string, you have to convert it into other data types.

=> Reverse indexing in python

M i c h a e l J a c k s o n
-15 -14 -13 -12 -11 -10 -9 -8 -7 -6 -5 -4 -3

Notes
Appointment

APR
MAY
JUN

14 March
Friday # Arrays

=> we can store different datatypes in single array.

nums = [4, 8, "yash", 3.2]

N Dimensional lists

number = [[1, 2], [3, 4]]

list functions / creating arrays

=> friends = [ ]
friends.append("yash")
friends.append("abhay")
friends.insert(1, "Kevin")
↳ inserts Kevin to index position 1
friends.remove("Kevin")
print(friends)
↓
friends.index("oscar") # returns index
friends.count("oscar") # returns no. of times an element occures.

Notes

print[0:3]
sorted(<tuple name>)

Appointment

friends.sort() # sorts in alphabetical order.

friends.clear() # clears list.

=> friends[3] = "ashish" # you can't modify tuples like that

functions

=> def add_nums(num1, num2 = 99):
    return num1 + num2

sum = add_nums(4)
sum2 = add_nums(4,2)
print(sum)
print(sum2)

output => 103
6

Sunday 16

if statements

=> if 1 > 3:
    print("yash")

=> if P == "cat":
    print("cat")

Notes

Appointment

17 March
Monday

MARCH					2008
Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

else-if statements -

```
if is_student and is_smart:
    print("a")

elif is_studest and not(is_smart):
    print("b")

else:
    print("c")
```

dictionaries

```
=> test_grades = {
        "yash" : "A+"
        "abhay" : "C"
        # 3 : 4.5
    }
print(test_grades["Andy"])
print(test_grades.get("Andy", "N/A"))
```

keys
values

values can be repeated but keys should be unique

searches for value of Andy

default output

Notes

Appointment

while loop

```
index = 1
while index <= 5:
    print(index)
    index += 1
```

for loops

-> for index in range(5)
    print(index)

creates a variable

=> for letter in "Giraffe"
    print(letter)

goes from indices to indices

=> for i in [0,1,2]
    print("Hello")

exception catching

a = 10/0 gives ZeroDivisionError

a = 10/"p" gives ValueError

=> try:
    a = 10/0
except:
    print("Divided by zero")

Notes

Appointment

19 March
Wednesday

MARCH					2009
Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

⇒ try:
    a = 10/0
except ZeroDivisionError as e:
    print(e)
except:
    print("Error")

⇒ class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
    def read_book(self):
        print("Reading", self.title, "by",
              self.author)

book1.Book("Harry Potter", "Roling")
book1.title = "yash"
print(book1.title)
book1.read_book()

Notes

Appointment

#Getters & setters

```
class Book:
    def __init__(self, title, author):
        self.title = title;
        self.author = author

    @property
    def title(self):
        print("getting")
        return self._title

    @title.setter
    def title(self, value):
        print("setting")
        self._title = value

    @title.deleter
    def title(self)
        del self._title

    def read_book(self):
        print("reading", self.title)
```


Notes

Appointment

21 March
Friday

INHERITENCE

class chef:

class itachef(chef):

ychef = chef()
ychef.make_chicken()

yichef = itachef()
yichef.make_chicken()

super().__init__(name, age)
↓
using init fn of super class

Notes

Appointment

⇒ Big O notation is used to know how running time or space requirements grow for our program as input size grows.

time = a * n + b
↳ × } neglect

time = O(n)

→ after making eq^n by graph, keep fasted growing term and drop constants.

⇒ Measuring running time growth
↓
time complexity

⇒ Measuring space growth
↓
space complexity

Sunday 23

⇒ static array → not allows inserting more elements than specified

⇒ Dynamic array → you can go on adding elements -
↳ They follow GP for memory allocation
↳ they double the previous array & add to it.
10 → 10+20 → 30+30.

Notes
Appointment

24 March
Monday

MARCH					2008
Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

Cs50 - Pset. Sequence types in python

range(5) → gives you an array from zero to 5

list → array(dynamic)

tuple → comma separated values

dict → we store keys & values

set → storing group of data with no duplicates.

⇒ to choose odd place elents.

```
for i in range(3,2)
    print(i)
```

⇒ swapping in python.

→ x, y = y, x

Dynamic typing - A technique in some languages where depending on how a value is used, data type of a variable is dynamically & Automatically assigned.

Notes

Static typing - Data type is declared before it is used.

Appointment

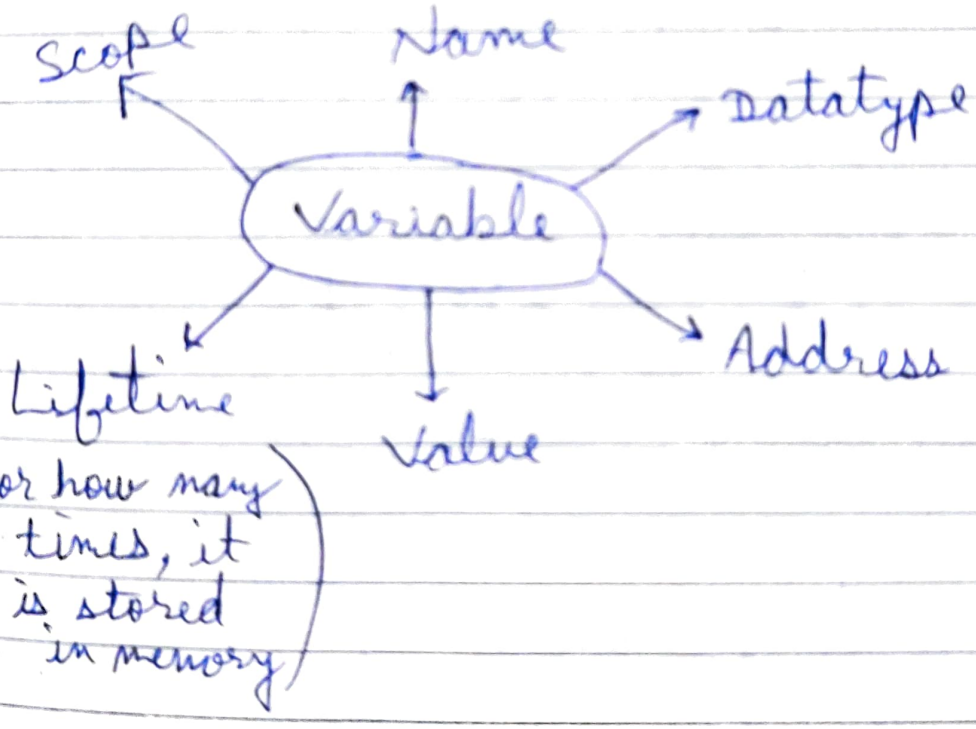
Types of control structures

- Selection
- Iteration

Python — if, else, for, while, def, print, raise, try, except

Javascript — if, else, for, while, var, function, object, break, continue

Go — if, else, for, var, func, import, package, main



=> To read two spaced integers.

```
(a, b) = map(int, input().split(' '))
```

26 March Wednesday

MARCH					2008
Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

=> num = "one"
num = num * 3
num : "oneoneone"

=> string methods
A.upper()
A.replace("J","A") → Replace J to A

CS50

=> from cs50 import get_string
ans = get_string("Your name?")
print("hello, " + ans)
or
print(f"Hello, {ans}")

=> i++ → doesn't exist in python.

=> if x < y:
print("x is smaller")
else:
print("y is smaller")

=> In python → else if → elif

=> while True:
print("while loop")

Notes

In python → True ✓ true ✗
False ✓ false ✗

Appointment

=> for i in [0,1,2]:
    print ("hello, world")

=> for i in range(3):
    print ("hello, world")

=> if s=="Y" or s=="y":
    print("Yes")

=> if s in ["y","yes"]:
    print ("Agree")

=> def meow():
    print("meow")

=> def main():
    for is in range(3):
        meow()

def meow():
    print("meow")

main()

=> scope is not present in python.

=> default print fg^n

print("?", end="\n")

can be changed to remain in same line

print("?", end="")

Notes

Appointment

28 March
Friday

MARCH 2008

Mon	31	03	10	17	24
Tue		04	11	18	25
Wed		05	12	19	26
Thu		06	13	20	27
Fri		07	14	21	28
Sat	01	08	15	22	29
Sun	02	09	16	23	30

=> Command line arguments

```
from sys import argv
if len(argv)==2:
    print(f"Hello, {argv[1]}")
```

=> opening a csv file.

```
import csv
file = open("phonebook.csv","a")
name = input("Name: ")
number = input("Number: ")

writer = csv.writer(file)
writer.writerow([name, number])
file.close()
```

=> with ~~func~~ keyword is used to close file automatically after use.

→

```
with open("phonebook.csv","a") as file:
    writer = csv.writer(file)
    writer.writerow((name, number))
```

Notes

Appointment

EdX

Constants - whose values remain same
Reserved words - words has specific meaning in python & they can only be used in that cases
Ex - False, and, for, from, break, etc -
variable - named place in memory where a data is stored -

NRTI

=> age = 5

print("You are", age, "Years old")

=> Tuples

t = 1, yash, 2, amey
t = () # empty tuple
t = 1, # singleton tuple

t = tuple("789")
print(t)
↳ ("7", "8", "9")

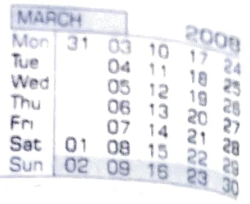
len(t) → gives no. of elements in a tuple

unpacking -
>>> student = 'yash', 32, (KV, patna)
>>> name, roll, aca = student
>> print(name)
'yash'
>>> print(roll)
32

Notes

Appointment

31 March Monday



=> Unpacking arbitary numbers

>>> x, y, *z = (10, 'yash', 'a', 5, 10, 15)

→ only one * assignent is allowed

=> using 'or' & 'not' notions

0 → False ← empty string ("")

any otha number → True ← any other string

>>> print (1 and 2)

↳ output = 2

↳ it gives last evaluated element

=> If you use 'not', answer will be True or false

>>> print(x, end="", sep = " ")

eps youtube

List = [1, 2, 3, 4]

List. insert (1, 9)

List = [1, 9, 2, 3, 4]

=> Insert f^n first swaps all rest elements than inserts new element

Notes

Appointment

=> Benefits of linked list over array -

(1) We don't need to preallocate space

(2) Insertion is easier

Notes

Appointment

02 April Wednesday

APRIL 2008

Mon		07	14	21
Tue	01	08	15	22
Wed	02	09	16	23
Thu	03	10	17	24
Fri	04	11	18	25
Sat	05	12	19	26
Sun	06	13	20	27

Greedy Algorithms

Applications —

1. Activity Selection problems
2. Huffman coding
3. Job sequencing problem
4. Fractional Knapsack Problem
5. Prim's Minimum Spanning Tree

(1) i) sort activities according to finish time
ii) select first activity from sorted array & print it.
iii) If next activity has start time greater or equal to end time of prev. printed activity, then print it.

else.
leave

Notes

Appointment