Phython bootCamp by Ehmad Zubair

1) Variable ;

key → name = " Talha " ✓ value.

Variable cannot be initiated with some symbols like " " , ! and others.

" if , while , elif " these can be used to declare variable name.

2) Statement :

like

1 + 1

We can write multiple statements on one line with use of semi - colon.

name = "Talha" ; print ('name')

3) Comments ;

' # ' Everything written with hashmark is comment.

In python indentations matters. Anything indented belongs to a block.

23 Thursday
27 Rabi-II

4) Datatypes:

To check datatype we use "type"

```
name = "Talha"
print (Type (name)) == Str
```

Output
<class 'str'>
True

↑ To verify the datatype.

---

24 Friday
28 Rabi-II

"isinstance" is also used

```
print (isinstance (name, str))
```

Output
True.

=

We can convert datatype from one form to another. This is called casting.

```
number = "20"   ← This is string.
age = int (number)
print (isinstance (age, int))
```

Output
True

---

25 Saturday
26 Sunday
29 Rabi-II
01 Jamadi-I

Complex	for	complex numbers
bool	for	booleans / True, False
list	for	lists
tuple	for	tuples
range	for	ranges
dict	for	dictionaries
set	for	sets

27 Monday
02 Jamadi-I

5) Operators;

Airthmetic operators

+	
-	
*	
/	divide
%	Reminder / Modulus
**	Square
//	floor division

'+' operator is also used for concatenating.

28 Tuesday
03 Jamadi-I

e.g:

```
age = 5
age += 5 #   age = age + 5.
print (age)
```

Output: 10

Comparison Operators

==	equal
!=	not equal
>	greater than
<	less than

29 Wednesday
04 Jamadi-I

Boolean Operator

not , and , or

Bitwise operators

&	performs binary AND
|	performs binary OR
^	performs binary XOR
~	performs binary NOT
<<	shift left operation.
>>	shift right operation.

30 Thursday
05 Jamadi-I

is, in Operators

"is" is identity operator

"in" is membership operator

Ternary operator:

e.g: def is_adult(age):
return True if age > 18 else false.

---

01 Friday
06 Jamadi-I

6) Strings

" "
' '

Another method is triple quotes for multiple lines.

```
print ("""  Talha
is
Student
""")
```

Output:
Talha
is
Student

---

02 Saturday
03 Sunday
07 Jamadi-I
08 Jamadi-I

7) String Methods

```
print ('Talha'.upper())
"  ("  ".lower())
   (    .title())
```

Output
TALHA
talha
Talha * Capitilize first letter

"\n" is for new line

04 Monday
09 Jamadi-I

8) Boolean.

```
done = True
if done:
    print('Yes')
else
    print('No')
```

Output

Yes

9) Complex numbers:

05 Tuesday
10 Jamadi-I

=> Complex = 2 + 3i (real, imaginary)

=> num = complex(2, 3) (real, imaginary)

=

10) Built-in functions;

```
print(abs(-5.5))   → 5.5
print(round(5.5))  → 6
```

(abs: absolute)

Output

06 Wednesday
11 Jamadi-I

11) ENums;

Are readable name that are bound to constant value.

e.g#

```
from enum import Enum
class State(Enum):
    Inactive = 0
    Active = 1
print(State.Active.value)
```

Output

1

07 Thursday
12 Jamadi-I

12) User Inputs.

"Input" is used to taking input. The program will stop until user input something.

```
age = input(5)
print("your age is" + age)
```

13) Control Statement;

08 Friday
13 Jamadi-I

```
condition = True

if condition == True
    print("Yes")
```

Output

Yes

14) Lists;

e.g#

```
dog = ["Alpha", "Beta", 2, "2.4"]
print("Beta" in dog)
```

Output

True

09 Saturday
10 Sunday
14 Jamadi-I
15 Jamadi-I

e.g#

```
dog[2] = "Gamma"
print(dog[2])
```

Output

Gamma.

Index method is also used in lists. It is similar to arrays.

day

15) Append / Extend

↳ is used to add item

like

```
dog.append("Charlie")
```

Extend is used to extend the list like

---

day

```
dog.extend(["Johnny", 5]).
```

↳ This will become part of dog list.

"+=" operator is alternative to extend.

---

nesday Don't forget square brackets in extend method.

```
dog.remove("Charlie")
```

This will be removed from the list.

16) Insert method:

14 Thursday
19 Jamadi-I

To add item at a specific index.

index number

```
dog.insert(2, "Tutty Futty")
```

Item to be added.

17) Sorting the list

15 Friday
20 Jamadi-I

```
dog.sort()
```

→ It will sort in alphabetical order.

→ Upper case first and lower case later.

→ The list must contain same datatypes. only string, only integers. etc.

18) Copy the list.

16 Saturday
17 Sunday
21 Jamadi-I
22 Jamadi-I

This is called slice

```
dogcopy = dog[:]
print(dogcopy)
```

Output:

The original list without sorting.

Alternative;

"Sorted" global function can be used

19) Tuples;

Tuples cannot be modified unlike lists. We use paranthesis instead of square brackets.

```
names = ( "Roger", "Syd" )
print ( name[0] ).
```

output

Roger

We can use new variable like

```
newnames = name + ( "Tina", "Goincy" )
print ( newnames ).
```

=> You cannot modify the original Tuple.

20) ~~Dictionaries~~;

Scope of Variable;

Global variable that is declared outside ~~program~~ function can be used anywhere in the program.

But, variable declared inside function can't be used outside.

21 Thursday
26 Jamadi-I

21) Nested function;

Function inside a function.

e.g #

```
def talk(phorse):
    def say(word):
        print(word)

    words = phrase.split(" ")
    for word in words:
        say(word)

talk("I am buying milk")
```

Output
I
am
buying
milk.

---

22 Friday
27 Jamadi-I

22) Objects;

```
age = 8

print(age.real)
print(age.imag)
```

Output
8
0

---

23 Saturday
24 Sunday
28 Jamadi-I
29 Jamadi-I

23) Loops;

While loop;

```
Count = 0
while count < 10
    print " It is true "
    count t=1
print "After the loop"
```

25 Monday
01 Jamadi-II

For loop:

```
for item in range(15):
    print(item)
```

=

To find index;

```
item = [1, 2, 3, 4]
for index, item enumerate(items):
    print(index, item)
```

Output

0	1
1	2
2	3
3	4

26 Tuesday
02 Jamadi-II

Use to continue / breaks statement before print will run / stop program.

=

24) Classes;

From classes we can intiate object. Object is instance of class.

```
class dog:
    def bark(self):
        print("woof")

roger = Dog()
```

object →

27 Wednesday
03 Jamadi-II

28 Thursday
04 Jamadi-II

25) Constructor:

```
Class dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        print("woof")

roger = dog('roger', 8)
print(roger.name)
print(roger.age)
```

Output:

Roger
8

29 Friday
05 Jamadi-II

26) Inheritance;

```
Class Animal:
    def walk(self):
        print("walking")

Class dog(Animal):
```

It will inherit above properties.

```
roger.walk()
```

30 Saturday
31 Sunday
06 Jamadi-II
07 Jamadi-II

27) Modules:

Every python file is module. It is used to define functions in one files and import them in another files.

01 Monday
08 Jamadi-II

Scenario:

If use store modules in a subfolder. For accessing it outside we use

__init__.py file

It tells the phython that this folder has modules.

=

02 Tuesday
09 Jamadi-II

28) Lambda function;

Lamba num = num * 2

multiply = lambda a, b : a * b
print ("multiply" 2, 4))

03 Wednesday
10 Jamadi-II

29) Map, filter, Reduce.

We can replace map, reduce, filter using the lambda functions.

e.g# expenses = [
('Dinner', 80),
('Car repair', 120)
]

04 Thursday
11 Jamadi-II

```
sum = 0
for expense in expenses
        sum += expense[1]

print (sum).
```

=

Alternative:

```
from functools import reduce

expenses = [
        ('Dinner', 80)
        ('Car repair', 120)
]

sum = reduce(lambda a, b : a[1] + b[1], expenses)
print (sum).
```

=

Conclusion:

Lambda function reduces the length of code.

=

05 Friday
12 Jamadi-II

06 Saturday
07 Sunday
13 Jamadi-II
14 Jamadi-II

30) Recursion;

for example:

$3! = 3 \times 2 \times 1 = 6$

We can do this using recursion.

```
def factorial (n):
    if n == 1 : return 1        ✓ Base case
    return n * factorial (n-1)  ✓ recursive case

print (factorial (3)).
```

08 Monday
15 Jamadi-II

At 1000 call python give recursion error.

31) Decorators;

```
def hello();
    print("hello")
```

We use decorators to not modify the real function. We declare

09 Tuesday
16 Jamadi-II

decorators by "@" this sign.

e.g#

```
def logtime():
    def wrapper():
        print("before")
        val = func()
        print("after")
        return val
```

10 Wednesday
17 Jamadi-II

```
    return wrapper

@logtime
def hello():
    print("hello")

hello()
```

11 Thursday
18 Jamadi-II

32) Docstrings;

Another way of comments is docstrings.

```
"""
    This is
    a doc
    string
"""
```

To understand your code aftermonths.

12 Friday
19 Jamadi-II

Docstring also help in documentation.

```
eg#   """ Defining car function"
      def car():
          input model = 2018
          input name = Picanto

      print (car.model , car.name).

      print (help(car))
```

13 Saturday
14 Sunday
20 Jamadi-II
21 Jamadi-II

33) Annotation;

phython does not require to add datatypes, if we want to declare datatype we use annotations.

```
def increment (n: int) -> int
    return n+1
```

*!- MyPi toot is used to checkbugs before executing.

15 Monday
22 Jamadi-II

34) Exceptions ;

So we use try and except; in error handling.

```
try :
    results = 2/0
except zerodivisionerror :
    print " zero error "
finally:
    result = 1
print (result)
```

Output

zero error

1

16 Tuesday
23 Jamadi-II

35 With statement :

```
filename = '/User/PC/ docx.txt
with open (filename, 'r') as file
    content = file.read()
    print (content)
```

17 Wednesday
24 Jamadi-II

"with" is capable of lot do more.

18 Thursday
25 Jamadi-II

270,000 packages at Pipe.org.

36 Third-party packages.

"Pip" ↙ http library:

pip install request

Upgrade version;

pip install -U request

---

19 Friday
26 Jamadi-II

Uninstall;

pip unistall request

37 List Compression;

```
number = [1, 2, 3, 4, 5, 6]
number_power_2 = [n**2 for n in numbers]
print(number_power_2)
```

---

20 Saturday
21 Sunday
27 Jamadi-II
28 Jamadi-II

List compression are used to replace loops.

38 Polymorphism;

Different class names but ~~different~~ same parameters.