# makes it a comment -------- python ignores any **INPUT** words on that line.

**F-Strings**

In order to combine strings with other data types, you can format them using an f-string.

An f-string is created like a regular string but with the letter f preceeding the first quote. Then, to format other data types into the string, they are placed inside curly brackets, { }.

For example:

>>> age = 25

>>> f"John is {age} years old"

'John is 25 years old'

>>> f"John will be {age+10} years old in 10 years!

'John will be 35 years old in 10 years!'

This applies to other data types, not just to integers, and also allows us to embed expressions in our strings.

var1 = "Good"

var2 = "Day"

var3 = "Ainsley"

print (var1,var2,var3)

word1 = "Good"

word2 = "Day"

word3 = "Ainsley"

word4 = 'Again'

sentence = word1,word2 ,word3,word4

print(sentence)

number1 = input("Enter whole number: ")

number2 = input("Enter decimal number: ")

integer\_number = int(number1)

float\_number = float(number2)

round\_number = int(round(float\_number))

print(number1)

print(number2)

print(round\_number)

# Pet

name = "Pep Guardogiola"

age = 3

bark = True

tweet = False

print("My pet is called", name, ", He is", age, "years old.")

print("Statement:", name, "barks.", bark)

print("Statement:", name, "tweets.", tweet)

The back-slash “\” allows us to escape characters, telling the interpreter to treat them differently.

>>> "Hello, they call me "Jeff""

SyntaxError: invalid syntax

>>> "Hello, they call me \"Jeff\""

Output 🡪 'Hello, they call me "Jeff"'

Exercise on holypython.com (3)🡪#Type a couple of words or a short sentence for your variable, then print it.

my\_reason\_for\_coding= **str**("its a challenge")

print(my\_reason\_for\_coding)

#Assign True or False to the variable below then print it.

var1=True

var2=False

staying\_alive=var1

print(staying\_alive)

type(), type() function shows the type of a variable.

int(), numbers

float() ----,

str() , letter and numbers of any length

Exercise on holypython.com (4)🡪

1men\_stepped\_on\_the\_moon=12

2

​ 3

#Type your code here.

4

​ 5 answer\_1=type(men\_stepped\_on\_the\_moon)

6

7

8 print(answer\_1)

9

10

​ 11

​

======<type **'int'**>

1# my\_reason\_for\_coding="intergalactic travel"

2

3

#Type your code here.

4

5 answer\_2 = type(my\_reason\_for\_coding)

6

7

8 print (answer\_2)

9

10

11

<type **'str'**>

1global\_mean\_sea\_level\_delta\_2018=21.36

2

3#Type your code here.

4

5 answer\_3 = type(global\_mean\_sea\_level\_delta\_2018)

6

7

8 print(answer\_3)

9

10

11

​

=========<type **'float'**>

Now let's convert a string into a float.

#shoe\_price variable is a **string (because it's in quotes**). On line 9, convert it into a **float**.

-shoe\_price="69.99"

-answer\_7 = **float**(shoe\_price)

-print(answer\_7)

Finally, str will help you convert a data into a string.

#GWP denotes the total economic activity created by the world population collectively in a year.

-gross\_world\_product=84.84

-gwp\_str = str(gross\_world\_product)

-answer\_8="In 2018 gross product of the world (GWP) was **"** **+ gwp\_str +** **"** in trillion US dollars."

-print(answer\_8)

======= In 2018 gross product of the world (GWP) was 84.84 in trillion US dollars.

DAY 2

Exercise on holypython lists

6a

Assign the first element of the list to answer\_1 on line 2

lst=[11, 100, 99, 1000, 999]

answer\_1=

print(answer\_1)

answer\_1 = lst[0]

-> 🡪every item in the list has a value, starting at 0 (=11 in this case)

And let's print the second element directly inside print function.

This time print the second element of the list directly on line 3. You should get 100.

lst=[11, 100, 101, 999, 1001]

print()

print(lst[1])

returns the result of 100 ------ its got the index of 1 (11 =0)

using -**1** starts the list form the other end. You then count downwards

.append method will let you add items to your lists.

On line 3, add the string "pajamas" to the list with .append() method.

gift\_list=['socks', '4K drone', 'wine', 'jam']

gift\_list.append('pajamas')

print(gift\_list) **returns['socks', '4K drone', 'wine', 'jam', 'pajamas']**

Lists can hold many type of data inside them. You can even add another list to a list as its element. This is called nested data in Python.

#On line 3, this time add the sub-list: ["socks", "tshirt", "pajamas"] to the end of the gift\_list.

gift\_list=['socks', '4K drone', 'wine', 'jam']

# Type your code here.

gift\_list.append(["socks","tshirt","pajamas"])

print(gift\_list) **returns ['socks', '4K drone', 'wine', 'jam', ['socks', 'tshirt', 'pajamas'**]]

……..FYI Actual: ['socks', '4K drone', 'wine', 'jam', ['socks', 'tshirt', 'pajamas']]

Expected: ['socks', '4K drone', 'wine', 'jam', ['socks', 'tshirt', 'pajamas']] ------ after using ‘gift\_list.append’

.insert() lets you specify the index you want to add your item.

On line 3, this time insert "slippers" to index 3 of gift\_list.

gift\_list=['socks', '4K drone', 'wine', 'jam']

# Type your code here.

print(gift\_list)

**HINT 1**

.insert() takes 2 parameters:

1st: index number  
2nd: element you’re adding:

list.insert(index, element)

**HINT 2**

Remember indexing in Python starts with 0.

So, first item in a list is index 0, second item in a list is index 1 and so on.

code: gift\_list.insert(2, “slippers”) this goes in after wine as that’s at index 2

gift\_list=['socks', '4K drone', 'wine', 'jam']

gift\_list.insert(2, "slippers")

print(gift\_list)

RETURNS ['socks', '4K drone', 'slippers', 'wine', 'jam']

Actual: ['socks', '4K drone', 'wine', 'jam']

Expected: ['socks', '4K drone', 'slippers', 'wine', 'jam']

|  |
| --- |
|  |

With .index() method you can learn the index number of an item inside your list.

Assign the index no of 8679 to the variable answer\_1.

lst=[55, 777, 54, 6, 76, 101, 1, 2, 8679, 123, 99]

# Type your code here.

answer\_1= answer\_1=lst.index(8679)

print(answer\_1)

code: answer\_1=lst.index(8679)

result: 8

Using .append() method, add a new list to the end of the list which contains strings: "Navigator" and "Suburban".

lst=["CRV", "Outback", "XC90", "GL", "Cherokee", "Escalade"]

# Type your code here.

print(lst)

**HINT 1**

.append() adds an item to the end of a list.

It’s very commonly used in programming and it can be very useful.

lst=["CRV", "Outback", "XC90", "GL", "Cherokee", "Escalade"]

# Type your code here.

lst.append(["Navigator", "Suburban"])

print(lst)

code: lst.append(["Navigator", "Suburban"])

result: ['CRV', 'Outback', 'XC90', 'GL', 'Cherokee', 'Escalade', ['Navigator', 'Suburban']]

Actual: ['CRV', 'Outback', 'XC90', 'GL', 'Cherokee', 'Escalade']

Expected: ['CRV', 'Outback', 'XC90', 'GL', 'Cherokee', 'Escalade', ['Navigator', 'Suburban']]

**Conditionals**

In this module we will cover the use of conditionals in Python, looking at if , elif, and else statements.

**The Use of Conditionals**

At this point, a lot of our programming has been *step by step* or one instruction after another.  
What happens when we have *branching paths*? What if the program has more than one possible outcome?  
Conditional statements are used for this, to accommodate for different paths a program may take.

We use if statements to run a block of code if the condition is met.  
You can chain if - else statements to build more complex conditional statements.

my\_boolean = False

**if** my\_boolean:

**print**("My boolean is truthy!")

**else**:

**print**("My boolean is not truthy...")

In this program we have defined a variable, my\_boolean, and set it to False.

The if statement will check if this variable evaluates to True, but since it is False, this *condition* is not met.

It moves onto the next block in the chain, being the else statement. The else statement will always trigger when all if and elif fail, so it will execute the block, and print "My boolean is not truthy".

**Syntax**

The general syntax of an if statement is:

**if** some\_condition:

...

**else**:

...

Where the condition must be followed by a :, and anything we want to run when the condition is met must be indented.

If the block is not indented, Python will not understand what needs to be done when the condition is met.

If we had nested if statements, then we increase the indentation for each block:

**if** some\_condition:

**if** some\_other\_condition:

...

**else**:

...

**else**:

...

Notice that we only ever have an elsestatement after an if statement - we cannot do this the other way round, and cannot have a lone else statement.

**Comparators**

Booleans are great, but sometimes we need to compare multiple values.  
We do this using comparators.

Some comparators include:

* Equal to - ==
* Not equal to - !=
* Less than - <
* Less than or equal to - <=
* Greater than - >
* Greater than or equal to >=

Using these comparators, we can now do things like:

my\_money = 10

**if** my\_money > 10:

**print**("You're pretty rich!")

**else**:

**print**("You're quite poor.")

We can even chain together multiple comparators into a single condition:

deposit = 10

**if** 0 < deposit <= 100:

**print**(f"Thank you for the £{deposit} deposit!")

**else**:

**print**("This is not a valid amount to deposit.")

**Logical Operators**

We can chain together multiple conditions using the and / or operators, and reverse conditions using the not operator:

* and requires both of the conditions to be met.
* or requires either of the conditions to be met.
* not will reverse the condition, making True into False, and vice versa.

For example:

deposit = 10

password = "password"

**if** 0 < deposit <= 100 **and** password == "password":

**print**(f"Thank you for depositing £{deposit}!")

**else**:

**print**("Failed to make a deposit.")

or equivalently,

deposit = 10

password = "password"

**if** **not** (0 < deposit <= 100) **or** password != "password":

**print**("Failed to make a deposit.")

**else**:

**print**(f"Thank you for depositing £{deposit}!")

The in keyword works in a similar way with its counterpart, not in:

name = "Ollie"

**if** name **in** ("root", "admin"):

print("This is not a valid username!")

**else**:

print(f"Welcome, {name}!")

and,

name = "Ollie"

**if** name **not** **in** ("root", "admin"):

print(f"Welcome, {name}!")

**else**:

print("This is not a valid username!")

**elif**

We don't always need to check if every ifstatement evaluates to True.  
In a lot of cases, only one outcome can be True.  
So to make our code more efficent, we have elif statements.  
elif short for else if will run the following code block only if no other if statements have been evaluated as True.

age = **int**(**input**("Enter your age: "))

**if** age >= 85:

**print**("You are above 85")

elif age >= 50:

**print**("You are between 50 and 85")

elif age >= 20:

**print**("You are between 20 and 50")

**else**:

**print**("You are below 20 years old")

Notice how the elif statements do not need to specify if the age is below 85, as if it wasn't the previous if statement would've been evaulated as True.

**Tutorial**

Create a new Python file and type the following lines out.

devs\_money = 100

dev\_can\_play\_smash = False

Here we are assigning 2 variables and storing the data, 100 and False inside them.

**if** devs\_money > 10 **and** dev\_can\_play\_smash:

**print**("Dev enters a smash tournament!")

Here we a starting an if statement with the conditional being, if devs\_money is bigger than 10 and if dev\_can\_play\_smash is True.

**elif** devs\_money < 10 and dev\_can\_play\_smash:

print("Dev is too poor to enter")

Here we add an elif statement, so that if the previous conditional is False as well as devs\_money is less than 1 and dev\_can\_play\_smash is True the program will print "Dev is too poor to enter"

**else**:

**print**("Dev just can't play smash")

Now if all the previous conditionals have been False we execute the else statement.

Tutorial:

money = 10

**if** money > 10:

print("You're pretty rich!")

**else**:

print("You're quite poor.")

We can even chain together multiple comparators into a single condition:

deposit = 10

**if** 0 < deposit <= 100:

**print**(f"Thank you for the £{deposit} deposit!")

**else**:

**print**("This is not a valid amount to deposit.")

money = 11

if money > 10:

print("You're pretty rich!")

else:

print("You're quite poor.")

deposit = 140

if 1 < deposit **and** deposit <= 100:

~~#if 0 < deposit <= 100:~~

print(f"Thank you for the £{deposit} deposit!")

else: print("This is not a valid amount to deposit.")

Create a new Python file and type the following lines out.

devs\_money = 100

dev\_can\_play\_smash = False

Here we are assigning 2 variables and storing the data, 100 and False inside them.

**if** devs\_money > 10 **and** dev\_can\_play\_smash:

**print**("Dev enters a smash tournament!")

Here we a starting an if statement with the conditional being, if devs\_money is bigger than 10 and if dev\_can\_play\_smash is True.

**elif** devs\_money < 10 and dev\_can\_play\_smash:

print("Dev is too poor to enter")

Here we add an elif statement, so that if the previous conditional is False as well as devs\_money is less than 1 and dev\_can\_play\_smash is True the program will print "Dev is too poor to enter"

**else**:

**print**("Dev just can't play smash")

Now if all the previous conditionals have been False we execute the else statement.

**Code:**

1. devs\_money = 100
2. dev\_can\_play\_smash = False
3. **if** devs\_money > 10 **and** dev\_can\_play\_smash:
4. **print**("Dev enters a smash tournament!")
5. **elif** devs\_money < 10 **and** dev\_can\_play\_smash:
6. **print**("Dev is too poor to enter")
7. **else**:
8. **print**("Dev just can't play smash")

**result**:

Dev just can't play smash

Mark = int(input(“what grade did I receive”))

If mark >= 85:

Print (“distinction”)

Elif mark*< 85 and* mark >= 65:

Else: print(“fail”)

FIND OUT ABOUT LOOP….WATCH THE VIDEO BACK FROM TUESDAY!!

functions

Def lovechoc(inputvar):

Print (name + “l loves choc”)

Name = “Ainsley”

Lovechoc(name)

def soda():

    print('Kel loves orange soda')

var1 = input('Who loves orange soda? ')

if var1 == 'kel':

    soda()

else:

    print(var1 + ' loves orange soda')

def grading(inputvar):

if var1 > 85:

return “a”

elif var1 > 65 :

return “b”

else:

return “f”

typedinvar = input(“type: “)

var 1 = 90

gradvar = grading(var1)

print (“-------------------“)

print (“ you got an: “)

print (“ “ + gradvar “ “)

print (“-------------------“)

setting a function

CODE:def greet(first\_name, last\_name):

print (f"hi {first\_name},{last\_name} ")

print ("welcome aboard")

greet("A", "D")

greet("S", "T")

output: hi A,D

welcome aboard

hi S,T

welcome aboard

multiply code:

def multiply(\*numbers):

total = 1

for number in numbers:

total \*=number

return total

print(multiply(2,3,4,5))

DOUBLE \*\*

Allows you to pass multiple key values into it and puts it into a dictionary

Code:

def save\_user(\*\*user):

print(user)

save\_user(id=1, name="Ainsley", age=35)

output: **{**'id': 1, 'name': 'Ainsley', 'age': 35**}**

adding square brackets allows you to get the details of that key

def save\_user(\*\*user):

print(user["name"])

save\_user(id=1, name="Ainsley", age=35)

OUTPUT: student@Students-MacBook-Air DFESW11py % /usr/loca

l/bin/python3 "/Users/student/Desktop/ains Course/

python learning/DFESW11py/DFESW11py/condi.py"

**Ainsley**

def myFave(drink):

if drink == "water":

var1 = drink + " best drink on the land"

return var1

else:

var1 = "this is the wrong choice"

return var1

print(myFave(input("whats your choice of drink ")))

def greet(name):

return f"Hello {name}"

name = input("What is your name? ")

print(greet(name))

when I input my name it returns Hello “Name”

def sum(numbers):

total = 0

for x in numbers:

total += x

return total

print(sum((8, 2, 3, 0, 7)))

num1 = 18

num2 = 50

num3 = 2

lst = [num1, num2, num3]

if (num1 >=num2,num3):

largest = num1

elif (num2 >= num1,num3):

largest = num2

else:

largest = num3

print("The largest of the 3 numbers is : ", max(lst))

code : gives you the largest number form a list using IF and ELIF and ELSE

def multiply(numbers):

total = 1

for x in numbers:

total \*= x

return total

print(multiply((2,2,2,2)))

multiply numbers

num1 = 10

num2 = 2

num3 = 5

lst = [num1, num2, num3]

total = 1

for x in lst:

total= total\*x

print(total)

multiply numbers form a list

def reverse(itr):

return itr[::-1]

str1 = '1234abcd'

print("Original string:",str1)

print("Reverse string:",reverse('1234abcd'))

str1 = 'reverse'

print("\nOriginal string:",str1)

print("Reverse string:",reverse(str1))

reverse a string

find out what itr and :: are ?????

import math

number = float(input("Enter a number: "))

answer = math.sqrt(number)

print(answer)

imported math function in python and used sqrt to find answer

def string\_test(x):

d={"UPPER\_CASE":0, "LOWER\_CASE":0}

for c in x:

if c.isupper():

d["UPPER\_CASE"]+=1

elif c.islower():

d["LOWER\_CASE"]+=1

else:

pass

print ("Original String : ", x)

print ("No. of Upper case characters : ", d["UPPER\_CASE"])

print ("No. of Lower case Characters : ", d["LOWER\_CASE"])

string\_test('he Loud child woKe UP his slEEping DoG')

number of upper and lower case in text

pick up on modules in python

python3 -m pip install – install pip programmes