

Sprint 00

Marathon Python

April 9, 2021



Contents



Engage	2
Investigate	3
Act Basic: Task 00 > Hello!	5
Act Basic: Task 01 > Data types	6
Act Basic: Task 02 > Variables	7
Act Basic: Task 03 > Concatenate	9
Act Basic: Task 04 > Bridge of Death	10
Act Basic: Task 05 > Math operations	12
Act Basic: Task 06 > Casting	14
Act Basic: Task 07 > Condition	16
Act Advanced: Task 08 > What is the address?	18
Act Advanced: Task 09 > Get rid of it	20
Act Advanced: Task 10 > Do op	21
Act Advanced: Task 11 > Bot	22
Share	24

Engage



DESCRIPTION

Welcome to the `Python` programming language!

Python is one of those rare languages that can claim to be both simple and powerful. You will be surprised how easy it is to concentrate on the solution to the problem rather than the syntax and structure of the language you are programming in. Python can be used for things like:

- backend (or server-side) web and mobile app development
- desktop app and software development
- processing big data and performing mathematical computations
- writing system scripts (creating instructions that tell a computer system to "do" something)

But don't let Python's broad range scare you. Python is an easy to learn, in-demand programming language that can exponentially increase your chances of getting hired and increasing your income in a matter of months.

But first, you need to start with the basics. Therefore, in this challenge, you will get acquainted with variables, numbers, strings, operators, and other topics.

Let's start!

BIG IDEA

Introduction to Python.

ESSENTIAL QUESTION

How to start programming with Python?

CHALLENGE

Learn the basics of Python.

Investigate



GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What are Python scripts?
- How to execute a `Python` script?
- What does it mean when people say that Python is an `interpreted programming language`?
- What areas of programming is Python best suited for?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- What is it? Why is Python so popular?
- Watch `The Story of Python`.
- Prepare your environment for development. Make sure that you have `Python` with a `3.8` version, or higher. If not consider downloading it from `the official Python website`.
- Find some resources with articles about Python to gain the latest info about it (e.g. `Medium`).
- Learn about `different Python interpreters`.
- Open the terminal and create a Python script called `main.py` containing a line `import this`. Then, run the command `python3 main.py`. Here is how to do this in the console:

```
>touch main.py
>echo 'import this' > main.py
>cat main.py
import this
>python3 main.py
The Zen of Python, by Tim Peters

...
>
```

- Enjoy the zen.
- Then let's try the same thing in the Python Interpreter. Stay in the terminal and do the following:

```
>python3
>>> import this
The Zen of Python, by Tim Peters

...
>>>
```



- You got exactly the same result as when running the script.
- Attentively watch and investigate learning videos available on the challenge page. Try to repeat all actions.
- Clone your git repository issued on the challenge page in the LMS.
- Proceed with tasks.

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story.
- All tasks are divided into **Act Basic** and **Act Advanced**. You need to complete all basic tasks to validate the **Sprint**. But to achieve maximum points, consider accomplishing advanced tasks also.
- Analyze all information you have collected during the preparation stages. Try to define the order of your actions.
- Perform only those tasks that are given in this document.
- Submit only those files that are described in the story. Only useful files allowed, garbage shall not pass!
- Run the scripts using `python3`.
- Make sure that you have **Python** with a **3.8** version, or higher.
- Use the standard library available after installing **Python**. You may use additional packages/libraries that were not previously installed only if they are specified in the task.
- Complete tasks according to the rules specified in the **PEP8 conventions**.
- The solution will be checked and graded by students like you. **Peer-to-Peer learning**.
- Also, the challenge will pass automatic evaluation which is called **Oracle**.
- If you have any questions or don't understand something, ask other students or just Google it.

Act Basic: Task 00



NAME

Hello!

DIRECTORY

```
t00_hello/
```

SUBMIT

```
hello_world.py
```

LEGEND

- Who's that then?
- I dunno, must be a king.
- Why?
- He hasn't got sh** all over him.

-- *Monty Python and the Holy Grail*

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- How to print text using Python?
- How to run Python scripts?
- What is the difference between single and double quotes in Python?

DESCRIPTION

Create a script that prints the message below to the console:

```
'Hello world! It is I, Arthur, son of Uther Pendragon, from the castle of Camelot.'
```

See how your script must work in the section [CONSOLE VIEW](#).

CONSOLE VIEW

```
>python3 hello_world.py
Hello world! It is I, Arthur, son of Uther Pendragon, from the castle of Camelot.
>
```

SEE ALSO

[Python print\(\) Function](#)
[Single and Double Quotes | Python](#)

Act Basic: Task 01



NAME

Data types

DIRECTORY

```
t01_data_types/
```

SUBMIT

```
data_types.py
```

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- What are data types?
- What are data types for?
- What data types are there in Python?
- What is the difference between values: `0`, `'0'`, and `0.0`?

DESCRIPTION

Create a script that prints values in different data types:

- integer with a value `50`
- float with a value `-42.00001`
- string with a value `'Monty Python'`
- boolean with a value `True`

Your output must be like in the [CONSOLE VIEW](#).

CONSOLE VIEW

```
>python3 data_types.py
50
-42.00001
Monty Python
True
>
```

SEE ALSO

[The Python Standard Library Data Types](#)
[Python Data Types](#)

Act Basic: Task 02



NAME

Variables

DIRECTORY

```
t02_variables/
```

SUBMIT

```
var.py
```

LEGEND

```
- All right, but apart from the sanitation, the medicine, education, wine, public order,  
irrigation, roads, the fresh-water system, and public health, what have the Romans ever  
done for us?  
- Brought peace?  
- Oh, peace? SHUT UP!
```

```
-- Monty Python's Life of Brian
```

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- What is a variable in programming?
- What is the advantage of using variables to store values?
- How to define a variable?
- How to determine the data type of a variable?
- How to import a variable from a script?
- What characters can you use in a variable name?
- Which of these cannot be the first character of a variable name: uppercase letter, digit, lowercase letter, underscore?

There are good and bad ways to name a variable, but the main goal is to make your code readable and understandable. Find out what are the best practices of variable naming, and what are the conventions for variables in Python, according to [PEP 8](#).

DESCRIPTION

Create a script that:

- defines the following variables:
 - an integer called `my_int` with the value `1`
 - a float called `my_float` with the value `2.123`
 - a string called `my_str` with the value `'Tis but a scratch.'`
 - a boolean called `my_bool` with the value `False`



- prints the types of each of these variables using the function `type()` in the correct order

Your output must look like the **CONSOLE VIEW**. You can also find the output in the **resources** for the challenge.

The **PYTHON INTERPRETER** section demonstrates a way to check if your script works correctly on the example of the float. As you can see, we have imported the variables from the script. Once imported, the script was executed and generated the same output as the **CONSOLE VIEW**, but that's not what we're interested in. The next command: `>>> type(my_float)` outputs the type of the variable `my_float`. If your script has defined the variable correctly, the output must be `<class 'float'>`. You can test other variables using the same approach.

CONSOLE VIEW

```
>python3 var.py
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
>
```

PYTHON INTERPRETER

```
>python3
>>> from var import my_int, my_float, my_str, my_bool
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
>>> type(my_float)
<class 'float'>
>>> my_float
2.123
>>>
```

SEE ALSO

[Python Variables](#)
[Python Data Types](#)

Act Basic: Task 03



NAME

Concatenate

DIRECTORY

t03_concatenate/

SUBMIT

concatenate.py

BEFORE YOU BEGIN

- How to concatenate strings using an operator? What operator is it?
- What other methods of concatenation are available in Python?

DESCRIPTION

Create a script that:

- contains a variable `first_name` with a value `'Monty'`
- contains a variable `last_name` with a value `'Python'`
- contains a variable `full_name` that must store the result of concatenation of `first_name` and `last_name` connected with a space `' '`
- prints the variable `full_name`

Your output must be like in [CONSOLE VIEW](#).

CONSOLE VIEW

```
>python3 concatenate.py
Monty Python
>
```

SEE ALSO

[Python String Concatenation](#)

Act Basic: Task 04



NAME

Bridge of Death

DIRECTORY

```
t04_bridge_of_death/
```

SUBMIT

```
input.py
```

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- How to get input from the user?
- How to format a string with f-strings?

Let's try to apply the new knowledge.

Start the Python Interpreter and repeat the following steps (follow line by line, include your name).

PYTHON INTERPRETER

```
>python3
>>> name = input('Enter your name: ')
Enter your name: ucoder
>>> f'Hello {name}'
'Hello ucoder'
>>> quit()
>
```

DESCRIPTION

Create a script that:

- reads several variables from the `stdin` using the `input()` function
 - name
 - quest
 - color
- prints the entered information

In the **CONSOLE VIEW** example, the user-entered values are:

- **Sir Lancelot of Camelot** on the question 'What is your name?'
- **to seek the Holy Grail** on the question 'What is your quest?'
- **blue** on the question 'What is your favorite color?'

The full output can be found in the challenge **resources** in the LMS.

Don't hardcode the output, values must only be accepted via `input()`.



CONSOLE VIEW

```
>python3 input.py
BRIDGEKEEPER: Stop.
Who would cross the Bridge of Death must answer me these questions three.
BRIDGEKEEPER: What is your name?
Sir Lancelot of Camelot
BRIDGEKEEPER: What is your quest?
to seek the Holy Grail
BRIDGEKEEPER: What is your favorite color?
blue
-----
Traveler info:
Your name is Sir Lancelot of Camelot
Your quest is to seek the Holy Grail
Your favorite color is blue
-----
BRIDGEKEEPER: Right. Off you go.
>
```

SEE ALSO

[Python input\(\) Function](#)

[How To Use f-strings to Create Strings in Python 3](#)

Act Basic: Task 05



NAME

Math operations

DIRECTORY

```
t05_math_operations/
```

SUBMIT

```
math.py
```

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- What arithmetic operators exist in Python?
- What is the difference between these two division operators: `/` and `//`?
- How can this operator `%` help determine whether a number is odd or even?

DESCRIPTION

Create a script that (the first two requirements were done for you):

- asks to enter two numbers in sequence
- stores values in variables `a` and `b` respectively, and casts them to type `integer` (you can find an example of casting below in the [SYNOPSIS](#))
- performs primitive math operations on them and stores the result of each operation in variable `c`
 - addition (+)
 - subtraction (-)
 - multiplication (*)
 - division (/)
 - modulus (%)
 - exponent (**)
 - floor division (//)
- prints the result for each operation

You must write your implementation with operations and their output to get the result as in the [CONSOLE VIEW](#).

Use `f-strings` to format strings.



SYNOPSIS

```
a = input('Enter the first number: ')\nb = input('Enter the second number: ')\na = int(a)\nb = int(b)
```

```
# write your code here
```

CONSOLE VIEW

```
>python3 math.py\nEnter the first number: 7\nEnter the second number: 4\n7 + 4 = 11\n7 - 4 = 3\n7 * 4 = 28\n7 / 4 = 1.75\n7 % 4 = 3\n7 ** 4 = 2401\n7 // 4 = 1\n>python3 math.py\nEnter the first number: 33\nEnter the second number: 4\n33 + 4 = 37\n33 - 4 = 29\n33 * 4 = 132\n33 / 4 = 8.25\n33 % 4 = 1\n33 ** 4 = 1185921\n33 // 4 = 8\n>
```

SEE ALSO

[Python Arithmetic Operators Example](#)
[f-Strings](#)
[Python Casting](#)

Act Basic: Task 06



NAME

Casting

DIRECTORY

t06_casting/

SUBMIT

cast.py

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- How can casting be useful in programming?
- What integers do the values `True` and `False` represent in numerical contexts?
- Does Python allow implicit type conversions?
- What error does this code produce: `'1' + 1`?
- If you define a variable as an integer (`a = 5`), is it possible to later assign it a value of a different data type? (e.g., like this: `a = 'now a string'`)
- What happens if you subtract a boolean from an integer?

DESCRIPTION

Create a script that:

- performs math operations with variables:

```
a = 3
b = 10
c = -14.8
d = True
```
- casts the variables or their math operations results into different data types (use the code snippet in the **SYNOPSIS**, but instead of `...` put the correct keywords to get the needed result)
- prints operations and results to the console

The output must look like **CONSOLE VIEW**.

Use the lines given in the **SYNOPSIS** as they are (with keywords replaced). Add your code around and/or in-between the given lines to achieve the required output.

SYNOPSIS

```
result = float(a + b)
result = ...(c - b)
result = ...(c) + ...(b)
result = ...(a - a)
```




CONSOLE VIEW

```
>python3 cast.py
Available variables:
a = 3
b = 10
c = -14.8
d = True

Result:
3 + 10 = 13.0
-14.8 - 10 = -24
-14.8 + 10 = -14.810
3 - 3 = False
>
```

SEE ALSO

[Python Casting](#)

Act Basic: Task 07



NAME

Condition

DIRECTORY

t07_condition/

SUBMIT

directions.py

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- What is a condition in programming?
- What are comparison operators (also known as relational operators)?
- What value does this conditional expression evaluate to: `5 == 3`?
- How to write an `if` statement?
- What is the difference between the keywords `else` and `elif`?

DESCRIPTION

Create a script that asks the user for input and outputs a message that will be different depending on the input.

If the user enters the word 'right', the output message must be 'The right sign says: "DEAD PEOPLE ONLY"'.
If the user enters the word 'left', the output message must be 'The left sign says: "BEWARE!"'.
If the user enters the word 'middle', the output message must be 'The middle sign says: "CERTAIN DEATH"'.
In all other cases the output message must be 'There is no [entered input] sign'.

See the [CONSOLE VIEW](#) for examples of how your script must work. The contents of this section are also available in the [resources](#).

CONSOLE VIEW

```
>python3 directions.py
There are 3 signs in front of you. Which one would you like to read? right
The right sign says: "DEAD PEOPLE ONLY"
>python3 directions.py
There are 3 signs in front of you. Which one would you like to read? left
The left sign says: "BEWARE!"
>python3 directions.py
There are 3 signs in front of you. Which one would you like to read? middle
The middle sign says: "CERTAIN DEATH"
```



```
>python3 directions.py
There are 3 signs in front of you. Which one would you like to read? top
There is no top sign
>
```

SEE ALSO

[Python Conditional Statements](#)
[Relational Operators Python](#)

Act Advanced: Task 08



NAME

What is the address?

DIRECTORY

```
t08_what_is_the_address/
```

SUBMIT

```
address.py
```

LEGEND

BORING PROPHET:

And there shall in that time be rumours of things going astray, and there will be a great confusion as to where things really are, and nobody will really know where lieth those little things with the sort of raffia work base, that has an attachment...at this time, a friend shall lose his friend's hammer and the young shall not know where lieth the things possessed by their fathers that their fathers put there only just the night before around eight o'clock...

-- Monty Python's Life of Brian

BEFORE YOU BEGIN

In order to complete this task, you must be able to answer the following questions:

- How does Python store `variables` in memory?
- What is the Python `id` function, and when should we use it?
- In what ways can you check the identicalness of variables in programming?

DESCRIPTION

Create a script that contains three integer variables and does the following:

- assigns the value `1000` to the first variable
- assigns the value of the first variable to the second variable
- defines a third integer variable with the value `999`
- prints variable addresses using the `id` function
- compares address of the first variable to the second and third with keyword `is`
- prints the result to the console

Do not be alarmed that addresses may differ from the **CONSOLE VIEW**.



CONSOLE VIEW

```
>python3 address.py
first_var = 1000, address is 4529357712
second_var = 1000, address is 4529357712
third_var = 999, address is 4527480496
1000 is 1000 = True
1000 is 999 = False
>
```

SEE ALSO

[Variables and Memory Addresses in Python](#)
[is keyword in Python](#)

Act Advanced: Task 09



NAME

Get rid of it

DIRECTORY

```
t09_get_rid_of_it/
```

SUBMIT

```
rid.py
```

BEFORE YOU BEGIN

There is a keyword in Python that can be used to get rid of an object (a variable, function, class, etc.). This keyword unbinds the name of that object. So, if you call this keyword on a variable, you will no longer be able to use it. Find this keyword, you will need it in this task.

Tip: you can see the list of all Python keywords inside your console. In the Python Interpreter, run the command `help('keywords')`. This will show you the list. To get more information on a particular keyword, run the command with the keyword you need, e.g., `help('await')`.

DESCRIPTION

There is a script in the **SYNOPSIS** (also available in the **resources**). There is one line missing, marked by the comment. Replace the comment line with exactly one line of code that uses a particular Python keyword on your variable, so that, when the script runs, a **NameError** exception is raised. Do not edit anything except for the one line marked to be replaced.

Your output must match the **CONSOLE VIEW**.

SYNOPSIS

```
my_number = 1
print(my_number)
# replace me with code
print(my_number)
```

CONSOLE VIEW

```
>python3 rid.py
1
Traceback (most recent call last):
  File "rid.py", line 4, in <module>
    print(my_number)
NameError: name 'my_number' is not defined
>
```

Act Advanced: Task 10



NAME

Do op

DIRECTORY

t10_do_op/

SUBMIT

do_op.py

DESCRIPTION

Create a calculator script that:

- sequentially reads two integer numbers and an operator (addition (+), subtraction (-), multiplication (*), or division (/)) from the console input
- checks the correctness of the operator and prints usage
"usage: the operator must be '*' or '+' or '-' or '/'"
in case of an invalid operator
- then prints the result of the operation with the two given numbers

CONSOLE VIEW

```
>python3 do_op.py
---- Simple calculator ----
Let's add some numbers
Input your first value: 2
Input your operator: +
Input your second value: 2
2 + 2 = 4
---- Simple calculator ----
>python3 do_op.py
---- Simple calculator ----
Let's add some numbers
Input your first value: 4
Input your operator: ++
usage: the operator must be '*' or '+' or '-' or '/'
---- Simple calculator ----
>
```

Act Advanced: Task 11



NAME

Bot

DIRECTORY

```
t11_bot/
```

SUBMIT

```
bot.py
```

LEGEND

Nobody expects the Spanish Inquisition! Our chief weapon is surprise! Surprise and fear. Fear and surprise. Our two weapons are fear and surprise - and ruthless efficiency! Our three weapons are fear, and surprise, and ruthless efficiency, and an almost fanatical devotion to the Pope. Our four, no, among our weapons are such elements as fear, surpr- I'll come in again.

-- Monty Python's Flying Circus

DESCRIPTION

Create your own bot-like program.
The bot must:

- read two strings. If both strings are empty, or one of the strings is empty - print a message
`'One of the strings is empty.'`
- read a command to run from a predefined list of commands (`'find'`, `'concat'`, `'beatbox'`). If the command is not present in the list of predefined commands - print a message
`'usage: command find | concat | beatbox'.`
The work of commands:
 - `'find'` - prints `True` if the second string is in the first string and `False` if not
 - `'concat'` - prints a message to the screen by concatenating the first string and second string with a space
 - `'beatbox'` - reads two integers, prints the concatenation of the first string times the first number and the second string times the second number.

Look at the examples in the [CONSOLE VIEW](#).

CONSOLE VIEW

```
>python3 bot.py
Enter your first string:
Enter your second string:
One of the strings is empty.
>python3 bot.py
Enter your first string: hello
Enter your second string:
```




```
One of the strings is empty.
>python3 bot.py
Enter your first string: qwe
Enter your second string: qwe
Enter your command: qwe
usage: command find | concat | beatbox
>python3 bot.py
Enter your first string: hello
Enter your second string: world
Enter your command: concat
Your string is: hello world
>python3 bot.py
Enter your first string: hello
Enter your second string: world
Enter your command: find
False
>python3 bot.py
Enter your first string: helloooo
Enter your second string: hello
Enter your command: find
True
>python3 bot.py
Enter your first string: qw
Enter your second string: e
Enter your command: beatbox
Enter your first beatbox number: 3
Enter your second beatbox number: 4
qwqwweeee
>
```

Share



PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences. During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- [Canva](#) - a good way to visualize your data
- [QuickTime](#) - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- [Facebook](#) - create and share a post that will inspire your friends
- [YouTube](#) - upload an exciting video
- [GitHub](#) - share and describe your solution
- [Telegraph](#) - create a post that you can easily share on Telegram
- [Instagram](#) - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use [#ucode](#) and [#CBLWorld](#) on social media.

