

# Part 1: Functions

**Exercise 1:** What is the purpose of the “def” keyword in Python?

- a) It is slang that means “the following code is really cool”
- b) It indicates the start of a function
- c) It indicates that the following indented section of code is to be stored for later
- d) b and c are both true
- e) None of the above

ANSWER: D

**Exercise 2:** What will the following Python program print out?

```
None
def fred():
    print("Zap")

def jane():
    print("ABC")

jane()
fred()
jane()
```

- a) Zap ABC jane fred jane
- b) Zap ABC Zap
- c) ABC Zap jane
- d) ABC Zap ABC
- e) Zap Zap Zap

ANSWER: D

**Exercise 3:** Rewrite your pay computation with time-and-a-half for overtime and create a function called `compute_pay` which takes two parameters (`hours` and `rate`).

```
None
Enter Hours: 45
Enter Rate: 10
```

Pay: 475.0

ANSWER:

```
def computepay(hours, rate):  
    if hours > 40:  
        regular_pay = 40 * rate  
        overtime_pay = (hours - 40) * (rate * 1.5)  
        total_pay = regular_pay + overtime_pay  
    else:  
        total_pay = hours * rate  
    return total_pay  
  
# Main program  
hrs = float(input("Enter Hours: "))  
rte = float(input("Enter Rate: "))  
pay = computepay(hrs, rte)  
print("Pay:", pay)
```

ONLINE PYTHON

Learn PythonTry New IDE

main.py

+

```
1 def computepay(hours, rate):
2     if hours > 40:
3         regular_pay = 40 * rate
4         overtime_pay = (hours - 40) * (rate * 1.5)
5         total_pay = regular_pay + overtime_pay
6     else:
7         total_pay = hours * rate
8     return total_pay
9
10 # Main program
11 hrs = float(input("Enter Hours: "))
12 rte = float(input("Enter Rate: "))
13 pay = computepay(hrs, rte)
14 print("Pay:", pay)
```

Ln: 13, Col: 27

RunShare\$

Command Line Arguments

Enter Hours: 45  
Enter Rate: 10  
Pay: 475.0

\*\* Process exited - Return Code: 0 \*\*

**Exercise 4:** Rewrite the grade program from the previous chapter using a function called `computegrade` that takes a score as its parameter and returns a grade as a string.

None

Score	Grade
<code>&gt;= 0.9</code>	A
<code>&gt;= 0.8</code>	B
<code>&gt;= 0.7</code>	C
<code>&gt;= 0.6</code>	D
<code>&lt; 0.6</code>	F

None

`Enter score: 0.95`  
A

None

Enter score: perfect

Bad score

None

Enter score: 10.0

Bad score

None

Enter score: 0.75

C

None

Enter score: 0.5

F

Run the program repeatedly to test the various different values for input.

ANSWER:

```
def computegrade(score):
```

```
    if score < 0.0 or score > 1.0:
```

```
        return "Bad score"
```

```
    elif score >= 0.9:
```

```
        return "A"
```

```
    elif score >= 0.8:
```

```
        return "B"
```

```
    elif score >= 0.7:
```

```
        return "C"
```

```
    elif score >= 0.6:
```

```
    return "D"
```

```
else:
```

```
    return "F"
```

```
# Test
```

```
try:
```

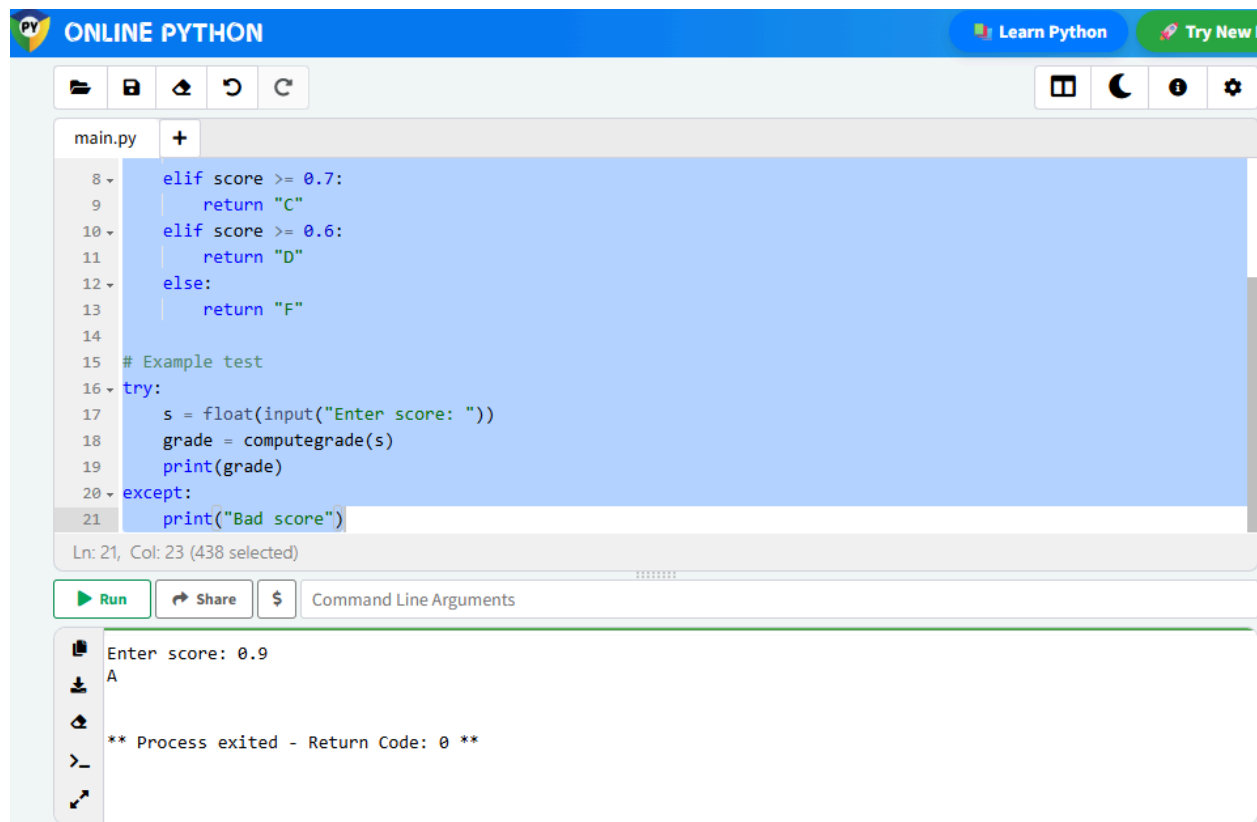
```
    s = float(input("Enter score: "))
```

```
    grade = computegrade(s)
```

```
    print(grade)
```

```
except:
```

```
    print("Bad score")
```



The screenshot shows the Online Python IDE interface. The top bar is blue with the Python logo and the text "ONLINE PYTHON". On the right, there are links for "Learn Python" and "Try New". Below the bar is a toolbar with icons for file operations (new, open, save, undo, redo) and window management (toggle, close, info, settings). The main editor area shows a file named "main.py" with the following code:

```
8 elif score >= 0.7:
9     return "C"
10 elif score >= 0.6:
11     return "D"
12 else:
13     return "F"
14
15 # Example test
16 try:
17     s = float(input("Enter score: "))
18     grade = computegrade(s)
19     print(grade)
20 except:
21     print("Bad score")
```

The status bar at the bottom of the editor indicates "Ln: 21, Col: 23 (438 selected)". Below the editor is a "Run" button, a "Share" button, and a "Command Line Arguments" input field. The output area at the bottom shows the execution results:

```
Enter score: 0.9
A
** Process exited - Return Code: 0 **
```

ONLINE PYTHON

Learn Python

Try New I

main.py

+

8

9

10

11

12

13

14

15

16

17

18

19

20

21

elif score >= 0.7:

return "C"

elif score >= 0.6:

return "D"

else:

return "F"

# Example test

try:

s = float(input("Enter score: "))

grade = computegrade(s)

print(grade)

except:

print("Bad score")

Ln: 15, Col: 15

Run

Share

\$

Command Line Arguments

Enter score: perfect  
Bad score  
  
\*\* Process exited - Return Code: 0 \*\*  
>\_

Enter score: 10.0  
Bad score  
  
\*\* Process exited - Return Code: 0 \*\*  
>\_

Enter score: 0.75  
C  
  
\*\* Process exited - Return Code: 0 \*\*  
>\_

Enter score: 0.5  
F  
  
\*\* Process exited - Return Code: 0 \*\*  
>\_

# Part 2: Loops and Iterations

**Exercise 1:** Write a program which repeatedly reads integers until the user enters “done”. Once “done” is entered, print out the total, count, and average of the integers. If the user enters anything other than a integers, detect their mistake using `try` and `except` and print an error message and skip to the next integers.

```
None
Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
16 3 5.333333333333333
```

ANSWER:

```
total = 0
```

```
count = 0
```

```
while True:
```

```
    num = input("Enter a number: ")
```

```
    if num == "done":
```

```
        break
```

```
    try:
```

```
        value = int(num)
```

```
        total += value
```

```
        count += 1
```

```
    except:
```

```
        print("Invalid input")
```

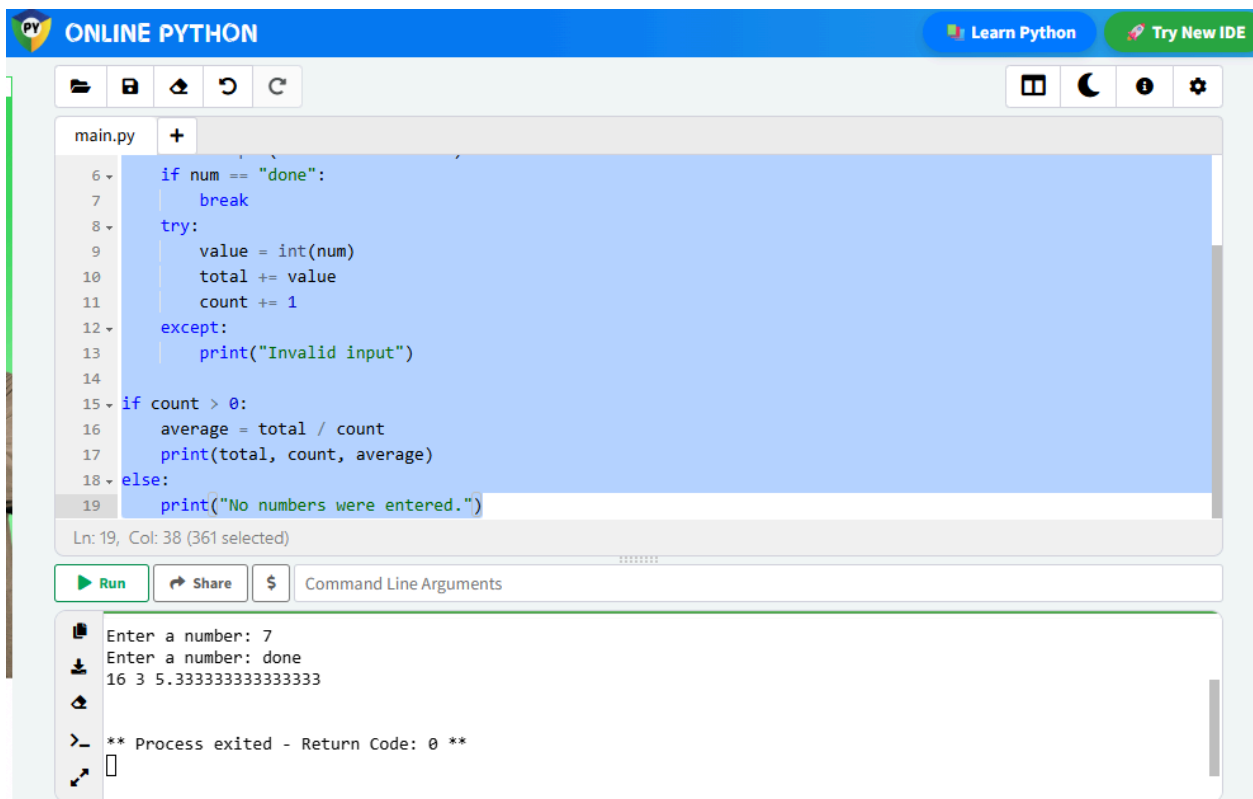
```
if count > 0:
```

```
    average = total / count
```

```
    print(total, count, average)
```

```
else:
```

```
    print("No numbers were entered.")
```



The screenshot shows the Online Python IDE interface. The top bar is blue with the 'ONLINE PYTHON' logo and buttons for 'Learn Python' and 'Try New IDE'. Below the bar is a toolbar with icons for file operations. The main editor area displays a Python script in a file named 'main.py'. The script includes a loop that prompts for numbers, calculates the average, and prints the results. The output window at the bottom shows the execution results, including the input numbers and the calculated average.

```
main.py +
6  if num == "done":
7      break
8  try:
9      value = int(num)
10     total += value
11     count += 1
12 except:
13     print("Invalid input")
14
15 if count > 0:
16     average = total / count
17     print(total, count, average)
18 else:
19     print("No numbers were entered.")
```

Ln: 19, Col: 38 (361 selected)

Run Share \$ Command Line Arguments

Enter a number: 7  
Enter a number: done  
16 3 5.333333333333333

\*\* Process exited - Return Code: 0 \*\*

**Exercise 2:** Write another program that prompts for a list of numbers as above and at the end prints out both the maximum and minimum of the numbers instead of the average.

ANSWER:

```
def find_min_max():
```

```
    numbers = []
```



```
while True:
```

```
    user_input = input("Enter a number: ")
```

```
    if user_input == "done":
```

```
        break
```

```
    try:
```

```
        number = float(user_input) # Use float to allow non-integers
```

```
        numbers.append(number)
```

```
    except ValueError:
```

```
        print("Invalid input")
```

```
        continue
```

```
if numbers:
```

```
    minimum = min(numbers)
```

```
    maximum = max(numbers)
```

```
    print("Minimum:", minimum)
```

```
    print("Maximum:", maximum)
```

```
else:
```

```
    print("No numbers were entered.")
```

```
find_min_max()
```

ONLINE PYTHON

Learn PythonTry New IDE

main.py

+

```
1 def find_min_max():
2     numbers = []
3
4     while True:
5         user_input = input("Enter a number: ")
6
7         if user_input == "done":
8             break
9
10        try:
11            number = float(user_input) # Use float to allow non-integers
12            numbers.append(number)
13        except ValueError:
14            print("Invalid input")
15            continue
```

Ln: 1, Col: 1

RunShare\$

Command Line Arguments

Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
Minimum: 4.0

Exercise 1.2 FunctionsBigBlueButton - CPEBigBlueButton - ROnline Python - IDE1.3.3 Functions: CPEHIPOLITO\_Exercise 1.2

online-python.com

ONLINE PYTHON

Learn PythonTry New IDE

main.py

+

```
1 def find_min_max():
2     numbers = []
3
4     while True:
5         user_input = input("Enter a number: ")
6
7         if user_input == "done":
8             break
9
10        try:
11            number = float(user_input) # Use float to allow non-integers
12            numbers.append(number)
13        except ValueError:
14            print("Invalid input")
15            continue
```

Ln: 1, Col: 1

RunShare\$

Command Line Arguments

Enter a number: 4
Enter a number: 5
Enter a number: bad data
Invalid input
Enter a number: 7
Enter a number: done
Minimum: 4.0

